Efficient development and execution of environmental applications on high performance parallel and distributed computing infrastructures

Rodila, Diana-Denisa

UNIVERSITÉ DE GENÈVE
Département F.-A. Forel des sciences de l'environnement et de l'eau

FACULTÉ DES SCIENCES
Docteur N. Ray
Professeur A. Lehmann

UNIVERSITÉ TECHNIQUE DE CLUJ-NAPOCA
Département d'Informatique

FACULTÉ D'INFORMATIQUE ET D'AUTOMATISATION
Professeur Ing. D. Gorgan

Efficient Development and Execution of Environmental Applications on High Performance Parallel and Distributed Computing Infrastructures

THÈSE

présentée à la Faculté des sciences de l'Université de Genève
pour obtenir le grade de Docteur ès sciences, mention Sciences de l'Environnement
et
présentée à la Faculté d'Informatique et d'Automatisation de l'Université Technique de Cluj-Napoca, pour obtenir le grade de Docteur ès sciences, mention Informatique

par
Diana Denisa RODILĂ
de
Cluj-Napoca (ROMANIA)

Thèse n^{o} 4942

GENÈVE
Atelier d'impression ReproMail

2016

UNIVERSITÉ
DE GENÈVE

FACULTÉ DES SCIENCES

Doctorat ès sciences
Mention environnement

En cotutelle avec UNIVERSITATEA TEHNICÁ DIN CLUJ-NAPOCA

Thèse de ***Madame Diana-Denisa RODILĂ***

intitulée :

"Efficient Development and Execution of Environmental Applications on High Performance Parallel and Distributed Computing Infrastructures"

La Faculté des sciences, sur le préavis de Monsieur A. LEHMANN, professeur associé et directeur de thèse (Institut F.-A. Forel, Institut des sciences de l'environnement), Monsieur D. GORGAN, professeur et codirecteur de thèse (Computer Science Department, Technical University of Cluj-Napoca, Romania), Monsieur N. RAY, docteur et codirecteur de thèse (Institut des sciences de l'environnement), Monsieur J.-L. FALCONE, docteur (Centre Universitaire d'Informatique), Monsieur L. C. MICLEA, professeur (Automation Departement, Technical University of Cluj-Napoca, Romania), Monsieur G. SEBESTYEN-PAL, professeur (Computer Science Department, Technical University of Cluj-Napoca, Romania) et Monsieur N. TAPUS, professeur (Computer Science & Engineering Department, University Politehnica of Bucharest, Romania), autorise l'impression de la présente thèse, sans exprimer d'opinion sur les propositions qui y sont énoncées.

Genève, le 16 juin 2016

Thèse - 4942 -

Le Doyen

N.B. - La thèse doit porter la déclaration précédente et remplir les conditions énumérées dans les "Informations relatives aux thèses de doctorat à l'Université de Genève".

I dedicate this thesis to my family, with all my love!

"Surely, in the light of history, it is more intelligent to hope rather than to fear, to try rather than not to try. For one thing we know beyond all doubt: Nothing has ever been achieved by the person who says, 'It can't be done!' "

(Eleanor Roosevelt)

"What counts in life is not the mere fact that we have lived. It is what difference we have made to the lives of others that will determine the significance of the life we lead."

(Nelson Mandela)

Abstract

Global environmental changes, linked to climate, biodiversity, environmental degradation, pollution, and others are threatening our planet more and more every day, with negative outcomes such as global sea level rise, intensified droughts, glaciers melting, etc. Environmental Sciences community and the human society at large need to find effective and operational responses to these complex changes. These actions imply more than ever a better understanding of the complex Earth system, the inter-linkage between its sub-systems and the impacts of human induced activities on natural phenomena. Such a challenge requires not only access to a huge amount of environmental data, from various disciplines and geographic scales, but also storing and processing resources as well as standardized performant tools, algorithms and services, able to extract useful and meaningful information from raw data, information which will support better decision making and better actions towards a sustainable development and a sustainable planet.

This thesis focuses on analyzing and exploring solutions in which Information Technology, and especially parallel and distributed high performance systems, can improve the urgent needs of environmental community in managing this unprecedented amount of environmental data to provide meaningful information and knowledge in a timely manner. The required computational and storage capacity for such a challenge exceeds most of the time what an average computational center can offer. We propose in this thesis a general methodology and framework for easily porting and executing environmental applications simultaneously on different parallel and distributed infrastructures such as cluster, Grid and Cloud. Such a Hybrid Computing Environment introduces complex challenges to tackle with, especially in supporting the interoperability and the coexistence of the underlying distributed, heterogeneous computing infrastructures. The interoperability between Environmental Sciences (environmental data and environmental applications) and such a Hybrid Computing Environment is also an important goal to achieve in this thesis and the proposed approach is based on a mediation solution, through the introduction of an intermediate “broker” layer (mediator), able to hide the complexity of the computing environment and to provide access to its functionalities and capabilities in an easy and flexible manner.

Based on a heuristic approach, we have analyzed the lessons learned from integrating different types of environmental applications with different high performance parallel and distributed infrastructures and together with the accumulated theoretical knowledge we defined a conceptualization method for environmental applications, which allows an easy and flexible integration with any computing environment. The conceptual model is a key component in the general proposed methodology for porting and executing environmental applications on a Hybrid Computing Environment. Based on a conceptual model, the application is decomposed and scheduled to be executed on an appropriate set of computing resources, belonging to one or more distributed infrastructures. The proposed scheduler take into account factors like application type, task complexity, user preferences and specifications, platforms availability, execution history, etc. and the execution on different computing infrastructures is done using specialized adaptors.

The definition of such a methodology and supporting framework came as a response to the urgent need of the environmental community of using as much computational resources as possible to efficiently and effectively analyze and process the huge amounts of data that are available today, to be able to make better informed decisions regarding the changes that are threatening our environment and our society. The proposed approach is easy and flexible enough to be applied on a large range of scientific applications, belonging also to other research fields. We believe that such a methodology is a step forward into a standardize way of accessing large heterogeneous storage and computing facilities for complex scientific applications.

Abstract

Schimbările globale de mediu, legate de climă, biodiversitate, degradarea mediului, poluare dar și altele, amenință planeta noastră din ce în ce mai mult în fiecare zi, cu rezultate negative precum: creșterea nivelului global al mării, secete intensificate, topirea ghetarilor, etc. Comunitatea Științelor Mediului, și societatea umană în general, trebuie să găsească răspunsuri eficiente și operaționale pentru aceste schimbări complexe. Aceste acțiuni necesită mai mult ca oricând o mai bună înțelegere a sistemului complex numit Pământ, a interdependenței dintre sub-sistemele sale și a impactului activităților umane induse asupra fenomenelor naturale. O astfel de provocare necesită acces nu numai la o mare cantitate de date de mediu, care provin din diferite domenii și se află la diferite scări geografice, dar și la resurse de stocare și de procesare, precum și la unelte, algoritmi și servicii performante și standardizate, capabile de a extrage informații utile și semnificative din datele brute, informații care vor sprijini procesul de luare a unor decizii și acțiuni mai bune, care vor duce către o dezvoltare durabilă și o planetă mai bună.

Această teză se concentrează pe analiza și explorarea de soluții prin care tehnologia informației, și în special sistemele de calcul de înaltă performanță, paralele și distribuite, pot îmbunătăți nevoile urgente ale comunității de mediu legate de gestionarea acestei cantități fără precedent de date, pentru a furniza informații semnificative în timp util. Capacitatea de calcul și de stocare cerută pentru o astfel de provocare depășește de cele mai multe ori ceea ce un centru de calcul mediu poate oferi. În această teză propunem o metodologie și o platformă generală pentru portarea cu ușurință și execuția aplicațiilor de mediu, în mod simultan, pe diferite infrastructuri paralele și distribuite, cum ar fi cluster, Grid și Cloud. Un astfel de mediu hibrid de calcul introduce provocări complexe ce trebuie rezolvate, în special în sprijinirea interoperabilității și coexistenței infrastructurilor distribuite și eterogene care stau la baza acestui mediu hibrid. Interoperabilitatea dintre Științele Mediului (date de mediu și aplicații de mediu) și un astfel de mediu hibrid de calcul este un obiectiv important de realizat în această teză iar abordarea propusă se bazează pe o soluție de mediere, prin introducerea unui nivel intermediar "broker" (mediator), capabil să ascundă complexitatea mediului de calcul și să ofere acces la funcțiile și capacitățile sale într-un mod simplu și flexibil.

Pe baza unei abordări euristice, am analizat lecțiile învățate din integrarea diferitelor tipuri de aplicații de mediu pe diferite infrastructuri de calcul de înaltă performanță, paralele și distribuite, și împreună cu cunoștințele teoretice acumulate am definit un model conceptual pentru aplicații de mediu, ceea ce permite o integrare ușoară și flexibilă cu orice mediu de calcul. Modelul conceptual este o componentă cheie în metodologia generală propusă pentru portarea și execuția aplicațiilor de mediu pe un mediu hibrid de calcul. Pe baza modelului conceptual, aplicația este descompusă și planificată spre execuție pe un set adecvat de resurse de calcul, aparținând uneia sau mai multor infrastructuri distribuite. Planificatorul propus ia în considerare factori precum: tipul aplicației, complexitatea procesării, preferințele utilizatorului și specificațiile aplicației, disponibilitatea platformelor, istoricul de execuție, etc. iar execuția pe diferite infrastructuri de calcul se face folosind adaptoare specializate.

Definirea unui astfel de metodologii, și a platformei aferente, a venit ca un răspuns la nevoia urgentă a comunității de mediu de a utiliza cât mai multe resurse de calcul, pentru a analiza în mod eficient prelucrarea unor cantități uriașe de date, disponibile în prezent, și pentru a lua decizii informate mai bune, cu privire la schimbările care amenință mediul nostru și societatea noastră. Abordarea propusă este ușoră și suficient de flexibilă pentru a fi aplicată pe o gamă largă de aplicații științifice, care aparțin altor domenii de cercetare. Noi credem că o astfel de metodologie este un pas înainte într-un proces de standardizare a accesului la infrastructuri mari de stocare și de calcul, eterogene, pentru aplicații științifice complexe.

Résumé

Les changements environnementaux globaux liés notamment au climat, à la biodiversité, à la dégradation des habitats, et à la pollution mettent de plus en plus en danger notre écosystème planétaire avec des conséquences néfastes comme la hausse du niveau de la mer, une intensification des sécheresses, une fonte accélérée des glaciers, etc. La communauté des scientifiques œuvrant dans les sciences environnementales, et la société au sens large, ont un besoin accru de trouver des solutions efficaces et opérationnelles à ces changements complexes. Ces actions impliquent, plus que jamais, une meilleure compréhension du système de la planète Terre, des relations entre ses sous-systèmes, ainsi que des impacts induits par la société humaine sur les dynamiques naturelles de l'environnement. Un tel challenge requiert non seulement un accès à une immense quantité de données, provenant de diverses disciplines et à des échelles géographiques variées, mais également à des ressources massives pour le stockage et l'analyse de ces données, ainsi qu'à des outils standardisés performants, des algorithmes et des services capables d'extraire de l'information pertinente et utile à partir des données brutes. Ceci en vue de permettre un meilleur processus de prise de décision, et des actions ciblées et utiles pour satisfaire les buts liés au développement durable.

Ce travail de thèse se focalise sur l'analyse et l'exploration de solutions permettant aux technologies de l'informations, et particulièrement les systèmes de calcul à haute performance, d'apporter des réponses utiles aux besoins de la communauté des sciences de l'environnement dans le domaine de la provision d'information et de nouvelles connaissances. Beaucoup de ces besoins demandent une capacité d'analyse et de stockage des données qui excède ce qu'un unique centre de calcul de taille moyenne peut offrir. Nous proposons dans cette thèse une méthodologie générale pour facilement porter et exécuter des applications environnementales de manière simultanée sur diverses infrastructures de calcul parallèles et distribuées telles que des clusters, Grid et Cloud. Cet environnement de calcul hybride introduit des problèmes très complexes à résoudre, notamment ceux en lien avec l'interopérabilité et la coexistence de ces infrastructures hétérogènes distribuées. Facilitée l'interopérabilité entre les données et applications environnementales et cette infrastructure hybride a été un des buts majeurs de cette thèse, et la solution présentée se base sur une approche utilisant une couche intermédiaire de

médiation (*mediator*) capable de cacher la complexité de l'environnement computationnel tout en donnant accès à ses fonctionnalités de manière simple et flexible.

Nous basant sur une approche euristique, nous avons analysé les avantages et désavantages de l'intégration de différentes applications environnementales sur les infrastructures hautes performances parallèles et distribuées. Nous avons ensuite défini une méthode de conceptualisation pour les applications environnementales qui permet une intégration facilitée dans n'importe quel environnement computationnel. Le modèle conceptuel est un élément clé de la méthode générale proposée. Basée sur ce modèle, une application est décomposée et planifiée pour être exécutée dans un ensemble approprié de ressources computationnelles appartenant à une ou plusieurs infrastructures distribuées.

Le composant de planification (*planificateur*) prend en compte des facteurs tels que le type d'application, la complexité de la tâche à exécuter, les préférences de l'utilisateur, la disponibilité des plateformes, l'historiques d'exécution, etc. L'exécution sur différentes infrastructures computationnelles est alors faite en utilisant des adaptateurs spécialisés.

La définition et le développement d'une telle méthodologie vient en réponse aux besoins de la communauté environnementale à devoir utiliser une quantité croissante de ressources computationnelles, afin d'informer au mieux les processus décisionnels liés à l'environnement et la société. L'approche proposée est suffisamment flexible pour être appliquée à un large spectre d'applications environnementales, et même d'applications liées à d'autres thématiques. Nous croyons qu'une telle méthodologie est une avancée significative vers une façon standardisée d'accéder à un ensemble hétérogène de ressources de stockage et de calcul.

Acknowledgements

Coming to an end of this tremendous journey, I would like to express my special appreciations and to thank to a list of people who are of great importance to me.

First of all I would like to warmly thank and express my exceptional appreciations to my three advisors and directors of this thesis: Prof. Dorian Gorgan, Dr. Nicolas Ray and Prof. Anthony Lehman.

Prof. Dorian Gorgan is not only a tremendous mentor for me, who constantly encouraged my research and guided my path as a scientific researcher, but he is also a true source of inspiration through his pedagogy, scientific discussions, passion, constant enthusiasm and humanity. His presence in both my professional and personal development has been invaluable and the experience gathered under his supervision has enriched me as a scientist and as a person.

It is my pleasure to express my profound gratitude to my advisor **Dr. Nicolas Ray** for his invaluable support, supervision and encouragement during this research work. His continuous support and guidance, patience, passion and outstanding organization make him an excellent advisor, a great researcher and a nice human being. This thesis would have not been possible without his presence.

I am more than grateful to **Prof. Anthony Lehmann** who brought invaluable contributions towards the completion of this thesis. He is an excellent leader and coordinator, who can bring out the best of his students, a nice human being who is honest, and helpful to others. I thank him for giving me the chance and the privilege to work with him.

I would also like to thank the members of my doctoral evaluation committee, Prof. Nicolae Tăpuș, Prof. Gheorghe Sebestyen-Pal, Dr. Jean-Luc Falcone, and Prof. Liviu Miclea, for their efforts in reading the thesis, their brilliant comments and suggestions, but also for their support and inspiration.

I am very pleased to thank my dear colleges from the CGIS Laboratory: Victor Bâcu, Teodor Stefănuț, Adrian Sabou, Dănuț Mihon, Vlad Colceriu, Cornelia Melenti and Adrian Coleșa. I started my research path working with them as a team and besides having great colleagues I also gained special friends. We have worked together in many projects, we have published numerous scientific papers, we came up with innovative solutions, we debated on numerous topics, we celebrated birthdays and personal events together and for all that I have not enough words to thank them.

A warm and big thank to all my colleagues from UNEP/GRID-Geneva: Pascal Peduzzi, Gregory Giuliani, Bruno Chatenoux, Andrea De Bono, Jean-Phillippe Richard, Yaniss Guigoz, Frederic Mozer, Karin Allenbach, Mila Lomarda, Pierre Lacroix, Pierric de Laboire, Stefan Schwarzer, Hy Dao, and Julien Goy. Special thanks go to Bruno Chatenoux and Gregory Giuliani for their scientific support and personal discussions. It was a great pleasure working with all my colleagues, they are leading specialists and each one of them contributed in one-way or another to my enrich both as a professional and as a human being. I thank them for all the special moments spent together and for warmly receiving me in the GRID family.

I would also like to thank my colleagues from the Institute of Environmental Sciences who immediately received me in their group and always supported me during my research with not only scientific discussions but also with good humor and pleasant moments: Charles-Antoine Kuszli, Ana Silva, Kazi Rahman, Enrique Moran, Thierry Froidevaux, Jan Ries, Clara Rodriguez Morata, Martin Schlaepher, Martin Lacayo, Maria-Theresa Miranda, Francis Bergeron, Maura Brunetti, Marjorie Perroud, Martin Beniston.
Warm thanks go to Martin Lacayo, who kindly offered his support for proofreading and correcting the English in this thesis.

Special thanks to the secretaries of the Institute for Environmental Science (ISE) and also my friends Nomi de Brito and Laure Sollero who always helped me not only with the administrative part but also with their positive and friendly attitude.

I want to give my special thanks and thoughts to all my friends, for constantly supporting me, encouraging me, and most of all for being there for me during this entire journey. I always considered myself the luckiest person for having such wonderful people around me.

In the end, I would like to thank my family, my parents and my brother, who offered me more than I could ever ask for. Words cannot express how grateful I am to have them. Their love, their constant support in following my dreams, their patience and help in difficult moments, their encouragements and most of all their trust that I can do anything I want, motivated me beyond my limits!

I also thank God for guiding my steps throughout this journey and I have faith that He has even bigger plans for me in the future.

****

I would like to acknowledge the European Commission "seventh framework programme" that funded the enviroGRIDS (Grant Agreement number 226740) project and also the Scientific Exchange Programme between Switzerland and the New Member State (SCIEX) for funding the enviroPAD project (Grant number 12.206) and scholarship for 18 months at the University of Geneva.

Table of Content

List of Figures

Acronyms

95PPU	95% Prediction Uncertainty
ACI	Advanced Cyber Infrastructure
ALAP	As Late As Possible
AMDS	Annual Maxima Daily Streamflow
ANZLIC	Australia New Zealand Land Information Council
API	Application Programming Interface
ARS	Agricultural Research Service
ATLAS	A Toroidal LHC ApparatuS – One of the seven particle detector at CERN
CA	Certificate Authority
CCF	Cluster ready Children First
CE	Computing Element
CERN	Centre Européen pour La Recherche Nucléaire
CPIP	Cloud Portability and Interoperability Profiles
CPU	Central Processing Unit
CS	Catalog Service
CSML	Climate Science Modeling Language
CSV	Comma Separated Values
CWS	Catalog Web Service
DAG	Directed Acyclic Graph
DBMS	Database Management System
DCI	Distributed Computing Infrastructure
DEISA	Distributed European Infrastructure for Supercomputing Applications
DEM	Digital Elevation Model
DIANE	Distributed Analysis Environment
DGGS	Discrete Global Grid System
DHCP	Dynamic Host Configuration Protocol
DMC	Distributed Multiscale Computing
DMTF	Distributed Management Task Force
EC2	Amazon Elastic Compute Cloud
EDAP	Environmental Data Acquisition and Processing
EDE	Environmental Data Explorer
EDF	Earliest Deadline First
EGEE	Enabling Grid for E-SciencE
EGI	European Grid Infrastructure
EGII	European Geographic Information Infrastructure
e-IRG	e-Infrastructure Reflection Group
EMI	European Middleware Initiative
EML	Ecological Metadata Language
EO	Earth Observation
ESA	European Space Agency
ESFRI	European Strategy Forum on Research Infrastructure
ESGF	Earth System Grid Federation
ESML	Earth Science Markup Language

ETF	Earliest Time First
EUROGI	European Umbrella Organization for Geographic Information
FAO	Food and Agriculture Organization
FGDC	Federal Geographic Data Committee
FOSS4G	Free and Open Source Software for Geospatial
FP7	7th Framework Program for EU Research
FTP	File Transfer Protocol
GA	Genetic Algorithm
GAR	Global Assessment Report
GCI	GEOSS Common Infrastructure
GCM	Global Climate Model
GDAS	Geographic Data Attribute Set
GEE	Google Earth Engine
GEO	Group on Earth Observation
GEO DAB	GEO Discovery and Access Broker
GEOSS	Global Earth Observation System of Systems
GeoTIFF	Georeferenced Tagged Image File Format
GI	Geographic Information
GIN	Grid Interoperability Now
GIS	Geographic Information System
GIS	Grid Information Service
GML	Geographic Markup Language
GPU	Graphical Processing Unit
GRAM	Globus Resource Allocation Manager
GRID	Global Resource Information Database
GridFTP	Grid File Transfer Protocol
gSDI	Grid enabled Spatial Data Infrastructure
GSDI	Global Spatial Data Infrastructure
GSS	Generic Security Service (Grid)
GWS	Geo-WebService
HCE	Hybrid Computing Environment
HDDS	Hadoop Distributed File System
HDF	Hierarchical Data Format
HDF-EOS	Hierarchical Data Format – Earth Observing System
HLFET	Highest Level First with Estimated Times
HPC	High Performance Computing
HRU	Hydrological Response Unit
HTTP	HyperText Transfer Protocol
IaaS	Infrastructure as a Service
ICT	Information and Communications Technology
ICSU	International Council for Science
INSPIRE	Infrastructure for Spatial Information in the European Community
IoT	Internet of Things
IP	Internet Protocol
IPCC	International Panel on Climate Change
ISO	International Organization for Standardization

ISSC	International Social Science Council
IT	Information Technology
JDL	Job Description Language
JPEG	Joint Photographic Expert Group
KML	Keyhole Markup Language
KPB	K-Percent Best (Scheduling Algorithm)
LAN	Local Area Network
LCC	LHC Computing Grid
LFC	Logical File Catalog
LFN	Logical File Name
LH	Latin Hypercube
LHC	Large Hadron Collider
LHCb	Large Hadron Collider beauty – one of the seven particle detector at CERN
LIDAR	Light Detection And Ranging
LiMES	Live Monitoring of Earth Surface
M2M	Machine to Machine
MCP	Modified Critical Path
MCT	Minimum Completion Time
MDG	Millennium Development Goal
MET	Minimum Execution Time
MIMD	Multiple Instruction Multiple Data
MIPS	Million Instructions Per Second
MISD	Multiple Instruction Single Data
MnMn	Min-Min (Scheduling Algorithm)
MODIS	MODerate resolution Imaging Spectroradiometer
MonALISA	Monitoring Agents in A Large Integrated Services Architecture
MPI	Message Passing Interface
MPMD	Multiple Process Multiple Data
MPSD	Multiple Process Single Data
MxMn	Max-Min (Scheduling Algorithm)
NcML	NetCDF Markup Language
NCSA	National Center for Supercomputing Applications
NDVI	Normalized Difference Vegetation Index
NetCDF	Network Common Data Form
NIC	Network Interface Card
nIR	electromagnetic reflectance in Near Infra-Red
NSDI	National Spatial Data Infrastructure
NSF	National Science Foundation
OCC	Open Cloud Consortium
OCCI	Open Cloud Computing Interface
OGC	Open Geospatial Consortium
OGF	Open Grid Forum
OGSA	Open Grid Service Architecture
OGSA-DAI	OGSA Data Access and Integration
OLB	Opportunistic Load Balancing
OWS	OGC Web Service

P2P	Peer-to-peer
PaaS	Platform as a Service
PKI	Public Key Infrastructure
PTG	Parallel Task Graph
QoS	Quality of Service
RDF	Resource Description Framework
RDBMS	Relational Database Management System
RDI2	Rutgers Discovery Informatics Institute
REST	Representational State Transfer
RS	Remote Sensing
S3	Amazon Simple Storage Service
SA	Simulated Annealing (Scheduling Algorithm)
SaaS	Software as a Service
SDI	Spatial Data Infrastructure
SDN	Software-defined networkig
SE	Storage Element
SHIWA	SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs
SIMD	Single Instruction Multiple Data
SJF	Shortest Job First
SJN	Shortest Job Next
SLA	Service Level Agreement
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SPMD	Single Process Multiple Data
SQL	Structured Query Language
SUFI-2	Sequential Uncertainty Fitting Version 2
SwA	Switching Algorithm
SWAT	Soil and Water Assessment Tool
TBB	Threading Building Blocks
TIFF	Tagged Image File Format
TJS	Table Joining Service
TTC	Time to Completion
UML	Unified Modeling Language
UNEP	United Nations Environmental Program
URL	Uniform Resource Locator
USDA	U.S. Department of Agriculture
UNESCO	United Nations Educational, Scientific and Cultural Organization
UNU	United Nations University
VLAN	Virtual Local Area Network
VM	Virtual Machine
VMI	Virtual Machine Image
VMRC	Virtual Machine Image Repository and Catalog
VO	Virtual Organization
VOMS	Virtual Organization Membership Service
W3C	World Wide Web Consortium

WATERS	Watershed Assessment, Tracking and Environmental Results System
WB	World Bank
WCS	Web Coverage Service
WFS	Web Feature Service
WMO	World Meteorological Organization
WMS	Web Mapping Service
WMS	Workload Management System
WN	Worker Node
WPS	Web Processing Service
WSDL	Web Service Description Language
WSFR	Web Service Resource Framework
XML	eXtended Markup Language

Introduction

Chapter 1: Research Context and Questions

1.1. General Presentation of the Research Theme

At the beginning of the 21st century, global changes linked to climate, biodiversity and habitat loss, environmental degradation and pollution, are threatening our natural environment and the human society at large, with already tangible negative outcomes (IPCC Climate Change 2014 Synthesis Report [112]). Intensified droughts, ocean acidification, global sea level rise, increases in frequency of extreme weather events and glaciers melting are examples of such outcomes that are thought to intensify if appropriate international policies are not endorsed and applied.

Responding effectively to all these complex changes has become an important challenge for policy makers, but also for the scientific community that demands access to continuously increasing quantities of heterogeneous data and resources (ESFRI e-IRG Report on Data management, 2009 [66]). The need to understand the inter-linkage between natural phenomena and human-induced activities is urgent and an important aspect for achieving this is the accessibility and processing of environmental data from various disciplines and geographic scales (local, regional, national and global).

Scientists started to look and feel the need of an "innovative solution" which will allow them to access an operational infrastructure, supporting large scale multidisciplinary applications while providing highly elastic resources. The computational and storage capacity required for such a challenge exceeds most of the time what an average computational center can offer.

In this thesis we propose a general methodology and framework for easily porting and executing environmental applications simultaneously on different parallel and distributed infrastructures such as cluster, Grid and Cloud. To achieve this goal, we focus on solving two main problems: 1) the interoperability and the coexistence of different distributed, heterogeneous computing infrastructures within a Hybrid Computing Environment, and 2) the interoperability between Environmental Sciences (environmental data and environmental applications) and such a Hybrid Computing Environment with its underlying infrastructures.

The methodology proposed in this research will allow users to interact in a central and transparent manner with several parallel and distributed infrastructures to try to solve the most challenging issues that are threatening our environment.

1.2. Challenges and Research Questions

The advances in technology and the extraordinary growth of digital data sources (including geographically distributed sensors, mobile devices, satellites and instrumented infrastructures) led to an unprecedented amount of data coming in different formats, from different sources and at different time intervals. Due to this data deluge and to the heterogeneity and distribution of computing and storing resources, we have to rethink the ways to store, process and analyze the data to provide meaningful information and knowledge in a timely manner (Diaz-Montes et al., 2015 [55]).

Turning data into knowledge is not an easy task, especially when locating and accessing the right resources (e.g. data, information, tools and services, which can be information about the state of the Earth, relevant services, project results, applications, etc.) is done in a very scattered way through different state organizations, operators, service companies, data catalogues, scientific institutes, etc.

The complexity of many sciences and engineering problems and applications, which can potentially transform our ability to understand the Earth System and manage our lives and our environment, requires computational and storage capacity exceeding in most of the cases what an average single computational center can offer (Diaz-Montes et al., 2015 [55]). Although many of these applications can be divided into sets of independent jobs, their collective complexity still requires "millions of core hours on any state-of the-art supercomputer, and throughput that a single multi-user queuing system cannot sustain" (Diaz-Montes et al., 2015 [55]). A general framework for simultaneously running scientific applications on different parallel and distributed environments such as Multicore, cluster, Grid, Cloud, etc. could bring huge benefits. The simultaneous execution on multiple environments is a complex and challenging task as it involves not only the interoperability of scientific applications with different parallel and distributed environments but also the interoperability and the coexistence of these environments.

To this extent, the main goal of this research is: **To analyze and explore in which way Information Technology, and especially parallel and distributed high-performance**

systems, can improve the major challenges and needs the Environmental Sciences are facing in the process of extracting understandable and useful information from raw environmental data, leading in the end to a better informed decision making towards a sustainable development and a sustainable planet.

To be able to achieve this goal, we have formulated some associated research questions that we will try to answer in the chapters of this thesis:

1. **What are the current urgent *needs and challenges* of Environmental Sciences (focusing on environmental data and environmental applications)?**
2. **What is the *state of the art* of High Performance Computing (HPC) landscape (cluster, Grid, Cloud, Hybrid Computing)?**
3. **What are the *lessons learned* based on a heuristic approach of integrating environmental applications with HPC?**
4. **What are the flexible solutions to *integrate* environmental applications and environmental data with HPC infrastructures?**
5. **How can we solve the *interoperability* between Environmental Sciences and Computer Science?**
6. **Is there an efficiently way to take advantage of all the available *heterogeneous computing* infrastructures *simultaneously*?**
7. **How can we solve the *interoperability between different HPC infrastructures*? What are the challenges and how this solution can be *applied* to Environmental Sciences?**
8. **How can we *evaluate and validate* the proposed solutions?**

1.3. Objectives

Based on the research questions formulated in the previous subsection, we have established the following main objectives for this thesis:

- Analyze the main issues and problems needed to be solved in the Environmental Sciences field: gather, analyze, process, and distribute environmental data using dedicated services and applications.
- Analyze and study the new trends in parallel and distributed computing infrastructures. The infrastructures considered in this research are Grid, Cloud, cluster, and Hybrid

Computing. These platforms were chosen due to their capabilities to deliver the necessary computing and storage resources needed for executing scientific applications in general. They can be very heterogeneous and hierarchical, making thus the execution even more challenging and the programming more complex. All these platforms aim to increase the execution speed of large-scale applications, offering a high number of computing and storage resources. Depending on the applications, some of them perform better than others do, while some offer a better scalability or more flexibility. They have many similarities but they present also many differences, which make the simultaneous usage of these platforms a complex and challenging task.

- Experiment and extract common standards and properties of different kind of environmental services and applications: desktop applications, legacy code, Web applications, etc. At this point, different use cases will be chosen as a starting point for analyzing and confirming the need for a hybrid-computing environment. The aim is to experiment and to find solutions for optimal mapping of the execution of environmental applications onto parallel and distributed environments but also to explore and highlight the elements by which such mapping solutions converge toward an optimum. Based on these experiments and use cases, a minimum set of requirements will be collected and a list of lessons will be analyzed for the development of a new methodology.
- Define a conceptual model for environmental applications, which will allow an easy and flexible integration with any computing environment.
- Analyze the conditions in which a specific parallel and distributed environment is efficient and optimum for a certain application.
- Explore the complexity of a hybrid computing environment and the major challenges.
- Propose and explore a new methodology and a framework to solve the interoperability between environmental applications and a Hybrid Computing Environment. The mapping of environmental applications to an optimum and efficient computing environment should take into account several criteria such as applications features, user specifications, processing requirements, environment and resource availability, execution history, etc. Environmental applications usually require large computational and storage resources, which cannot always be delivered by a single platform. There are also cases in which the execution can be made on a smaller set of resources, even a local server, depending on

the application requests or even on the user specifications. Choosing the optimum execution environment, or the optimum combination of heterogeneous computing resources, becomes an important issue for obtaining better performances.

- Propose a set of metrics able to evaluate the efficiency of the proposed methodology and framework. Draw conclusions and set up future research directions.

1.4. Structure of the Thesis

This thesis is structured in five big sections, which are further sub-structured in twelve chapters. This structure is illustrated in Figure 1 and presented in detail in the following.

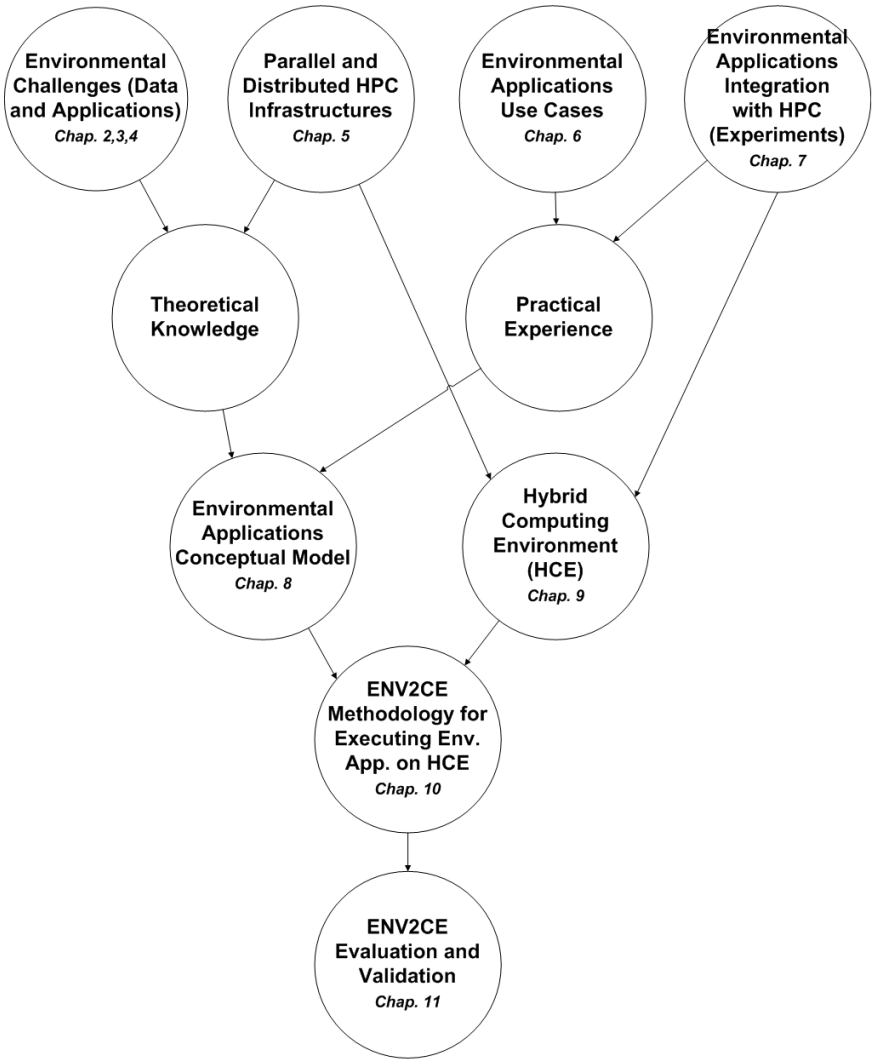

Figure 1: Thesis Structure

- **Introduction –** is the introductory part, which presents the big picture (general context) of the thesis, explaining the need for such a research work and the interdisciplinary nature of the thesis. This section contains one chapter:
    - **Chapter 1** – presents the general interdisciplinary research theme, what are the major challenges and research questions we address during the thesis and what are our main objectives for this work.
- **State of the Art –** gives a recent overview of the themes and fields approached in this thesis, both in Environmental Sciences and Computer Science research areas. This section contains four chapters:
    - **Chapter 2** – focuses on the existing data challenges present in Environmental Sciences and introduces environmental data together with its main attributes, properties and transformations.
    - **Chapter 3** – gives an overview on the basic concepts and notions underlying the Spatial Data Infrastructures, focusing on standards, interoperability, tools and initiatives used to support this idea.
    - **Chapter 4** – introduces environmental applications, their specificity and characteristics and what are the challenges related to them. This chapter focuses on two main categories of environmental applications: hydrological modeling and remote sensing applications.
    - **Chapter 5** – presents a general updated view of distributed systems (cluster, Grid, Cloud, etc.), focusing on the properties and tools that have been used more intensively in our research. We also introduce the concept of Hybrid Computing and we generally present the properties of distributed applications.
- **Development and Execution of Environmental Applications –** describes in detail the main environmental applications used and developed during our research. This section is composed out of two chapters:
    - **Chapter 6** – presents a set of environmental applications that were developed and/or used during our research in different national and international projects.
    - **Chapter 7** – gives an overview of the growing needs and requests identified in Environmental Sciences to take advantage of the capabilities offered by the parallel and distributed infrastructures. This chapter focuses on a list of

experiments addressed in our research for porting different environmental applications on different parallel and distributed infrastructures. We analyzed the results, drew the conclusions and came up with a list of lessons learned based on the performed experiments. These lessons learned are the foundation of the next chapters, as we based our next proposals on our gained experience so far.

- **New Methodology and Framework Proposal –** introduces a new methodology (ENV2CE) and a new framework for efficiently porting and executing environmental applications on a Hybrid Computing Environment (HCE). This section contains four chapters:
  - **Chapter 8** – presents a conceptual description of environmental applications by introducing a proposal for an application conceptual model together with the methodology to apply this model.
  - **Chapter 9** – gives an overview of what a HCE is, what are the properties, advantages, disadvantages and most of all the challenges of working with such an environment.
  - **Chapter 10** – focuses on the new methodology and the new framework proposed for efficiently executing environmental applications on a HCE. This chapter presents the methodology phases, the system architecture and describes the main components together with their main functionalities.
  - **Chapter 11** – presents the evaluation and the validation of the proposed solutions.
- **Conclusions –** presents the general conclusions and contains one chapter:
  - **Chapter 12** – concludes this research by answering the research questions addressed in the introduction, by emphasizing our contributions and by presenting recommendations for future directions of work.

The chapters presented in this thesis are derived from scientific papers published during this research:

- Chapter 2 is based on:
  - ***Rodila, D.,*** *Ray, N., Gorgan, D. (2015), Conceptual Model for Environmental Science Applications on Parallel and Distributed Infrastructures, Environmental System Research, Vol. 4/23, 2015, http://dx.doi.org/10.1186/s40068-015-0050-1.*

- Chapter 3 is based on:
  - ***Rodila, D.,*** *and Gorgan, D. (2012). Geospatial and Grid Interoperability through OGC Services Gridification, in International Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS), Vol. 5/6, December 2012, pp. 1650 – 1658, ISSN: 1939-1404, http://dx.doi.org/10.1109/JSTARS.2012.2217115.*
  - *Mihon, D., Colceriu, V., Bacu, V., Allenbach, K.,* ***Rodila, D.,*** *Giuliani, G., and Gorgan, D. (2013). OGC Compliant Services for Remote Sensing Processing over the Grid Infrastructure, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 32–40, ISSN 2158-107X.*
  - *Gorgan, D., Bacu, V., Mihon, D., Stefanut, T.,* ***Rodila, D.,*** *Cau, P., Abbaspour, K., Giuliani, G., Ray, N., and Lehmann, A. (2012). Software platform interoperability throughout enviroGRIDS portal, in International Journal of Selected Topics in Applied Earth Observations and Remote Sensing - JSTARS, Vol. PP/99, pp. 1-11.*
  - ***Rodila, D.,*** *and Gorgan, D. (2011). A Mediation Approach in Geospatial Web Services Gridification, in ICCP2011 – IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 25-27, 2011, pp. 541-548, http://dx.doi.org/10.1109/ICCP.2011.6047928.*
  - ***Rodila, D.,*** *Bacu, V., Ardelean, V., Borlea, C., and Gorgan, D. (2011). Geospatial Web Services Gridification in enviroGRIDS, in European Geosciences Union - General Assembly EGU 2011, Vienna, Austria, April 03-08, 2011, (abstract and presentation), http://meetingorganizer.copernicus.org/EGU2011/EGU2011-11469.pdf.*
  - *Gorgan, D., Bacu, V., Manca, S., Giuliani, G.,* ***Rodila, D.,*** *Stefanut, T, Mihon, D., Abbaspour, K., Rouholahnejad, E., van Griensven, A., Kokoszkiewicz, L., and Cau, P. (2011). Tools and Applications in enviroGRIDS Project for Spatial Data Processing of Black Sea Catchment Basin, EGI User Forum, Vilnius, 11-15 April, https://www.egi.eu/indico/contributionDisplay.py?contribId=154&sessionId=16&confId=207.*
  - ***Rodila, D.,*** *and Gorgan, D. (2010). Integration of Spatial Data Infrastructures with Grid Environment, in SYNASC 2010 – 12th International Symposium on*

Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, September 23-26, 2010, pp. 269-277, http://dx.doi.org/10.1109/SYNASC.2010.63.

- ***Rodila, D.,** Gorgan, D., and Bacu, V. (2010). The Interoperability between OGC Services and Grid Environment in EnviroGRIDS Project, in 3PGCIC 2010 – International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Fukuoka, Japan, November 4-6, 2010, IEEE Computer Press, ISBN: 978-0-7695-4237-9, pp. 387-392, http://dx.doi.org/10.1109/3PGCIC.2010.65.*
- *Gorgan, D., **Rodila, D.,** Bacu, V., Giuliani, G., and Ray, N. (2010). OGC and Grid Interoperability in enviroGRIDS Project, in European Geosciences Union - General Assembly EGU 2010, May 02-07, 2010, Vienna, Austria (abstract and presentation), http://meetingorganizer.copernicus.org/EGU2010/EGU2010-13457.pdf.*
- *Gorgan, D., **Rodila, D.,** Bacu, V., Giuliani, G., Ray, N., Charvat, K., and Lehman, A. (2010). Geospatial and Grid infrastructures interoperability in enviroGRIDS, 5th EGEE User Forum, April 12-16, 2010, Uppsala, Sweden (2010). Book of Abstracts, http://indico.cern.ch/contributionDisplay.py?contribId=161&confId=69338.*

- Chapter 6 is based on:
  - *Giuliani, G., Dao, H., De Bono, A., Chatenoux, B., Allenbach, K., De Laborie, P., **Rodila, D.,** Alexandris, N., and Peduzzi, P. (2016). Live Monitoring of Earth Surface (LiMES): a framework for monitoring environmental changes from Earth Observations, (to be submitted).*
  - *Silvestro, F., Campo, L., Rudari, R., De Angeli, S., D'Andrea, M., **Rodila, D.,** and Gabellani, S. (2016). Impacts of EC-Earth Global Climate Model RCP4.5 climate change scenario on maximum daily streamflow quantiles at global scale, Journal of Climate (submitted article).*
  - ***Rodila, D.,** Bacu, V., and Gorgan, D. (2012). Comparative Parallel Execution of SWAT Hydrological Model on Multicore and Grid Architectures, in International Journal of Web and Grid Services (IJWGS), Vol. 8/3, September 2012, pp.304 – 320, http://dx.doi.org/10.1504/IJWGS.2012.049172.*

- *Bacu, V., Mihon, D., Stefanut, T., **Rodila, D.,** Abbaspour, K., Rouholahnejad, E., and Gorgan, D. (2013). Calibration of SWAT Hydrological Models in a Distributed Environment Using the gSWAT Application, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 66–74, ISSN 2158-107X.*
- *Gorgan, D., Bacu, V., Mihon, D., **Rodila, D.,** Abbaspour, K., and Rouholahnejad, E. (2012). Grid based calibration of SWAT hydrological models, in Journal of Nat. Hazards Earth Syst. Sci., Vol. 12/7, pp. 2411-2423, http://dx.doi.org/10.5194/nhess-12-2411-2012.*
- *Mihon, D., Bacu, V., **Rodila, D.,** Stefanut, T., Abbaspour, K., Rouholahnejad, E., and Gorgan, D. (2012). Grid Based Hydrologic Model Calibration and Execution, Chapter in the book: Advanced in Intelligent Control Systems and Computer Science, Dumitrache I. (Ed.), Springer-Verlag, Vol. 187, pp 279-293, ISBN 978-3-642-32548-9, http://dx.doi.org/10.1007/978-3-642-32548-9_20.*
- ***Rodila, D.,** Bacu, V., and Gorgan, D. (2012). Geospatial Applications on Different Parallel and Distributed Systems in enviroGRIDS Project, in European Geosciences Union - General Assembly EGU 2012, Vienna, Austria, April 22-27, 2012, (abstract).*
- ***Rodila, D.,** Bacu, V., and Gorgan, D. (2011). Comparative Analysis of Distributed and Grid Based Execution of SWAT Model, in 3PGCIC 2011 - Sixth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Barcelona, Spain, October 26-28, 2011, pp. 273-278, http://dx.doi.org/10.1109/3PGCIC.2011.49.*
- *Bacu, V., Mihon, D., **Rodila, D.,** Stefanut, T., and Gorgan, D. (2011). Grid Based Architectural Components for SWAT Model Calibration, in HPCS 2011 - International Conference on High Performance Computing and Simulation, Istanbul, Turkey, July 4-8, pp. 193-198, ISBN 978-1-61284-381-0.*

- Chapter 7 is based on:
  - ***Rodila, D.,** and Gorgan, D. (2010). Integration of Spatial Data Infrastructures with Grid Environment, 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp. 269-277.*

- ***Rodila, D.,*** *Gorgan, D., and Bacu, V. (2010). The Interoperability between OGC Services and Grid Environment in enviroGRIDS project. MiDiS-2010, The First International Workshop on Middleware for Large Scale Distributed Systems, November 4-6, 2010, Fukuoka, Japan, pp. 387-392.*
- ***Rodila, D.,*** *and Gorgan, D. (2011). A Mediation Approach in Geospatial Web Services Gridification, 12th IEEE International Conference on Intelligent Computer Communication and Processing, ICCP2011, pp. 541-548.*
- ***Rodila, D.,*** *and Gorgan, D. (2012). Geospatial and Grid Interoperability through OGC Services Gridification, in International Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS), Vol. 5/6, December 2012, pp. 1650 – 1658, ISSN: 1939-1404, http://dx.doi.org/10.1109/JSTARS.2012.2217115.*
- ***Rodila, D.,*** *Bacu, V., and Gorgan, D. (2012). Comparative Parallel Execution of SWAT Hydrological Model on Multicore and Grid Architectures, in International Journal of Web and Grid Services (IJWGS), Vol. 8/3, September 2012, pp.304 – 320, http://dx.doi.org/10.1504/IJWGS.2012.049172.*
- ***Rodila, D.,*** *Chatenoux, B., and Giuliani, G. (2016). Landsat 8 resources - a short comparison of different data access providers and methods (submitted paper).*

- Chapter 8 is based on:
  - ***Rodila, D.,*** *Ray, N., Gorgan, D. (2015), Conceptual Model for Environmental Science Applications on Parallel and Distributed Infrastructures, Environmental System Research, Vol. 4/23, 2015, http://dx.doi.org/10.1186/s40068-015-0050-1.*
  - *Gorgan, D., Mihon, D., Bacu, V., **Rodila, D.,** Stefanut, T., Colceriu, V., Allenbach, K., Balcik, F., Giuliani, G., Ray, N., and Lehmann, A. (2013). Flexible Description of Earth Data Processing over HPC Architectures, Big Data From Space Symposium, Frascati, Italy, 5-7 June, Abstract Book, pp. 39.*
  - *Colceriu, V., Mihon, D., Minculescu, A., Bacu, V., **Rodila, D.,** and Gorgan, D. (2013). Workflow Based Description and Distributed Processing of Satellite Images, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 50–57, ISSN 2158-107X.*

- *Bacu, V., Mihon, D., **Rodila, D.,** Stefanut, T., and Gorgan, D. (2011). gSWAT Platform for Grid based Hydrological Model Calibration and Execution, in ISPDC 2011 - 10th International Symposium on Parallel and Distributed Computing, Cluj-Napoca, Romania, July 6-8, 2011, pp.288-291.*
- *Gorgan, D., Stefanut, T., Bacu, V., Mihon, D., and **Rodila, D.** (2010). Grid based Environment Application Development Methodology, in Large-Scale Scientific Computing, Springer Journal LNCS 5910, pp.499-506, ISBN 978-3-642-12534-8, http://dx.doi.org/10.1007/978-3-642-12535-5_59.*

- Chapter 9 is based on:
  - ***Rodila, D.,** Gorgan, D., Ray, N., and Lehmann, A. (2016). ENV2CE: Environmental Application Conceptualization and Execution on a Hybrid Computing Environment -Framework and Methodology Proposal, (to be submitted).*
- Chapter 10 is based on:
  - ***Rodila, D.,** Ray, N., and Gorgan, D. (2015). Conceptual Model for Environmental Science Applications on Parallel and Distributed Infrastructures, Environmental System Research, Vol. 4/23, 2015, http://dx.doi.org/10.1186/s40068-015-0050-1.*
  - ***Rodila, D.,** Gorgan, D., Ray, N., and Lehmann, A. (2016). ENV2CE: Environmental Application Conceptualization and Execution on a Hybrid Computing Environment -Framework and Methodology Proposal, (to be submitted).*
  - ***Rodila, D.,** and Gorgan, D. (2012). Mapping Geospatial Applications onto Parallel and Distributed Environments, in ePaMuS 2012 – 5th International Workshop on Engineering Parallel and Multi-Core Systems – The Multi-Core Workshop, Palermo, Italy, 4-6 July, 2012, pp. 443 – 448, http://dx.doi.org/10.1109/CISIS.2012.152.*
  - ***Rodila, D.,** Bacu, V., and Gorgan, D. (2012). Geospatial Applications on Different Parallel and Distributed Systems in enviroGRIDS Project, in European Geosciences Union - General Assembly EGU 2012, Vienna, Austria, April 22-27, 2012, (abstract).*

- ***Rodila, D.,*** *and Gorgan, D. (2011). A Mediation Approach in Geospatial Web Services Gridification, in ICCP2011 – IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 25-27, 2011, pp. 541-548, http://dx.doi.org/10.1109/ICCP.2011.6047928.*
- *Gorgan, D., Stefanut, T., Bacu, V., Mihon, D., and* ***Rodila, D.*** *(2010). Grid based Environment Application Development Methodology, in Large-Scale Scientific Computing, Springer Journal LNCS 5910, pp.499-506, ISBN 978-3-642-12534-8, http://dx.doi.org/10.1007/978-3-642-12535-5_59.*

State of the Art

Chapter 2: Data Challenges in Environmental Sciences

2.1. Environmental Data

2.1.1. Introduction

In the domain of Earth and Environmental Science there is an unprecedented avalanche of data due to a large extent to the fast evolution and availability of sensor/detector technologies. The advances in IT that enabled the capture, analysis and storage of massive amounts of data contributed also to this avalanche of data. The so-called "digital data deluge" is a phenomena caused not only by the ease with which these large quantities of new data can be created but also by the output of re-analysis of already existing archived data (ESRI e-IRG Report on Data management, 2009 [66]). This phenomenon is considerably changing the way science and research is being conducted in many disciplines as they are dealing with unprecedented sizes of data that need massive computing capacities to handle it.

Environmental data is spatial data, which means data describing features and phenomena and their locations relative to Earth coordinate reference system (McKee, 2015 [149]). This implies that environmental data are most of the time spatially referenced (i.e., referring to a geographic location) and as such belongs to geospatial data or geodata. Geospatial data describes geographical locations by giving attributes/information about their spatial and/or temporal extents (Giuliani et al., 2011 [84]). The amount of geospatial data has grown dramatically in the last 30 years mostly due to the rapid progress of communication means, as well as technologies to capture this type of data (e.g., GPS, sensors, satellites). Geospatial data is typically voluminous, complex, heterogeneous and geographically distributed. All these attributes make it generally difficult to access, share and distribute geospatial data, often with challenges to combine it with other types of data sets. Nowadays geospatial data is used and analyzed most of the times within a Geographical Information System (GIS) that has capabilities such as assembling, storing, manipulating, displaying, and merging data from different sources (Giuliani et al., 2011 [84]). In environmental sciences, GIS can be used in conjunction with Spatial Data Infrastructures (SDIs), that are widely used to share, discover, retrieve and visualize geospatial

data through standardized services (e.g., Open Geospatial Consortium (OGC) services). SDIs are therefore more than just data repositories, although suffering from limited analytical capabilities. Making use of GIS and SDI, a wealth of geospatial applications, technologies and initiatives have emerged recently in order to handle the increasing amount of environmental data, and to extract useful information out of it. A more detail description of all these technologies (GIS, SDI, OGC, etc.) will be given in the following sections.

2.1.2. Formats and Standards

Geospatial data is usually found in three forms: raster data (such as geo-images), feature data – usually represented as vector data (such as road networks and nation boundaries), and 3D geometry (Figure 2). Depending on the data capture technologies, the resolution of these data are usually quite high and are constantly growing as technologies improves (Mahdavi-Amiri et al., 2015 [144]).

Raster data includes geospatial imagery data sets, which are typically categorized in aerial or satellite photographs and that are very useful for visualization and analysis of locations (Mahdavi-Amiri et al., 2015 [144]). Aerial photographs are taken from aircraft (such as helicopters and balloons) that do not have a fix support on the Earth. Satellite photographs on the other hand are taken by satellites and they usually have a lower resolution and quality compared to aerial images but the later are subject to air traffic and other restrictions while satellites are operational all time (except for maintenance periods) and can even revisit frequently locations for time-laps captures (Mahdavi-Amiri et al., 2015 [144]). Recently, huge amounts of raster data are also collected by military, using drones (Nedelcu, 2015 [163]).

Other forms of raster data sets (grid data) are these that describe terrain on the surface of the Earth. These can come in two formats:

- Digital Elevation Models (DEMs) – also known as Digital Terrain Models (DTMs) or height maps, are regular grids of terrain elevation values.
- Triangulated Irregular Networks (TINs) – sets of triangular faces with arbitrary connectivity, having the elevation values stored in the triangle vertices as coordinates.

Acquisition of elevation data can be done using techniques such as ground surveying, which are expensive and time consuming but very accurate, but also laser scanning, using

LIDAR (Light Detection and Ranging) technology on low altitude aircraft. LIDAR is a remote sensing technique that uses laser pulses to determine the elevation with high accuracy, usually from an aerial survey (Nedelcu, 2015 [163]). Photogrammetry (deriving measurements from imagery data) is another option of determining elevation data (Mahdavi-Amiri et al., 2015 [144]).

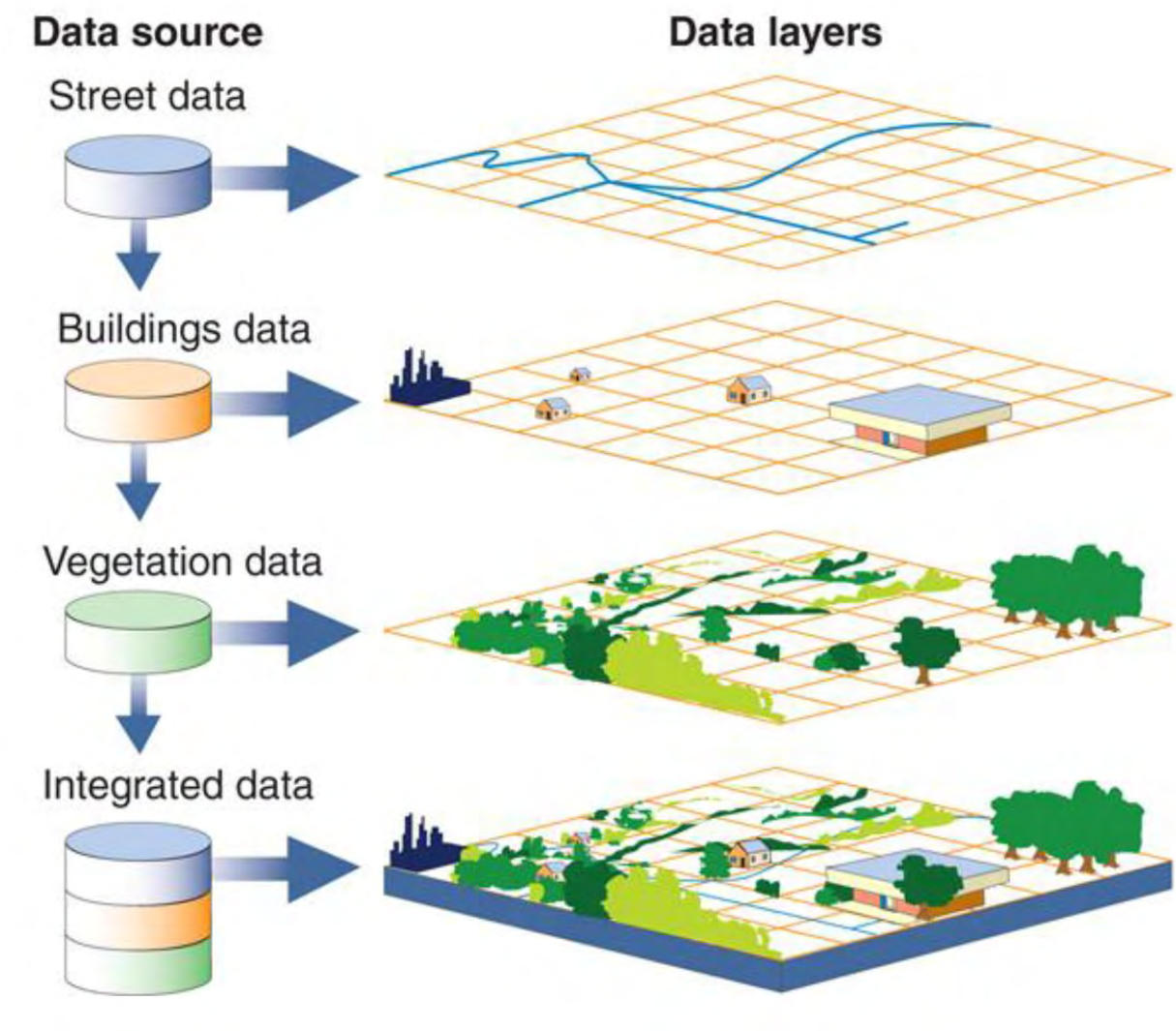

Figure 2: Geospatial Data Types (Source: Nedelcu, 2015)

Vector data sets (points, polylines, or polygons on the Earth) are composed of points connected by spherical ellipsoidal arcs and can represent region boundaries (such as nation or city boundaries) or geospatial networks and features (such as road and river networks or

residential house outlines) (Mahdavi-Amiri et al., 2015 [144]). Using rasterization, a vector can be represented as an ordered set of cells, where each cell corresponds to and contains one of the vector's vertices/points. Generation of vector data can be accomplished with ground surveying, LIDAR and photogrammetry.

3D geometric data sets model objects such as houses, office towers, bridges, cars or trees. These 3D models can be created using interactive modeling techniques or automatic methods, which can generate models based on large sets of images or LIDAR point cloud data (Otepka et al., 2013 [171]). An example of framework, which incorporates 3D models, is Google Earth.

In Environmental Sciences there is data that is considered "independent" and data considered "dependent". The "independent" variables are the ones being manipulated and selected to determine its relationship to an observed phenomenon. These are normally the input variables that are observed in its naturally occurring variation. The "dependent" variables are the observed results of the independent variables and are usually the output variables that cannot be directly controlled. The distinction of dependent and independent data is done by the researcher and by the context in which it is applied. Now considering the form of the response (dependent) environmental data, we can specify several types of data: continuous data (such as temperature, mass, distance), counts (simple – the number of plants infected by a disease, or categorical – the number of infected plants classified into tree species and town), proportions (such as: percent mortality, sex ratio), binary data (ex: alive or dead, present or absent), time to death/failure (ex. the time it takes juveniles to disperse out of the study area), time series (such as temperature data measured at fixed intervals, river discharged measured over time) and circular (ex: day of the year). A detailed description of all these types is done by Piegorsh and Bailer (2005) [176], and in (Analysis of Environmental Data [10]).

There are also many *de-facto* standards for delivering environmental data such as:

- **GeoTIFF** ([82]) – a TIFF (Tagged Image File Format) based interchange format for georeferenced raster imagery. 160 different remote sensing, Geographic Information System (GIS), cartographic, and surveying related companies and organizations have contributed to establish this format. This format supports a variety of compressions and tiling options, supports many numeric data representations, it is widely supported by GIS and it is the standard format for OGC Web Coverage Services output. As a negative side, this standard is voluminous and relative new.

- **HDF** (Hierarchical Data Format [105]) – technology and data format addressing problems on how to organize, store, discover, access, analyze, share, and preserve data in the face of enormous growth in size and complexity. It was developed to facilitate access to scientific data by the National Center for Supercomputing Applications (NCSA), and it's used in many fields, including Environmental Sciences. **HDF-EOS** (Hierarchical Data Format – Earth Observing System [106]) – the standard data format for all NASA Earth Observing System data products. Basically HDF-EOS datasets are HDF objects (images, tables, text, data arrays, etc.) with the added feature of being able to support geolocation information. There are two HDF formats and both **HDF4** (or just HDF) and **HDF5** have standard software interfaces available, they are widely used in modeling and remote sensing communities and the data model is general and flexible. The main drawback is that they require installation and use of the HDF4 and HDF5 libraries respectively. HDF and HDF5 are completely different and not compatible.
- **NetCDF** (Network Common Data Form [164]) – set of software libraries and self-describing, machine-independent data formats that support the creation, access, and sharing of array-oriented scientific data. This format also has a standard software interface and a wide range of tools available for manipulation and visualization. The data model is also general and flexible and it's used extensively in the atmospheric science research community. It also requires the installation of netCDF libraries.
- **NetCDF-4** (Network Common Data Form, version 4) – aim to combine the best of netCDF and HDF5 format with backward compatibility to netCDF. It allows new features such as multiple unlimited dimensions, groups and zlib compression.
- **GRIB** (Gridded Binary) – designed to exchange gridded data generated by numerical weather prediction models. It is efficient for transmitting and archiving large volumes of two-dimensional meteorological and oceanographic data and widely used for storage and exchange of gridded data within meteorological communities. The description of data is encoded in tables so the use of GRIB data has to always be done with the related tables.
- **XML** with initiatives such as
    - **GML** (Geography Markup Language) (OCG – GML [94]) – used for representing and storing the structure and content of geographic features. It's an OGC (Open Geospatial Consortium) standard developed for addressing geographic data

interoperability among GIS applications. It can be used to represent spatial and non-spatial aspects of geographic features and it's the only language formally recognized as a standard for geospatial data exchange. It can capture feature characteristics such as: topology, routing, units, and measurements and can model feature attributes and geometries. The GML is mostly used in the WFS as a way to send geographical features between servers and clients. This language can be seen as a grammar, able to define concrete features through GML Application Schemas, as the GML core schema does not contain definition of features. The GML is mostly used and generated by software answering a specific request and then receiving the result as a GML dataset. The information stored in a GML file can be easily shared with other information as it is based on available XML technologies and for developing a specific data model, specialized/specific applications could reuse, extend and/or refine GML component in an application schema. The Keyhole Markup Language (KML) has as main focus the data visualization. The difference between KML and GML is that the latest is not used only for data visualization but serves also as a modeling language as well as an open and interoperable exchange format over the Internet (Giuliani, 2011b [85]).

- **CSML** (Climate Science Modeling Language) – GML application schema attempting to encapsulate important semantics of climate science data.
- **ESML** (Earth Science Markup Language) – XML and API aiming to improve interoperability by defining a formal mechanism to describe scientific data formats. It is not a data exchange format itself.
- **NcML** (NetCDF Markup Language) – XML representation of netCDF metadata.
- **EML** (Ecological Metadata Language) – metadata specification used in ecology to document ecological data.

2.1.3. Metadata

Metadata is a structured, encoded data, used to describe the characteristics (documentation) of associated data for better identification, discovery, assessment and management. In other words, metadata is data about data. Given the dynamic nature of geospatial data, metadata are an essential component in supporting the discovery, evaluation, and

application of geospatial data beyond the originating source (organization) (Nebert, 2005 [161]). Metadata standards increase the value of data by facilitating data sharing through time and space. An example of metadata is a map legend, which contains information about the publisher of the map, the publication date, the type of map, a description of the map, spatial references, the map's scale and its accuracy, etc. The Metadata usually answers to the What, Who, Where, Why, When and How questions of the data (Nebert, 2005 [161]).

Data worth preserving should be preserved with the associated metadata that describes that data, including details about provenance and quality and should also be catalogued correctly so it can be found and used (Lu et al., 2015 [140]).

Geographical metadata is typically an expression of the coverage (spatial) area of a particular dataset. Most of the time this will be a (minimal) bounding box but it may also be a shape of arbitrary complexity (Reid et al., 2012 [186]). The discovery of geospatial data is usually done through metadata catalogs that one can operate to search and brows for specific items of interest. Metadata are typically made available both for human consumption, in HTML form, and as machine-readable form, often as ISO XML for compliance with international obligations (Reid et al., 2012 [186]).

The OGC - Catalog Service (CS) (Nebert et al., 2007 [162]) specification is the *de facto* standard for discovering information about geospatial data and services in a standards compliant fashion. It defines a standard method for defining, querying and organizing stores of geospatial metadata.

2.1.4. Open Data

Open data is data that can be freely used, re-used and redistributed by anyone, subject only, at most, to give the credit of the owner and possibly to share it in the same way (http://opendatahandbook.org). The first attributes of open data are the access and availability. This means that open data must be freely available (or at most it should have a reasonable reproduction cost) in a modifiable and convenient form. The property of re-use and redistribution of open data implies that the data must be provided under certain terms that would allow such activities as well as the mixing of this data with other datasets. Finally, open data must allow universal participation, without restriction on who can use it (even commercial use should be allowed).

McKee (2010) [148] gives 18 powerful reasons for open publication of Geoscience data, such as: data transparency, verifiability, useful unification of observation, cross-disciplinary studies, longitudinal studies, re-use, planning, return on investment, due diligence, value maximization, data discoverability, data exploration, data fusion, service chaining, pace of science, citizen science and outreach, forward compatibility, and timely intervention.

Open data is about how to unlock the potential of information to enable new services, new value, and to improve the decision making process for a better sustainable development. The European Commission supports Open Data initiatives to freely make available public data for use and reuse. Although the open data movement has begun, effective open data policies and practice depend on open interfaces and encoding standards, able to provide efficient publishing, discovery, assessment, access and use of data (Lu et al., 2015 [140]).

A brilliant example of open data initiatives was the free and open access policy of Landsat data, which showed how to maximize the return on the large investments in satellite missions (Wulder et al., 2012 [244]). The huge benefits of similar Open Data initiatives was emphasized also by Barbara Ryan, the Secretariat Director, Group of Earth Observation (GEO [81]) by pointing out the economic benefits brought back by these initiatives: “The economic value of geospatial data lies in its utility” (Ryan, 2016 [204]).

2.1.5. Linked Data

Linked Data refers to a set of best practices used for publishing and connecting structured data on the Web in such a way that it is machine readable, its meaning is explicitly defined, it is linked to other external datasets, and other datasets can in turn be linked to it (Grothe and Brentjens, 2013 [98]). Linked data is also advocated in INSPIRE (INSPIRE, 2007 [111]) and uses web technologies such as RDF (Resource Description Framework) (RDF [184]) and HTTP. It aims to explicitly include relations between data in the data itself through Uniform Resource Identifiers (URIs) and vocabularies.

Linked Data is less a technology than a set of best practices for publishing, sharing and connecting data and information.

2.1.6. Data Interoperability

Interoperability is defined as the ability of two or more systems (organizations) to inter-operate (work together) without additional changes. In the case of data, the interoperability refers to the ability of data to intermix (interoperate) with other different datasets. This is an extremely important quality of the data because it allows different components to work together and to build large, complex systems, otherwise not possible without interoperability.

The practical benefits of open data, able to combine different datasets together and to develop new and better products and services are strongly emphasized through interoperability. Communication, large-scale data collection and processing also depend on the ability of data to interoperate through standard-based mechanisms and encodings (Lu et al., 2015 [140]).

2.1.7. Big Data in Environmental Sciences – Big Geo Data

Even though over the past years several definitions of Big Data have been proposed, there is no agreed definition yet (Granell et al., 2016 [95]). Big Data are usually defined not just as massive data sets but also as data having very complex and varied structures, making further actions (e.g., storage, analysis, visualization, processing) very difficult. New satellite, airborne and ground-based remote sensing systems characterized by high spatial, temporal and radiometric resolution are, or will be soon, available. New sources of geospatial data are emerging with the advances of sensors and communication technologies. Traffic detectors on the road, electrical grids or environmental sensors for measuring the air quality, mobile devices such as smartphones, crowdsourcing, are just a few examples (Nedelcu, 2015 [163]). With the launch of three families of Sentinels satellites, Copernicus will be producing for example, 8 TB of Earth Observation data per day (3000 TB per year) (Copernicus Big Data Workshop, 2014 [47]), which will lead to an increase of data volume, diversity and also value. Data from Landsat satellites provides the longest continuous record as seen from space of the Earth's surface, starting with the launch of Landsat 1 in 1972 through Landsat 8 in March 2013 (Santos et al., 2014 [206], Landsat News, 2014 [129]). Landsat 9 is planned to be launched in 2023, maintaining this incredible continuous stream of data. Data is growing at an unprecedented rate, at least by 20% every year (Lee and Kang, 2015 [133]), becoming not only bigger in volume but also increasingly complex, and accessible in a variety of formats. New important opportunities (such as the monitoring of Earth surface changes with greater frequency than ever) are

envisioned but also new challenges especially regarding the access, analysis, processing, and sharing of these data.

Based on this, the main characteristics of Big Geo Data are gathered around the "5V" (Demchenko et al., 2012 [52], Nativi et al., 2015 [160], Kramer and Senner, 2015 [125]):

- **Volume** – available amount of data. Geospatial data sets tend to be very large, gathering data from modern sensor networks, LIDAR scanners and artificial satellites, which can produce several GB to TB per hour (Kramer and Senner, 2015 [125]).
- **Velocity** – rate of data collection. Large geospatial data streams are produced in short amount of time, requiring a faster data processing to get a higher derived value of information.
- **Variety** – Geospatial data is typically heterogeneous, so this property of Big Data refers to the variety of sources producing it but also to the implementation of services dealing with these different types of data. Most of the time, data from different sources has to be combined and different data exchanged formats, reference systems and accuracies have to be considered.
- **Veracity** – validity and accuracy of the data must be taken into account considering that data sources can be of different qualities, especially when it comes to coverage, accuracy and timeliness.
- **Value** – how meaningful the row data is and how valuable is the obtained information (**the main purpose of Big Data is to produce meaningful Small Data**).

Big Data is already embedded in environmental sciences studies and it is mainly produced by three important sources (Yang and Huang, 2013 [250]):

1) From the impressive array of sensors that are placed in space (via remote sensing satellites) and in situ, used to measure and monitor weather, precipitations, vegetation, land cover, water quality, as well as other geophysical parameters. These collections of data sets satisfy all the characteristics of Big Data (the 5 Vs).
2) From the various scientific models simulations used for predicting physical phenomena. Climate change for example can be considered one of the largest use cases of scientific modeling and simulation. Nowadays climate simulations can be run on a daily basis with increasingly higher horizontal (hundreds of meters rather than tens of kilometers) and

vertical (more model layers in the atmosphere) spatial resolution, as well as higher temporal resolution (minutes or hours rather than days or weeks). The update of these models is done more frequently and with much higher quantities of new data. Therefore the amount of data coming out of these simulations is very large, reaching typically petabytes of data from just one simulation. Based on this we can conclude that this data can as well be considered Big Data.

3) From data assimilation, the process by which models are updated with the latest observational data to be able to correct and validate the assumptions made in the model due to different factors like missing parameters, incorrect data, etc.

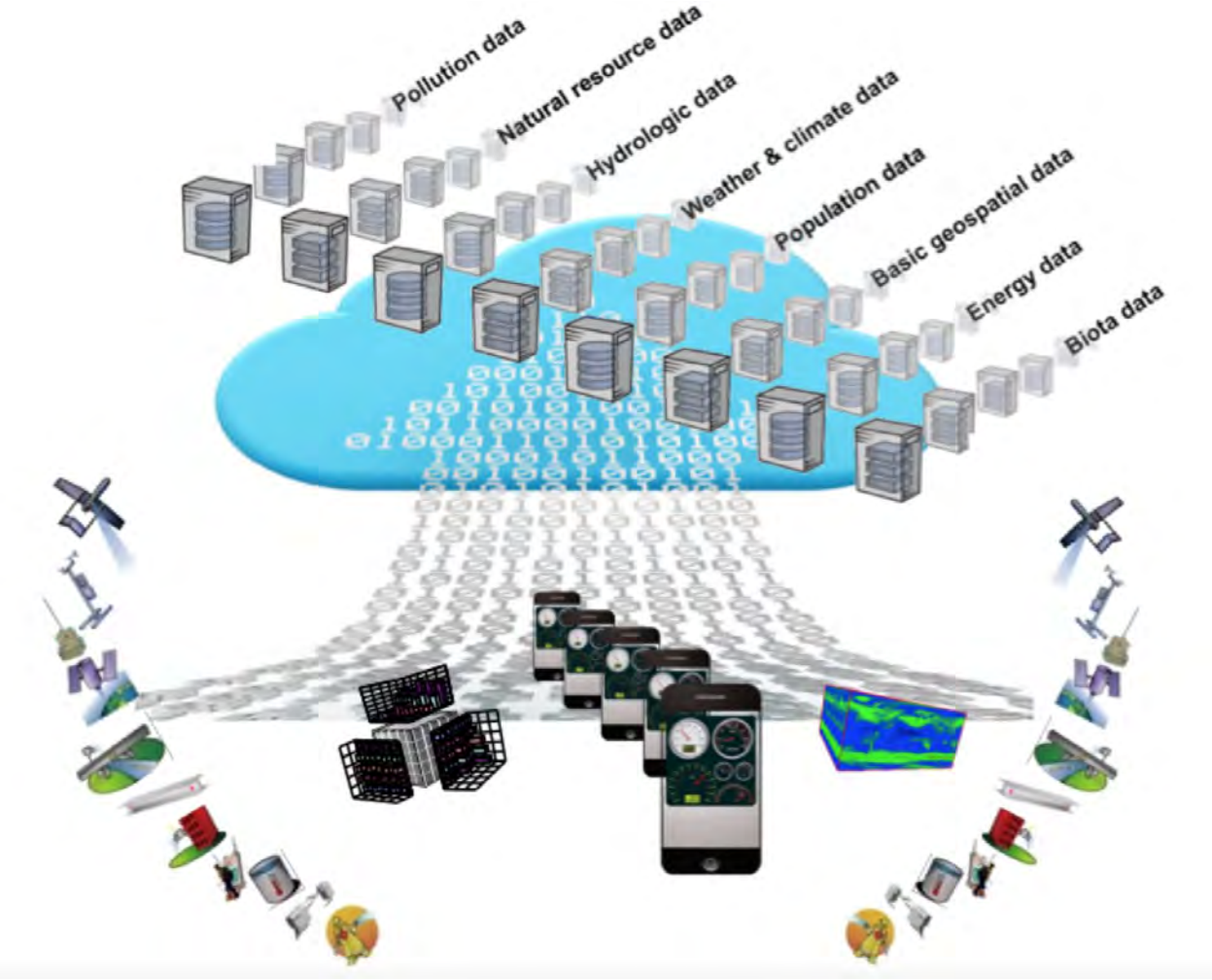

Figure 3: Big Data in Environmental Sciences (Source: Lu et al., 2015)

The rapidly increasing number of terabytes of data, coming from an explosion of sensors, satellites, crowd-sourcing, models, etc. have to be made valuable by making it easily discovered, accessed, assessed, aggregated, combined, open and interoperable (Lu et al., 2015 [140]).

Analysis of this Big Data can give unprecedented possibilities for better decision making for understanding and mitigating the effects of climate changes. Nativi et al. (2015) [160] emphasize the Big Data challenges in Global Earth Observation System of Systems – GEOSS (GEO, 2005 [80]) – and particularly its common digital infrastructure (GEOSS Common Infrastructure - GCI), which will be describe in detail in the next chapter. The presented challenges can be identified along all the Big Data dimensionalities: volume, variety, velocity, veracity and visualization.

Many new research efforts are now directed towards the development of enabling technologies and paradigms to support requirements for Big Data handling (Granell et al., 2016 [95]). Specific solutions for Big Data analytics have been developed, based on mobile code, such as in the European Grid Infrastructure – EGI (http://www.egi.eu/), the Earth System Grid Federation – ESGF (https://www.earthsystemgrid.org) and middleware through optimized SQL extensions, such as EarthServer (http://earthserver.eu/) raster query language or through Cloud solutions dedicated to Earth Science, such as Google Earth Engine – GEE (https://earthengine.google.com/). NoSQL databases, such as Google BigTables allow working with unstructured data, which is a big advantage but the geospatial processing capabilities of these solutions are still limited compared to GIS-enabled relational databases (Yang et al., 2011, 2013 [248], Granell et al., 2016 [95]). Brokered architecture for efficiently connecting existing infrastructures (Nativi et al., 2013 [159]) and providing large amount of heterogeneous resources such as GEOSS and GEO DAB, initiatives described in the next chapter, have also been developed.

To summarize, the main challenges identified in working with geospatial Big Data (Copernicus Big Data Workshop, 2014 [47]):

- Sharing Big Data, using free and open data policy which will provide easy and free access;
- Big Data coming from different sources, with different archival formats, require solutions to increase the speed of data encoding and thus to decrease the management time;

- Big Data cannot be moved easily meaning that the analysis and processing have to move to the data;
- Big Data has to be used in multi-disciplinary research implying a shared understanding of what data is and how it can be used. Not only the data has to be described but also the processes that have to be performed, using new services;
- Big Data dissemination and processing have to consider a broader user community;
- Big Data requires: new approaches, services and concepts as well as good technical and programmatic coordination;
- Big Data requires Big processing infrastructures.

In the context of these challenges, the main research directions are towards (Loekken and Farres, 2014 [138]):

- **Open Data** – all the data should be discoverable and accessible for free;
- **Open Computing** – users should have access to perform the data processing online (on different processing infrastructures);
- **Open Source Software** – all the platforms and software used for Big Data processing should be open source;
- **Open Collaboration** – sharing of both data and applications among users;

2.2. From Environmental Data to Knowledge and Wisdom

2.2.1. Introduction

Understanding the Earth System, having a more up to date, clearer picture of the world, has become an essential condition to respond to the current global changes (climate change, land use change, deforestation, loss of biodiversity, pollution, urbanization, etc.) that are threatening the natural environment and the society at large. Understanding such a system and predicting future behavior is a complex and challenging mission that requires not only a sustainable Earth observation, through satellites, airborne platforms, sensors, etc., but also an increased understanding of its components, processes and their interactions.

The ever increasing amount of geospatial data, resources and processing functions, available on the Web due to the IT technologies advances, has reached tremendous proportions and requires effective and efficient data processing methods and tools for geospatial information extraction and knowledge discovery.

Raw data was never good enough as it simply exists and has no meaning of itself. Data represents a fact or a statement of a phenomenon/event without relations to other things. The information is actually seen as the data that has already been processed, organized data, or data that has been given meaning while knowledge goes beyond that and uses the information (the meaningful data) to describe a deterministic process, extracting useful patterns (Bellinger, 2014 [26]). But is knowledge enough or do we need something more? Besides having the data, extracting what is useful out of it (information) and having the knowledge to apply the information in a useful manner (actionable information), people also need a vision, a way to synthesize new knowledge from what is already known and this is what we call wisdom.

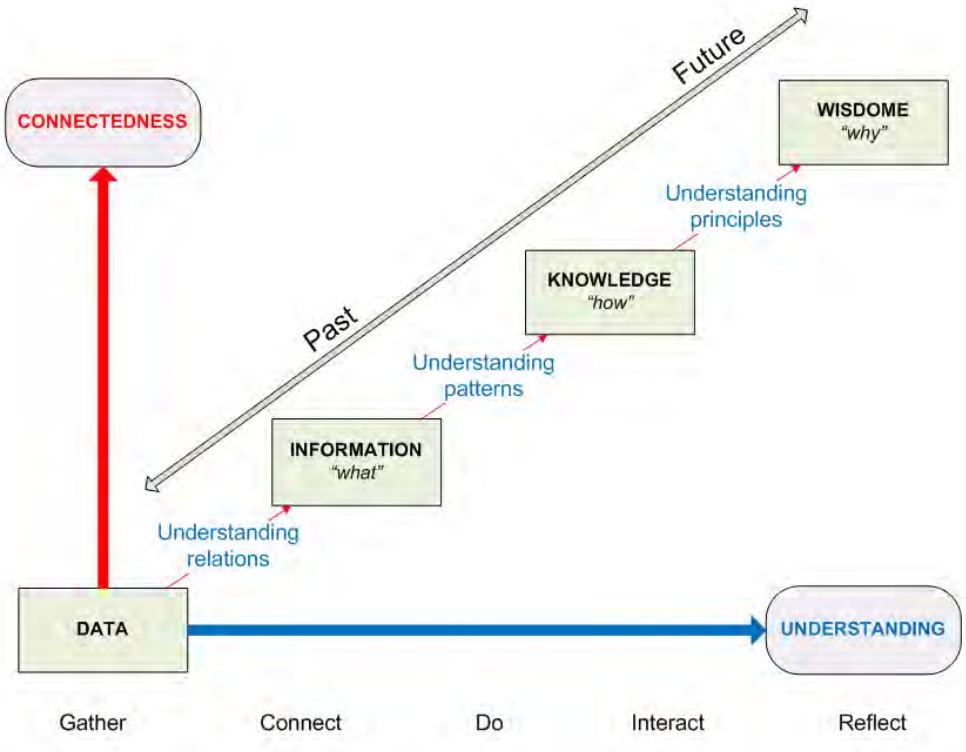

Figure 4: A Data-Information-Knowledge-Wisdom (DIKW) view

According to (Bellinger, 2014 [26]), "wisdom embodies more of an understanding of fundamental principles embodied within the knowledge that are essentially the basis for the knowledge being what it is". Russell Ackoff (Ackoff, 1989 [4]) is known as the father of the DIKW (Data Information Knowledge Wisdom) hierarchy end explained the elements of its layers and their interconnections. An interpreted view of this pyramid is illustrated in Figure 4.

The question that everybody should try to answer, not only in Environmental Sciences field, but in general, is "How can we more **effectively** get what we want?" but nowadays everybody is focusing on answering the question of "How can we more **efficiently** get what we want?" The distinction that one has to make here is in the difference between **effective** and **efficient**. According to (Ackoff, 1989 [4]), being effective is about doing the right things while being efficient is about doing the things right. Wisdom is concern with effectiveness but achieving wisdom isn't easy and one must move successively through the other categories to reach this level. Considering the tremendous amount of available data nowadays, and the percentage of geospatial data out of that, we need **efficient** methods to be able to get **effective** results.

2.2.2. Data Discovery and Access

Discovery of geospatial data involves the use of services such as metadata catalogs to find data of interest over a specific geographic region (Nebert, 2005 [161]). Geospatial data catalogs are discovery and access systems, which usually use metadata to query on raster, vector, and tabular geospatial information. They offer a way to publish descriptions of geospatial data in a standard way. Indexed and searchable metadata are therefore extremely important and provide dedicated vocabulary that can be further used in intelligent geospatial search (Nebert, 2005 [161]). Once the data is discovered, the access process involves the request, the packaging and the delivery of the specified data, either online or offline.

Access to geospatial data is part of a process that goes from discovery to evaluation, to access and finally to exploitation. Effective access and reuse of geospatial data by users and applications is strongly related to multi-source data integration and interoperability (Mohammadi et al., 2008 [152]). Many spatial applications and services model and analyze parts of the environment that require multi-source spatial data, managed by different institutions and ranging from fundamental datasets (such as cadastre, topography, vegetation, roads and imagery to

location data) to socio-economic and infrastructure data (such as demography, valuation, public transport and utilities) (Mohammadi et al., 2008 [152]). Different organizations use different strategies and policies to capture, manage and share their data, leading to many technical issues (inconsistent standards, semantic heterogeneity, poor metadata or no metadata at all, inconsistency in data models, etc.) and nontechnical inconsistencies (inconsistency of institutional arrangements and policies, lack of regulation, different understanding and knowledge, capacity building, etc.) and heterogeneity among datasets.

The integration and interoperability of geospatial data is a complex process and can only be addressed within a well-structured and holistic platform, able to all effective components and issues of geospatial data integration together (Mohammadi et al., 2008 [152]). This platform will facilitate the usage of geospatial data from different sources, within an acceptable time and cost, through the fastest channels. It will also establish interoperability both at technical and nontechnical levels and it will establish effective interactions between different components, including policies, standards, collaboration and access. A Spatial Data Infrastructure (discussed in details in the next section) is a step forward to developing such a platform.

Geo-Web Services (GWS) (Nebert, 2005 [161], Mohammadi et al., 2008 [152]) are significant technical tools, developed based on open standards, which facilitate the integration of multi-sourced datasets but also assess the integrability of these dataset. These services provide tools, services and formats, which comply with interoperability concepts. The assessment of geospatial data through GWS can be done through accessible and comparable measures such as: availability of metadata, format, coordinate system, bounding box, etc.

2.2.3. Data Processing

Data processing services provide operations for processing or transforming data in a manner determined by user specified parameters. The most common processing services available in the Geospatial domain are (Nebert, 2005 [161]):

- **Coordinate Transformation Services** – convert geospatial coordinates from one reference system to another.
- **Image Processing Services:**
    - **Image Manipulation Services** – manipulate images (resize, applying various filters, changing image resolution, color, contrast values, etc.). These services are

usually used in mathematical analysis of image characteristics such as image histograms, convolutions, etc.

  - **Image Exploitation Services** – support photogrammetric analysis of remotely sensed and scanned imagery, plus generation of reports and other products based on the analysis results.
  - **Image Synthesis Services** – create or transform images using computer-based spatial models, perspective transformations, and manipulations of image characteristics to improve different aspects such as: visibility, sharpen resolution, reduce cloud cover effects, etc.
- **Geospatial Analysis Services** – exploit available geospatial information to derive application-oriented quantitative results that are not available otherwise from raw data.
- **Gazetteers –** provide access to geospatial data indexed by place name instead of coordinate locations.

2.2.4. Data Visualization

The visualization of geographic data experienced a shift during the years from physical maps to graphical views and maps provided through online mapping interfaces within SDIs. Having the data in digital form has brought a lot of advantages, such as (Giuliani, 2011a [84]): easy storage and dissemination, facilitation of data exchange among groups / organizations / communities, faster and easier updates and corrections, ability to integrate data from different sources, ability to customize products and services, etc.

Visualization tools can quickly portray a large amount of information, without the need to download the full data set, satisfying the needs of many users. Portrayal services are used to visualize geospatial information by producing rendered output, given one or more inputs, in the form of maps, perspective views of terrain, annotated images, etc. Examples of such services (Nebert, 2005 [161]):

- Map Portrayal Services;
- Coverage Portrayal Services;
- Mobile Presentation Services.

2.2.5. Data Sharing

Users tend to develop their own data sets, specific for particular applications or use cases, instead of using available existing data sets, for several reasons (Nebert, 2005 [161], Giuliani, 2011a [84]):

- The users are not aware of the existence of available data sets that they can use (poor documentation, lack of standards, etc.);
- The users are not authorized to access and/or use data. The authorization problem is of great interest, especially for the organizations and institutes, which are not willing to share their data with everyone;
- The access to the datasets is too complex;
- The data is not trusted (history of the data capture/changes is not available). To be able to make a meaningful interpretation of the obtained data, the users need to know the history of the data, to trust the source of the data but also to be able to integrate it with data coming from other sources. The last issue refers to the standardizations problems, meaning that at the moment not all the data is exposed following well known standards in this area, making thus the integration of data coming from different sources a difficult and complex process;
- Users are not used to sharing data with other groups/organizations;
- Existing geospatial data stored in GIS systems may not be easily exported to other systems (data cannot be easily and meaningfully integrated with data from other sources);
- Data may be dependent on other datasets (which might not be accessible). The users need to know if the data they are using depend on other data and if this is the case, they need to have access to all the dependent data in order to be able to use the initial data.

In an Information Age, shared data has become a resource that contributes to national wealth (Lu et al., 2015 [140]) and the main benefits of sharing data are:

- Open science and new research;
- Data reusability;
- Data longevity;
- Greater exposure to data;
- Generation of value added products.

2.3. Personal Contributions

- Identification of major environmental challenges related to environmental data, from an engineering point of view, based on literature review:
    - Metadata, Open Data, Linked Data, Data Interoperability;
    - Environmental Big Data – issues and challenges;
    - Challenges in transforming raw data into meaningful and understandable information.
- Published Papers:
    - ***Rodila, D.,*** *Ray, N., Gorgan, D. (2015), Conceptual Model for Environmental Science Applications on Parallel and Distributed Infrastructures, Environmental System Research, Vol. 4/23, 2015, http://dx.doi.org/10.1186/s40068-015-0050-1.*

Chapter 3: Spatial Data Infrastructure

3.1. Definition and Concepts

The world we live today is undergoing a process of profound and continuous change, a world that is strongly influenced by IT and communication technologies. A better economically and environmentally management of spatial data could bring important benefits, starting from local levels, all the way up to national, regional and global levels (Rajabifard and Williamson, 2001 [179]). The concept of Spatial Data Infrastructure (SDI) was developed as a solution to these needs and it is used to describe the environment, the underlying mechanisms, technologies, policies and institutional arrangements able to support, in an interoperable and efficient way, the discovery, exchange and sharing of geospatial data and information between stakeholders at different levels, such as government, commercial and non-profit sectors, academia and citizens (Nebert, 2005 [161], Giuliani, 2011a [84]).

SDIs are more than databases as they also provide a list of extra functionalities (Nebert, 2005 [161]):

- Hosting of geographic data and their attributes (metadata);
- Means to discover, visualize, and evaluate the data (catalogs and Web mappings);
- Methods to provide access to geographic data;
- Services and software to support applications of data;
- Organizational agreements – to coordinate and administrate an SDI at local, regional, national and/or trans-national scale;
- Ideal environment to connect applications to data (through standards and policies).

To explore the full potential of geospatial data, an SDI must incorporate a set of components, which will allow users to find, discover, evaluate, access and use data. These components include (Giuliani, 2011a [84]):

- A defined core of geospatial data;
- Known and accepted standards and procedures;
- Databases to store data and associated metadata;
- Policies and practices to promote the exchange and reuse of information;

- Humans (both users and data providers) and technical resources to collect, maintain, manipulate and distribute geospatial data;
- Good communication channels between people/organizations to allow shared knowledge and new partnerships;
- Data dissemination procedures and supporting technologies;
- Institutional arrangements to collaborate, cooperate and coordinate actions.

The essential components of an SDI and their relations and the dynamic nature of such an environment, as presented in literature (Rajabifard and Williamson, 2001 [179], Giuliani, 2011a [84]) are illustrated in Figure 5. The main five components are: data, people, access network, policy, and standards. People are the key component to decision making, which is entirely based on data. The interaction between people and data is facilitated through components such as access network (technical infrastructures), policies (agreed by institutions) and standards. The rapid technological evolution make an SDI a dynamic environment in which all the components have to permanently adapt to provide improved or even new functionalities in the process of sharing and exchanging geospatial data.

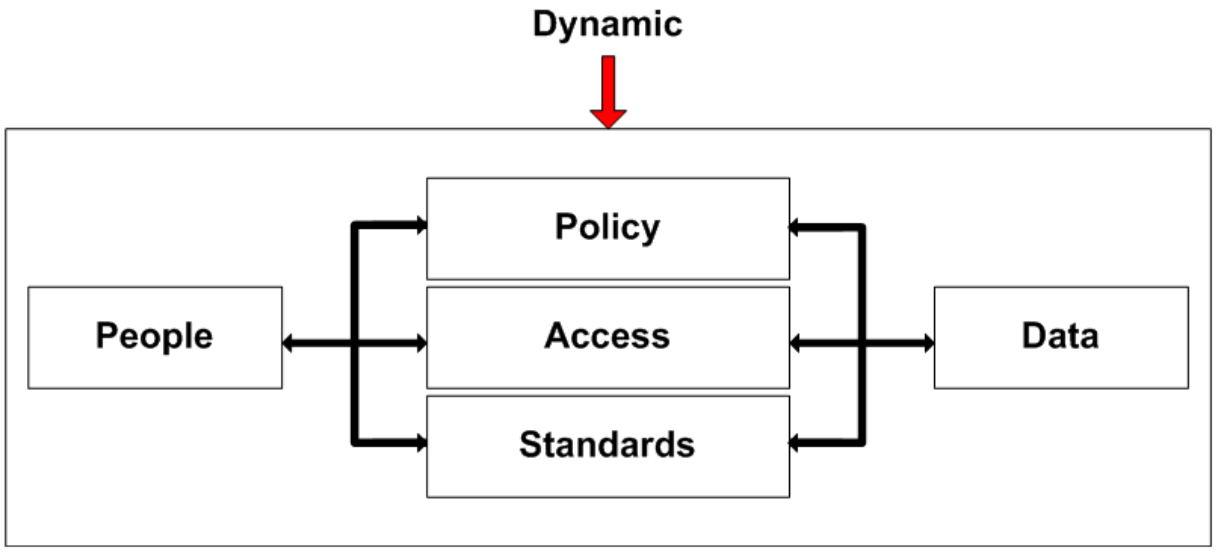

Figure 5: SDI Components: Nature and Relations (adapted from Rajabifard and Williamson, 2001)

The overall objective of an SDI is to maximize the reuse not only of geospatial data and information but also of technical capabilities, skills, invested effort and capital. All these are

done through coordination and a wide range of activities, which involve data, standards, interoperability, delivery mechanism, institutional agreements, policies, financial and human resources. The goal is to avoid duplication efforts and costs and to enable users to save resources, time and effort on acquiring and maintaining datasets (Rajabifard and Williamson, 2001 [179]). The goals and objectives of SDIs come as a support for sustainable development, economic development, environmental management and social stability.

The concept of SDI was developed to facilitate and coordinate the sharing and exchange of geospatial data especially between the boundaries of different countries (Rajabifard and Williamson, 2001 [179]), a very delicate and important issue, but the vision of an SDI incorporates different databases, ranging from local, to national, regional and global levels, into an information network and creates a framework able to make efficient use of geospatial data at different political and institutional levels (Rajabifard and Williamson, 2004 [180]).

There is a growing number of national and regional SDI initiatives (either developed or planned to develop) to encourage sharing and collaboration of geospatial data and practices. The national SDI in the US is mostly build up from the U.S. Federal Geographic Data Committee (FGDC) and the National Spatial Data Infrastructure (NSDI). FGDC was created in 1990, having the goal to "promote the coordinated development, use, sharing, and dissemination of geographic data" (Nebert, 2005 [161]). The National SDI in Australia started from the Australia New Zealand Land Information Council (ANZLIC), the peak inter-governmental body for spatial data issues. An example of regional collaboration is the European Umbrella Organization for Geographic Information (EUROGI), which was set up to foster geographic information outreach and capacity building at the regional level. Their main goal is to support the definition and implementation of a European geographic information policy and to facilitate the development of the European Geographic Information Infrastructure (EGII).

A model of SDI hierarchy is propose by (Rajabifard and Williamson, 2001 [179]) and it is illustrated in Figure 6. This model is made of inter-connected SDIs developed at different levels (from local to global), in which a higher level is formed through the integration of geospatial datasets developed and made available at lower levels. This hierarchical model allows decision makers to create dynamic and hierarchical relationships between any geographical levels and to uses this data for better informed decisions with an important impact across national boundaries.

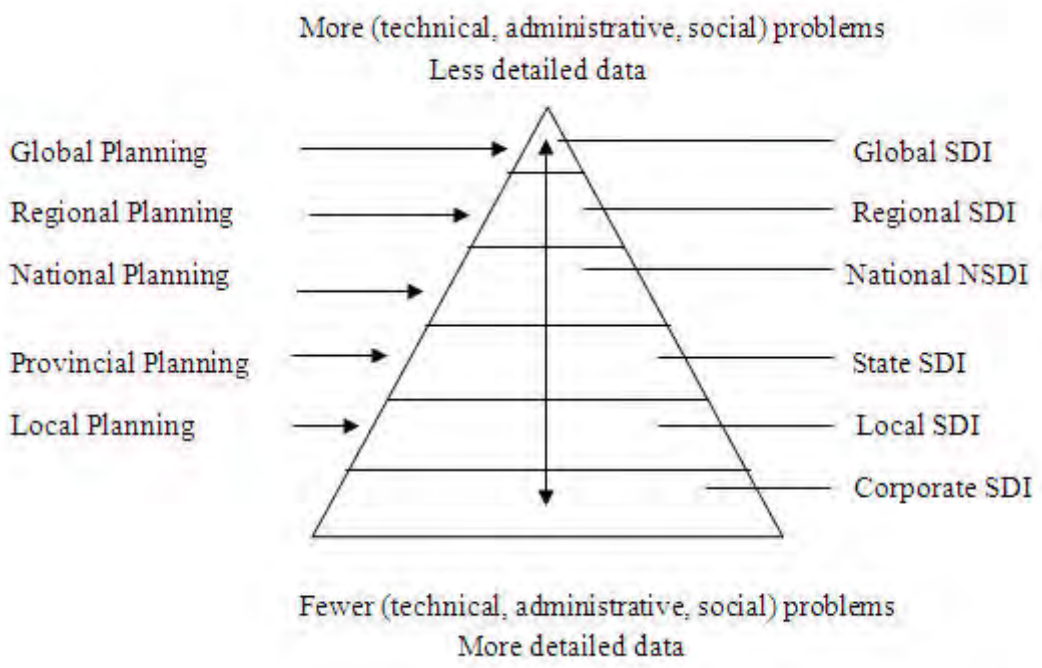

Figure 6: SDI Hierarchy (Source: Rajabifard and Williamson, 2001)

In literature (Rajabifard and Williamson, 2001 [179], Giuliani, 2011a [84]) there are two views of this hierarchy:

- An umbrella view, in which the SDI at the higher level gathers all the components of SDIs at lower levels;
- Building bock view, in which any level of SDI is a building block supporting the provision of spatial data at a higher level.

There are a number of obstacles and issues that can slow down the development and the efficient use of an SDI (Nebert, 2005 [161], Adams and Gahegan, 2014 [5]):

- Lack of standardized metadata;
- Poor data documentation;
- Lack of institutional co-ordination;
- Insufficient flow of information;
- Duplication of activities and results;
- Overlapping of initiatives;
- Poor resource management;

- Poor qualification of the technical qualifications;
- Integration of new, highly heterogeneous data sources (local sensor networks and crowd-sourcing).

As discussed before, the rapid technological evolution make an SDI a dynamic environment in which all the components have to permanently adapt to provide improved or even new functionalities in the process of sharing and exchanging geospatial data. One of the changes that next-generation SDIs have to keep up with is the incorporation of data from highly heterogeneous non-traditional sources, such as local sensor networks and crowd-sourced message databases (Adams and Gahegan, 2014 [5]). These data can come in many forms, can be generated by a variety of producers, using different processes, they can be originally intended for different purposes, they might have variable, loosely defined, and sometimes unknown provenance, semantics, quality and context. Even though these data are not produced by authoritative agencies, they can represent better coverage of specific geographical phenomena and be more accurate due to their distributed methods of generation (Coleman et al., 2009 [44]).

A next generation SDI will have to consider some important aspects (Adams and Gahegan, 2014 [5]):

- Locate, access and understand the limitations of each used dataset;
- Transform the used datasets into a consistent form (model) by: re-projecting, converting from raster to vector or harmonizing the semantics;
- Combine the datasets using a dedicated analytical workflow;
- Assess the accuracy and reliability of the results (and even publish them back into the SDI).

Possible next generation SDI architecture, able to harmonize heterogeneous data sources, is presented by Adams and Gahegan (2014) [5] (Figure 7). In this case, the SDI acts as a mediator that harmonizes data generated from heterogeneous sources. At the center of this architecture, there are three layers of functionality:

- **Federation and Analysis Layer** – at this level, all supported datasets are descriptively rich, interoperable and can be readily combined to perform different analysis.

- **Mediation Services Layer** – composed of software services to transform (harmonize) geospatial data sources, using supported conceptual models.
- **Knowledge Representation Kernel** – used to describe and create rich descriptions of geospatial knowledge, the conceptual models of geospatial information (which passes through various exchange formats and analysis methods) and their semantics.

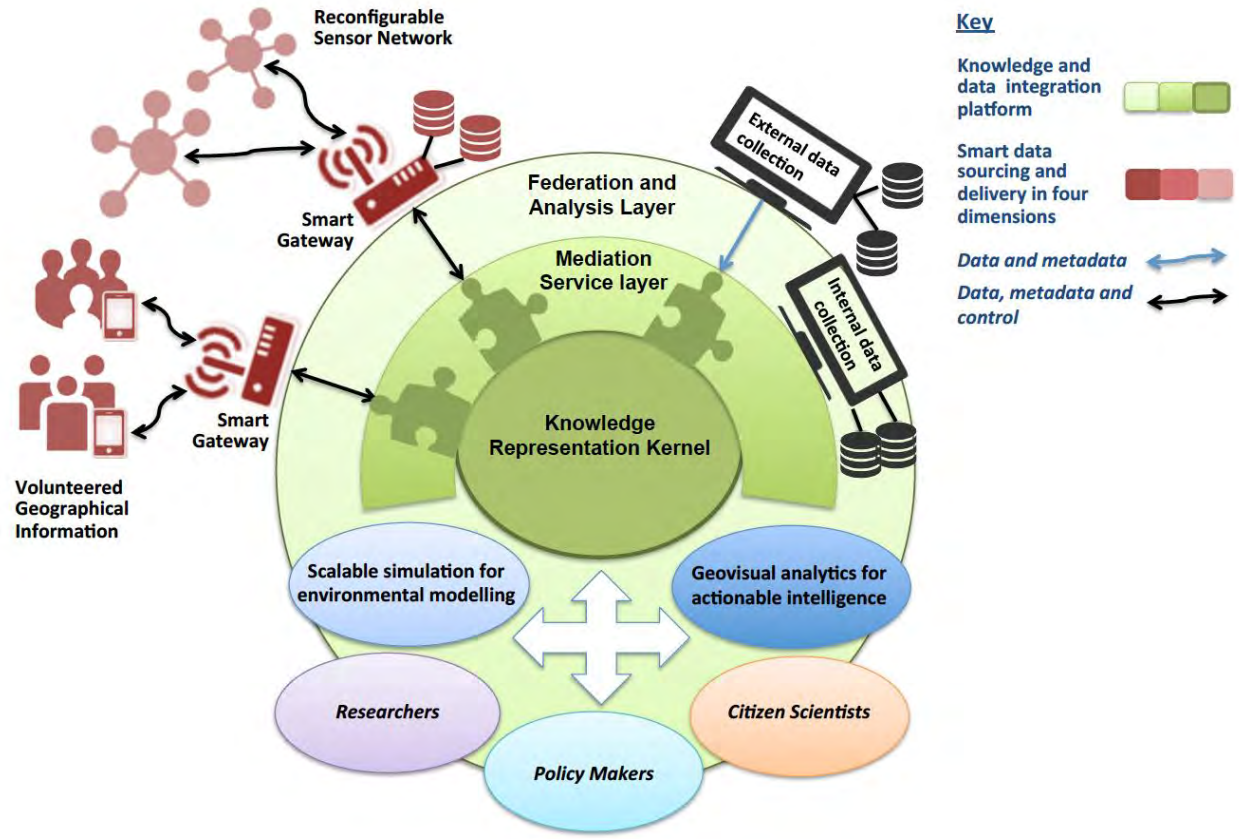

Figure 7: Next Generation SDI Architecture to Harmonize Heterogeneous Data Sources (Source: Adams and Gahegan, 2014)

SDI efforts around the world are motivated by the idea that shared data is an integral part of infrastructure and a resource that contributes to the national wealth in this Information Age we live (Lu et al., 2015 [140]).

Although the SDIs have a lot of important benefits for the Geospatial community, one important drawback is the lack of analytic capabilities, which is an essential step needed to turn raw data into understandable and meaningful information. This means that the SDIs are not

capable of processing geospatial data and usually this is done on desktop computers which have limited resources and which are not capable of handling such a vast amount of data, process that requires a large number of both storing and processing resources for obtaining an acceptable execution time (Giuliani, 2011b [85]).

3.2. Standards

The interoperability is a highly important goal to achieve in the Geospatial domain. For this reason there are several organizations involved in publishing standards to effectively achieve the goal of interoperability, heading to a "geo-enable" Web (Giuliani, 2011b [85]). Due to the incompatibility of data problems, which are making the interoperability between systems hard to achieve, the need for standard services capable to provide the stakeholders the information in a standard, easy and effective way, has become more demanding. The use of common conventions and technical agreements gives the possibility to local communities, nations and regional decision-makers to easily discover, exploit, and share information but it also helps to limit the cost of data integration from various sources and to eliminate the need to develop dedicated tools for data conversion (Nebert, 2005 [161]).

The main drawback in developing "open standards and protocols" is that it is time-consuming and requires long-term support (Taylor et al., 2004 [224]). Usually this is not done unless a large corporation or sometimes a government agency supports it. According to Taylor et al. (2004) [224], a standard is declared only when two independently developed, interoperating implementations, based on that standard, exists.

Standards and the consensus standards process are powerful agents for positive change. Standards are developed and evolve to meet market needs as those needs are perceived by the standards' developers, but the standards probably only survive if those perceptions match the actual reality and evolving needs of the market (McKee, 2010 [148]).

Open Geospatial Consortium (OGC [167]) and the International Organization for Standardization (ISO [113]) provide a range of standards for a Web service architecture that can handle spatial data in a specific way. Using these standards, the SDIs can be linked to form a network, providing thus more access to geospatial datasets (Kruger and Kolbe, 2008 [127]). A major role in supporting and promoting these standards is played by the INSPIRE (INSPIRE, 2007 [111]), involved in building the framework necessary for sharing the geospatial data among

communities, organizations and even worldwide (Padberg and Greve, 2009 [172]). The implementation of OGC specifications is a step forward into the process of sharing and making the spatial information accessible to different communities (Rodila and Gorgan, 2010 [195]) but also a step forward in achieving an interoperable environment.

3.2.1. Open Geospatial Consortium (OGC)

The Open Geospatial Consortium (OGC) (OGC [167]) is a non-profit, international consortium that is leading the development of open and publicly available standards for geospatial and location based services – OpenGIS Web Services (OWS) as well as for software application programming interface for the geospatial community.

In early 1990s, each GIS vendor had their own formats for publishing and exchanging their GIS data so a strong market requirement for a standard way of exchanging GIS data content was clearly needed and this was the start idea of the OGC (Reed et al., 2015 [185]).

The OGC standards are mostly based upon the HTTP protocol and interact through messages over the Internet but new trends also consider including the usage of SOAP (Simple Access Object Protocol) protocol and WSDL (Web Service Description Language) due to the large number of incompatibilities with standard Web services. These standards offer thus possibility to create complex geospatial services and applications, accessible to a wide variety of users and share data in a standardized and interoperable way. This consortium was developed with the purpose of sharing geospatial data and services among different geospatial processing platforms.

The main direction in the development of SDIs components focuses on the exchange of geospatial data in an interoperable way, using services capable to allow efficient access to spatially referenced data. The concept of Web services has been defined by Comet in (Comet, 2004 [45]) as a "new paradigm" in which different providers offer some services, for certain users, allowing an easy access to distributed data and underlying the necessity of two systems to communicate with each other in an easy and cost effective way. A traditional Web service is able to describe and expose the functionalities it offers, providing a standard way for communicating with it and offering thus possibility to other services and application to interoperate with it. In a similar way, through OGC standards, different Geographic Information System (GIS) applications can work together, exchange information over a network and interoperate.

Accessing and integrating geospatial data from different data sources is a challenging task or even impossible without interoperability and standardization, limiting organizations to work with specific software packages and fragmenting the geospatial data sources (Giuliani, 2011b [85]).

The main goals of OGC are (OGC [167]): 1) provide free and open standards to the market, 2) lead the creation and establishment of standards that allow geospatial content and services to be easily integrated into different processes, 3) facilitate the adoption of open, spatially enabled reference architectures, 4) support the formation of new and innovative applications for geospatial technologies through advanced standards, and 5) accelerate market assimilation of interoperability research through collaborative processes.

3.2.2. International Organization for Standardization (ISO)

The International Organization for Standardization (ISO [113]) is an independent, non-governmental international organization, with 161 national standards bodies, responsible for developing relevant International Standards, supporting innovation and providing solutions to global challenges.

Most of the existing standards already have a great deal in common with each other and the ISO standard has accommodated most of the various international requirements (Nebert, 2005 [161]). ISO international Standards cover almost every industry and have an impact everywhere and to everyone (ISO [113]).

ISO has a Technical Committee, TC 211, dedicated to the standardization of abstract concepts related to geospatial data, services, and the geomatics field in general (Nebert, 2005 [161]). These standards may specify methods, tools and services for data management, acquiring, processing, analyzing, accessing, presenting and transferring data in digital/electronic formats between users/systems/locations.

ISO 19115 (ISO 19115, 2014 [114]) provides an abstract or logical model (comprehensive vocabulary and structure) for the organization of geospatial metadata (used to characterize geographic data). This standard presents the schema required for describing geographic information and services by means of metadata. It provides information about the extent, the quality, the spatial and temporal access, the content, the spatial reference, the portrayal, distribution and other properties of digital geographic data and properties.

ISO 19115 defines:

- Mandatory and conditional metadata sections, metadata entities, and metadata elements;
- The minimum set of metadata required to serve the full range of metadata applications (data discovery, determining data fitness for use, data access, data transfer, and use of digital data);
- Optional metadata elements - to allow for a more extensive standard description of geographic data, if required.

The principles of this standard can be extended to other types of resources such as: maps, charts, textual documents as well as non-geographic data.

ISO 19139 (ISO 19139, 2007 [115]) standardizes the expression of 19115 metadata using the Extensible Markup Language (XML) and includes the logical model (UML – Unified Modeling Language) derived from ISO 19115. The development of national and discipline-oriented profiles of ISO 19139 will facilitate the exchange of information using common semantics and syntax (Nebert, 2005 [161]).

3.2.3. World Wide Web Consortium (W3C)

The World Wide Web Consortium (W3C [233]) is an international consortium responsible for the development of common protocols and specifications to support the evolution of the World Wide Web. Since 1994, W3C published more than 110 Recommendations aiming to discover the full potential of the World Wide Web by developing protocols and guidelines to ensure long-term growth (Giuliani, 2011a [84]). The web interoperability is essential for W3C, as this will allow the usage of compatible technologies. In the field of spatial data access, W3C woks on the Web graphic file formats, XML and metadata (Nebert, 2005 [161]).

3.3. Tools and Services

3.3.1. Geographic Information System (GIS)

A Geographic Information System (GIS) is a tool capable to integrate data coming from different research fields such as: space science, survey and mapping science, geography, information science, computer science, environmental sciences and management science (Xiao et al., 2001 [246], Ondieki and Murimi, 2004 [168]). GIS is a computer system capable of

assembling, storing, manipulating, and displaying geographically referenced information from different sources, offering therefore new possibilities to reuse existing data (Giuliani, 2011a [84]). GIS is therefore a new effective technical system (in complex processing and analysis of spatial data), capable not only of managing data, text information and graphs, but also of integrating and analyzing spatial data from different sources, with different formats, structures, projections and resolution levels.

GIS technologies have evolved from a traditional model of stand-alone systems, having the spatial data tightly coupled with the systems used to create them, to an increasingly distributed model based on specialized, interoperable and independent GIS services. This evolution was influenced by factors such as (Nebert, 2005 [161]):

- A growing role of GIS in organizations;
- An increasing availability of spatial data;
- Growing benefits of spatial data sharing and reuse;
- Maturity of Web and distributed computing technologies;
- Different functionalities needs of GIS users;
- Less costs and complexity to install and pay for unused functionalities;
- Flexibility to offer specialized services.

3.3.2. Geospatial Services (OGC Web Services - OWS)

The OGC Web service (OWS) technology was proposed to overcome the issues caused by the lack of interoperability between geospatial data and the processing systems. The OGC Web services are developed as a set of technologies, standards and interface protocols that allow sharing of geospatial resources in a distributed environment. The OGC Web services process data on demand based on users' requirements and returns the data under different formats, also specified by the users. The retrieved data will match the specified requirements both from content and structure point of view (Di, 2004 [54], Werder and Kruger, 2009 [239], Rodila and Gorgan, 2010 [195]).

3.3.2.1. Catalogue Service for the Web (CWS)

The Catalogue Service (CS-W) (OGC – CSW [49]) provides interface standards to publish, discover, search and query metadata about geospatial data, services or related resources.

CSW uses queryable properties, enabling clients to search for geospatial resources by different attributes such as: subject, title, abstract, data format, data type, geographic extent, coordinate reference system, originator, publisher, purpose etc. (Lee and Percivall, 2008 [132], Giuliani, 2011b [85]).

3.3.2.2. Web Map Service (WMS)

The Web Map Service – WMS (OGC – WMS [242]) provides interface standards to retrieve spatially referenced data dynamically from geographic information. It standardizes the display of information that comes simultaneously from multiple remote and heterogeneous sources (Lee and Percivall, 2008 [132]). In this context, a map means a graphical representation (JPEG, GIF or PNG files) of a geospatial data meaning that the service gives access only to the graphical representation of the geospatial data and not to the actual data. The WMS service is used for mapping purposes and can be combined with other WMS services. A request to a WMS services should contain the geographic layers and the area upon which the layer are applied - the area to be processed. The response to such a request will contain one or more map images.

3.3.2.3. Web Feature Service (WFS)

The Feature Service (WFS) (OGC – WFS [241]) provides a standardized way for accessing raw geographic data over the Web. It provides control over how to actually access the data: the data can be downloaded, analyzed, combined with other data from other Web services, etc. and not just visualized as in the case of WMS. The WSF is normally specified to access vector datasets consisting of features of geospatial data, encoded in Geography Markup Language (GML) (Rodila et al., 2010 [196]). There are both differences and similarities between WFS and WMS. The main difference is that WFS gives direct access to the geometry and the attributes of a selected geospatial data. Using the WFS, the user can work directly with a provided dataset. The main similarity between WFS and WMS is the invocation part. A WFS interface is invoked by a URL and it is able to perform a certain number of operations through which the client can manipulate the data. Based on these types of operations, there are two classes of WFS services (Giuliani, 2011b [85]):

- Basic WFS, through which a client can retrieve and/or query features;

- Transactional WFS, through which a client can create, delete or update a feature. A transaction refers to one or more operations of data manipulation that form a logical unit.

3.3.2.4. Web Coverage Service (WCS)

The Web Coverage Service (WCS) (OGC – WCS [240]) provides standards for accessing raster datasets. In this case, raster data refers to an abstraction of the real world where spatial data is expressed as a matrix of cells or pixels, each such cell containing a value. This service is specified thus to describe and provide multidimensional coverage data. It standardizes the access to spatially extended coverages, usually encoded in a binary format and offered by a server (Lee and Percivall, 2008 [132]). It only gives access to different type of gridded data (such as Digital Elevation Model (DEM), remote sensing imagery, etc.) but does not provide transactional capabilities.

3.3.2.5. Web Processing Service (WPS)

The Web Processing Service (WPS) (OCG – WPS [243]) provides standards for processing and calculations of geospatial data. It can expose GIS functionalities in a standard way over the Internet, having support for SOAP, GET and POST communication (Rodila et al., 2010 [196]). It was created and specified in 2007 having as purpose the distribution of geo-processing functionality among the Web by sharing processes in an OGC-compliant way. In this context, the process refers to either a single processing task or to several tasks encapsulated inside a process (Padberg and Greve, 2009 [172]). This service is the only service able to store intermediate results at an external resource and use it as input data in a later service call (Rodila and Gorgan, 2010 [195]).

The WFS and WCS standards are focusing on data accessibility: WFS allows a client to access vector data while WCS allows a client to retrieve raster data. Using the WPS standard we can extend the capabilities to processing the available data. A WPS service can offer a vast variety of GIS functionalities ranging from a simple calculation to complex models. This service acts as a sort of middleware between the client and the process that runs the calculations and allows users to know which processes are available, to select the required input data and their formats, to create a model and run it, to manage processes (status, storage for the output, etc.) and to return the output once the computation is completed (Giuliani, 2011b [85]).

3.3.2.6. Table Joining Service (TJS)

The Table Joining Service (TJS) (OGC-TJS [225], Grothe and Brentjens, 2013 [98]) is an OGC standard (since November 2010) used to describe and exchange tabular data that contains information about geographic objects. TJS offers a standardized web service interface, which can automatically join tabular data (e.g. statistical data) to geographic data (such as administrative boundaries, postal codes, and statistical units) by distributed access (Figure 8) (Grothe and Brentjens, 2013 [98]).

In addition to the specific OGC standards operation (GetCapabilities), TJS also specifies two distinct set of operation for data access and data joining:

- **DescribeFramework** – allows a client to obtain a list of spatial frameworks for which geo-tabular data is available from the server.
- **DescribeDatasets** – allows a client to obtain general descriptions of the attribute data tables that are available from the server.
- **DescribeData** – allows a client to obtain a list describing the specific data contents of the attribute data tables (i.e. the attributes) that are available from the server.
- **GetData** – obtains a specific set of geo-tabular data.
- **DescribeJoinAbilities** – gets the list of spatial frameworks to which the server can join geo-tabular data, and the forms of output products supported.
- **DescribeKey** – obtains a list of geographic identifiers for a spatial framework supported by the server.
- **JoinData** – requests the joining of a specified geo-tabular dataset to its spatial framework and receives references to the products of that join.

TJS encodes spatial framework data and attribute data in the GDAS (Geographic Data Attribute Set) format. This is an XML format defined in the TJS specification and optimized for joining data using TJS and it is mainly used for attribute data (Grothe and Brentjens, 2013 [98]). The attribute data is in most of the cases stored or exchanged in formats like CSV (Comma Separated Values), spreadsheets or XML and the data access service is responsible for transforming the data from these formats to GDAS. For geographic formats, relevant data formats could be: spatially enabled databases (PostGIS, Oracle spatial, ArcSDE), ESRi shapefiles, GML, etc.

TJS can be considered a supporting concept in SDIs for joining data coming from various distributed sources to view and download datasets. TJS is an open standard that increases interoperability, it is simple and powerful, can be used in service-oriented architectures, it uses the power of distributed computing and it offers an easy way to find data through registries (Grothe and Brentjens, 2013 [98]). This standard is still in its beginnings and some users consider it as just another service specification and another specific format for encoding data but implementations and successful stories of this standard might prove them wrong. Currently, there are only a few TJS implementations up-and-running (publicly available) but it's believed that the adoption of TJS will have an impact on the organizations and on its service infrastructure (Grothe and Brentjens, 2013 [98]) as it has the potential to replace the "manual" data joining operation in the daily practice of data management for thematic mapping and spatial statistics.

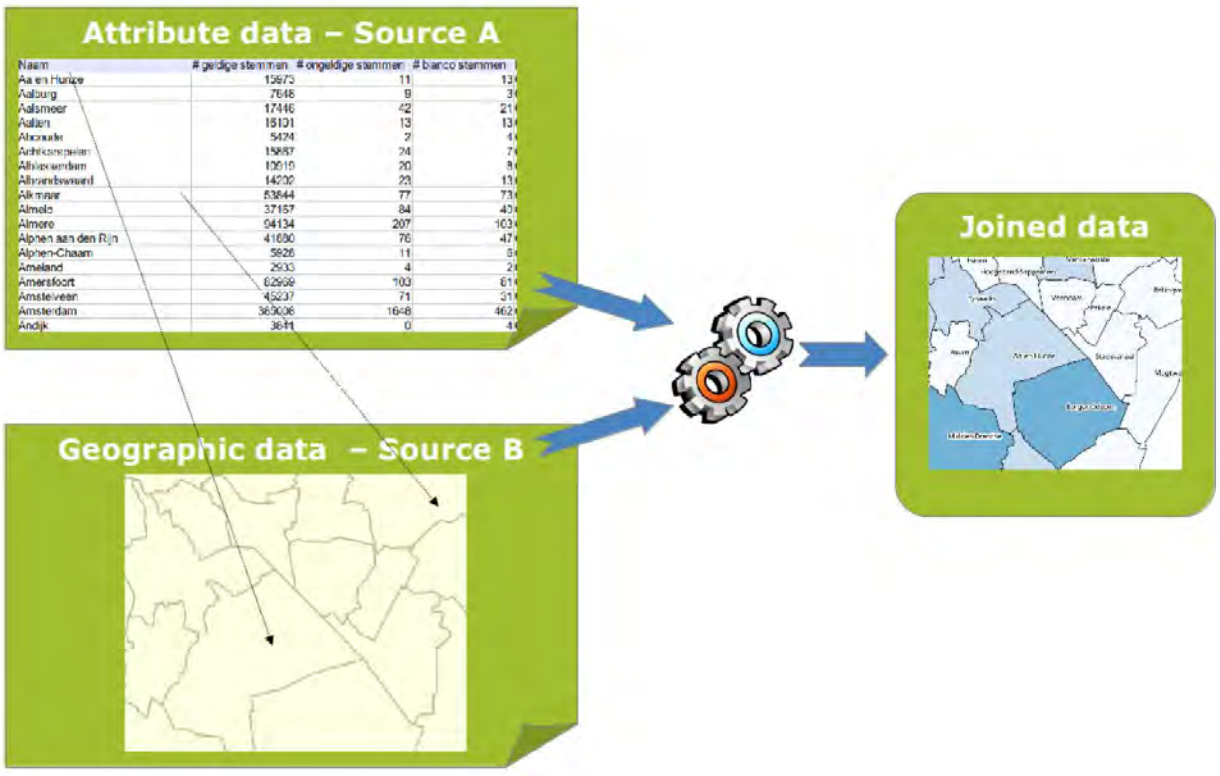

Figure 8: TJS - Joining Tabular and Geographic Data (Source: Grothe and Brentjens, 2013)

3.3.3. Geospatial Databases

3.3.3.1. PostgreSQL/PostGIS

PostgreSQL (http://www.postgresql.org/) is an open source relational database management system (RDBMS) providing functionality to store data and their relations in the form of tables. PostgreSQL itself cannot store geographical information and therefore the installation of a middleware to add support for geographic objects into the database (geo-enabled database) is needed. PostGIS (http://postgis.net/) is one of the software available to work in conjunction with PostgreSQL to add specific functions and tables for geospatial information. PostGIS follows the OGC Simple Features Specification for SQL.

The main features provided by PostGIS 2+ include:

- Geometry types for points, linestrings, polygons, multipoints, multilinestrings, multipolygons, and geometrycollections;
- Spatial predicators for geometries interactions;
- Spatial operators for geospatial measurements: area, distance, length and perimeter;
- Spatial operators for geospatial set operations: union, difference, symmetric difference, buffers;
- Spatial indexes for high speed spatial querying;
- Processing and analytical functions for both vector and raster data;
- Raster map algebra for fine-grained raster processing;
- Spatial re-projection functions for both vector and raster data;
- Support for importing/exporting ESRI shapefile vector data;
- Support for importing raster data from many standard formats: GeoTiff, NetCDF, PNG, JPG;
- Rendering and importing vector data support functions for standard textual formats such as KML, GML, GeoJSON, GeoHash and WKT using SQL;
- Rendering raster data in various standard formats GeoTIFF, PNG, JPG, NetCDF, to name a few using SQL;
- 3D object support, spatial index, and functions;
- Network Topology support;
- Limitations in supporting geospatial “big” data (Lee and Kang, 2015 [133]).

3.3.3.2. Rasdaman

Rasdaman (http://www.rasdaman.com/, http://www.rasdaman.org/, Baumann, 2009 [22], Baumann, 2014 [23]) is an Array Database Management System (DBMS) for big raster data storage that supports multi-dimensional arrays of very large sizes and therefore can handle inherently big satellite imaging data (Karmas et al., 2015 [118]). Conceptually, there is no size limitation for Rasdaman as a central DBMS of raster datasets and it offers a scalable, distributed environment to efficiently process very large number of concurrent client requests through a parallel server architecture which servers distributed datasets across the Web (Figure 9).

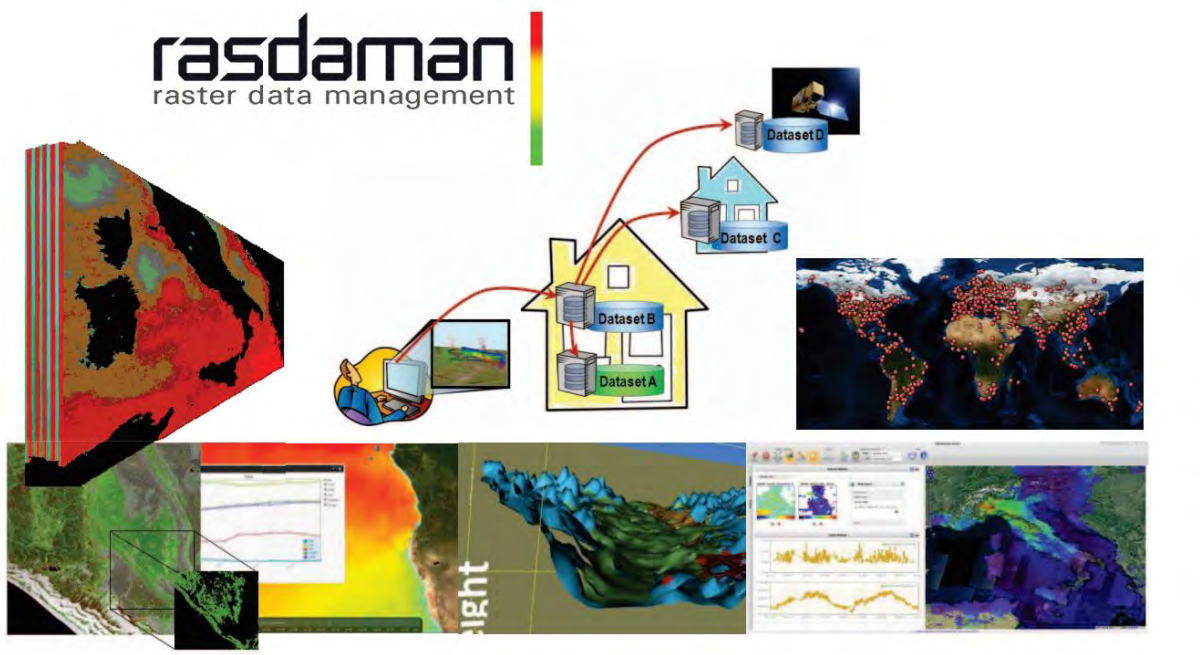

Figure 9: Rasdaman (Source: Baumann, 2014)

The extraction of data from Rasdaman is done through the execution of retrieval queries written in a query language defined by the OGC – WCPS (Web Coverage Processing Service) standard. This language allows the retrieval, filtering, processing and fast subsetting of multi-dimensional raster coverages such as sensor, simulation, image, and statistics data. Existing databases exceed 100s of Terabytes and are heading to Petabytes while array queries have been split across more than 1000 cloud nodes.

The main features of Rasdaman are (Baumann, 2014 [23], http://www.rasdaman.com/):

- Flexibility – any query, any time, from 1D to 4D spatio-temporal data and beyond;

- Scalability – individual dynamic optimization and parallelization for each query;
- Performance – real-time access, processing, mixing and filtering of any-size spatio-temporal data;
- Open Standards support as issued by OGC: WMS, WCS, WCS-T, WCPS;
- Free – available as open source, professionally managed in an open source project, under the OSGeo foundation;
- Cost efficient – through intelligent, economic resource utilization and free source code.

3.3.3.3. SpatialHadoop

SpatialHadoop (http://spatialhadoop.cs.umn.edu/) is an open source MapReduce extension, designed to handle huge datasets of spatial data on Apache Hadoop. SpatialHadoop is designed with built-in spatial high level language, spatial data types, spatial indexes and efficient spatial operations.

3.4. Interoperability

Interoperability is defined by the Open Geospatial Consortium (2004) [167] as "the ability of a system or a product to work with other systems or products without special effort on the part of the customer". In other words, the interoperability is the ability of different information technology components, systems and software applications to communicate (exchange data) accurately, effectively, and consistently but also to use the exchanged information. People and organizations working in an interoperable environment should be able to exchange knowledge (or information) as well as to use the extracted knowledge (information) to generate new one on top of them. Both interoperability and standards have gained an important role in organizations and communities also due to their economic component. They offer an environment which support sharing of data, information and computing resources, reducing therefore the cost needed to spend on developing and maintaining dedicated software and hardware (OGC, 2004 [167]).

The term interoperability was initially defined for information technology while referring to the interoperability of two systems but nowadays we can extend this term to different levels (Reynoso et al., 2014 [188]):

- Technical interoperability;

- Semantic interoperability;
- Organizational interoperability.

Software technology-based barriers to interoperability (Matott et al., 2009 [147], Laniak et al., 2013 [130]):

- Different programming languages, compilers and development platforms;
- Inconsistent separation of system and model components (user and model interface code, executable, algorithmic code, execution management code, warning and error handling, and statistical functionalities);
- Different input/output file formats.

3.5. Capacity Building

Capacity building is an important element for adoption, acceptance and commitment to SDI concepts. The capacity building activities that follow the more technical elements of building an SDI are considerable challenging because they depend on the willingness of people from different organizations and institutions to co-operate.

An important barrier to change is related to the community's capacity to adapt to new standards and technologies; therefore building local capacity is a major constraint to the success of an SDI in many developing countries (Nebert, 2005 [161]). Long-term projects, such as the developing of an SDI and the built-up of a GIS implementation, require not only long term financing but also long term planning in the field of human resource capacity building.

Building capacities on environmental issues can be applied to different categories: decision makers to make better informed decisions, scientists and technical people to better understand the underlying systems complexity and to general public, to better understand the impact of environmental, social and economic issues.

3.6. Initiatives and Projects

SDIs are becoming more and more used in international projects and initiatives. This is an important reason why different initiatives, both at the regional and global level, were already formed, promoting and influencing the creation of SDIs and the use of open standards (Giuliani, 2011b [85]). The main concerns of these initiatives are related to the data access, standardization,

harmonization, interoperability, integration and services. They support and coordinate action for the implementation of common standards and effective mechanisms for development and availability of interoperable digital geospatial SERVICES as well as for tools and technologies to support the users in their purposes. These actions refer to policies, standards, data technologies, organizational structures, delivery mechanisms and additional resources necessary to ensure that the users will be able to work both at regional and global level, without affecting their final objectives and results (Giuliani, 2011b [85]).

All these international initiatives come to support the development of SDIs and the sharing of geospatial data beyond the countries' boundaries by promoting the development, implementation and adherence to common standards able to sustain data interoperability and better efficiency.

3.6.1. INSPIRE

The European Directive 2007/2/EC INfrastructure for SPatial InfoRmation in the European Community (INSPIRE, 2007 [111]) was launched in 2001 and approved in 2007 by the European Environment Agency, Eurostat and the Joint Research Center. This initiative defines a legislative framework to improve usability of relevant, harmonized and quality georeferenced spatial data to support the formulation, implementation, monitoring and evaluation of environmental protection policies as well as infrastructure development. It is a legal initiative that addresses technical standards and protocols, organization and coordination issues as well as data policy issues, including data access and creation and maintenance of spatial information.

The INSPIRE directive will enable the sharing of environmental geospatial information among public organizations, facilitating the public access to data within Europe (EU, 2007 [67]) but also overcome the barriers affecting the data access and data exchange in Europe (INSPIRE, 2007 [111]):

- The collection of data should be done in a single place, only once, and the collected data should be kept in a single place such as to maximize the access efficiency of the others; in many cases the geospatial data is either missing and/or incomplete or IT is collected more than once by different organizations, case in which inconsistencies appear all the time;

- In many cases the geospatial data does not have attached metadata (documentation, description) making it meaningless and with all this it is still shared among users;
- The data collected at one level/scale should be shared at all levels/scales. This will consistently reduce the collecting and processing time;
- The incompatibility of data coming from different sources is a big problem and is a result of incomplete data and lack of standards. This is the reason why there is an urgent need to develop and use common standard for sharing the geospatial data;
- There should be promoting the infrastructures used to find available geospatial information but also to access and use them to meet a particular need. These infrastructures should be publically available and up and running;
- Overcome the barriers for sharing the geospatial data: cultural, linguistic, institutional, financial and legal.

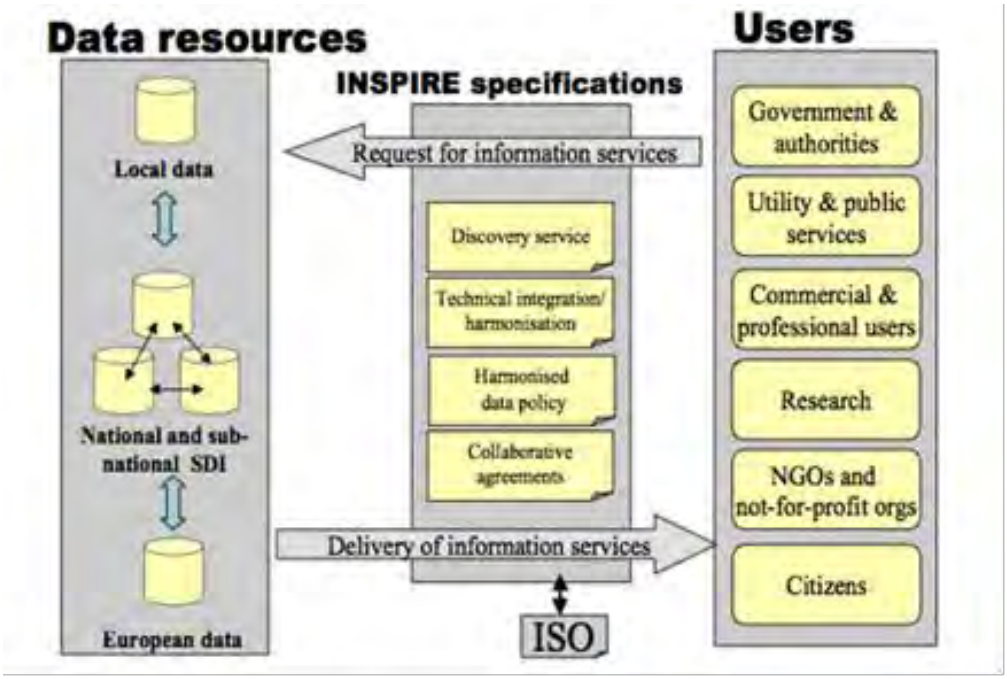

Figure 10: Data and Information within the INSPIRE Framework (source: INSPIRE, 2007)

The INSPIRE directives aims to offer data interoperability, giving the user the possibility to combine geospatial data and services from different sources in a consistent way without any additional efforts. The interoperability envisioned in INSPIRE is also presented in Figure 10.

According to the INSPIRE network architecture, all Member States shall develop and provide access to the following network services, presented also in Figure 11:

- **Registry services**: allow the registration of different data services.
- **Discovery services**: support the search and discovery of data sets and evaluation and use of geospatial data and services through their metadata properties. They provide functionalities for users both to manage and search catalogues for the purpose of discovery and evaluation.
- **View services**: support actions like display, navigate, zoom in/out, pan, or overlay spatial data sets and display legend information and any relevant content of metadata.
- **Download services**: enable copies of complete spatial data sets, or parts of such sets, to be downloaded and, when needed, accessed directly.
- **Transformation services**: enable spatial data sets to be transformed (projection and harmonization) to achieve interoperability. They main function is to help other services in achieving compliance with the relevant INSPIRE specifications.
- **Invoke spatial data services**: allow spatial data services to be invoked. They allow defining both the data inputs and data outputs expected by the spatial service and define a workflow or service chain combining multiple services.

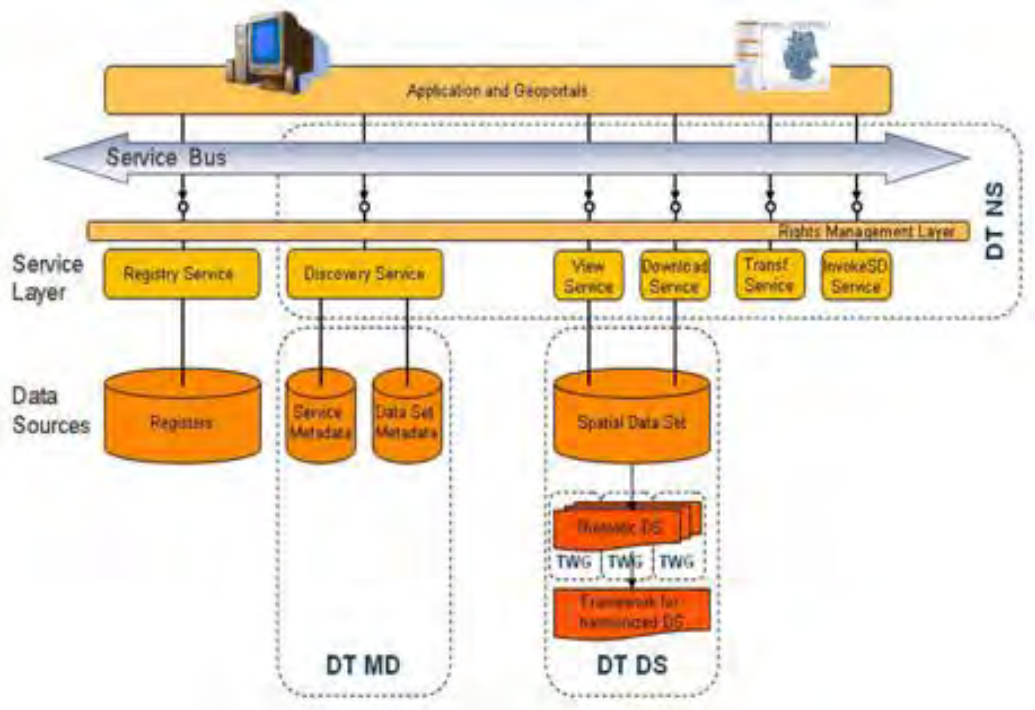

Figure 11: INSPIRE Network Architecture (INSPIRE, 2007)

3.6.2. Open Grid Forum (OGF)

The Open Grid Forum (OGF) (www.ogf.org) is an open community leading the global standardization effort for Grid computing and dealing with the development of Grid standards as well as the extension of Grid community. In 2007 a memorandum of understanding has been signed between the OGC and OGF to support the development of specifications for the geospatial applications on the Grid, applications that have distributed computing needs (Padberg and Kiehle, 2009 [173]). The challenges derived from this integration process spread from technological to conceptual level due to the major differences that exists between the two domains. The interoperability between the Environmental/Geospatial and the Grid infrastructures consists in the integration of OGC Web services into the Grid environment and has to be achieved without altering the functionality and the standardized interface of these services while taking advantage in the same time of the capabilities offered by the Grid infrastructure. Solving the interoperability between Geospatial and Grid infrastructures would bring important benefits and solve important challenges in both areas and especially in EO environment, dealing with processing and storing large amounts of data and performing computationally intensive and complex calculations while achieving high performance (Rodila and Gorgan, 2010 [195], 2011 [197]).

3.6.3. GEO/GEOSS – GEO-DAB

The Group on Earth Observation (GEO [81]) is a voluntary partnership of governments and international organizations launched in 2002 as a response to the calls for actions of the 2002 World Summit on Sustainable development and the Group of Eight (G8). Such an international collaboration is extremely important for exploiting the growing potential of Earth observations to support decision making (Nativi et al., 2015 [160]). GEO has helped establish a cooperation platform among different data providers interested in sustainable development (Lu et al., 2015 [140]).

The Global Earth Observation System of Systems (GEOSS) is the result of coordinating efforts make by GEO to develop a global and flexible network of content providers, allowing decision makers to access a huge range of data and information (Nativi et al., 2015 [160]). GEOSS is composed of contributed Earth Observation systems such as systems for collecting

primary data, systems focused on the creation and distribution of information products, etc., but all GEOSS systems continue to operate independently although the overall GEOSS represents much more than the sum of its components (GEO, 2007 [81]). The main objective of a system of systems is to allow users to perform functionalities that cannot be done with any single component (Bejar et al., 2009 [24]). This is done through a digital infrastructure (GEOSS Common Infrastructure - GCI), which coordinates the access to these systems, interconnects and harmonizes their data, applications, models and products (Nativi et al., 2015 [160]).

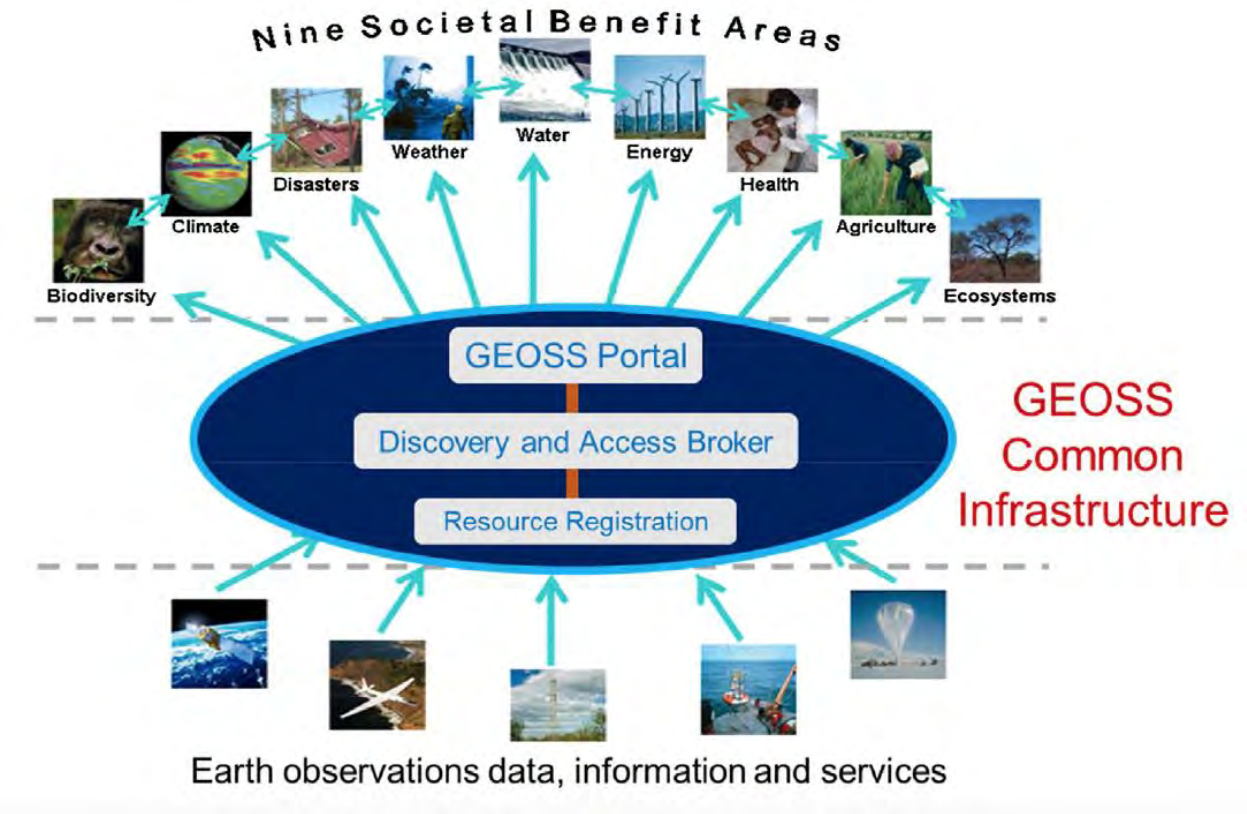

Figure 12: GEOSS High-Level Architecture and GCI (Source: Nativi et al., 2015)

GEOSS is a worldwide effort to build a system of systems and to connect already existing SDIs and EOs infrastructures. This system was not developed to create and/or store data but rather to work and build upon existing systems. To achieve this, the information providers must comply and implement a set of interoperability standards, including technical specification for discovering, collecting, storing, processing and analyzing geospatial data and metadata. This way the GEOSS interoperability is based on formal international standards, being focused on

interfaces and defining only how system components interfere with each other, emphasizing the minimum impact on affected system (Giuliani, 2011b [85]).

GEO Discovery and Access Broker (GEO DAB) is a brokering framework in GEOSS (a key component of the GCI) used to handle the interoperability between systems and the availability of the component services and to intermediate (through an intermediary layer called broker) the bindings between users and providers (Figure 13), providing transparent functionalities for discovery, access and semantic interoperability (Nativi et al., 2015, [160]). The Brokering approach relaxes the requirements for a common data model and common exchange protocols of the participating systems by providing the necessary mediation and transformation functionalities in a transparent way.

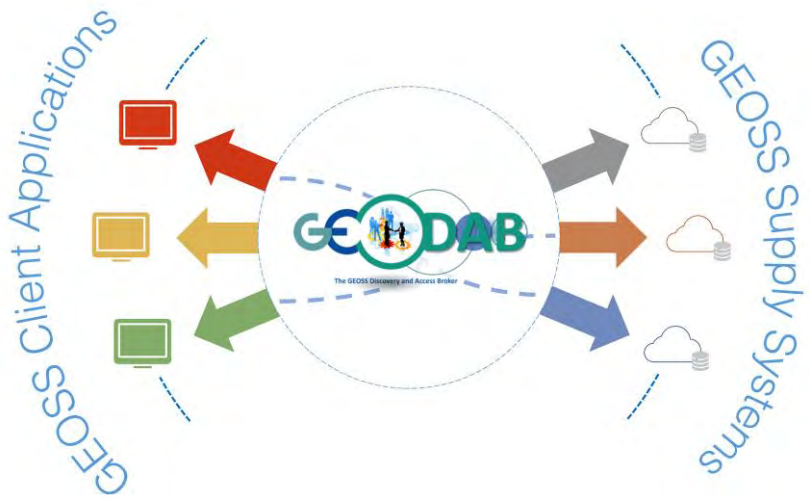

Figure 13: GEO DAB Brokering Framework (Source: www.geodab.net)

3.6.4. Global Spatial Data Infrastructure (GSDI)

The Global Spatial Data Infrastructure (GSDI) Association was set up in 2004 "to promote international cooperation and collaboration in support of local, national and international spatial data infrastructure developments that will allow nations to better address social, economic and environmental issues of pressing importance" (www.gsdi.org). Members of this organization include organization of all kinds, both from public and private sectors as well as non-profit organizations and academia from all over the world. The scope of the GSDI community is to develop and achieve the goal of a Global Spatial Data Infrastructure, relaying on

international and open standards, policies and guidelines and interoperable standards-based services, systems, software and products that operate in a web-based environment .The purpose is to focus on communication, education, scientific research and partnership, supporting all societal needs for access and use of geospatial data

The GSDI is intended to be non-competitive, collaborative, and to build on and unify common activities related to Geographic Information (GI) exchange and harmonization (Nebert, 2005 [161]) and it is seen as a central component in addressing the challenges of global sustainable development. Without a global reference environment, with a consistent set of policies, standards, best practices and co-operating organizations, effective solutions addressing the pressing issues in the global context is not possible.

3.6.5. Sustainable Development Goals – SDGs

The set of Sustainable Development Goal (SDGs) (UN 2014 [228], UN-SDGs [229]) are an intergovernmental set of aspirational and universal global Goals, officially known as "Transforming our world: the 2030 Agenda for Sustainable Development". They include 17 goals (Figure 14), 169 targets and 159 indicators and are considered the biggest attempt in the history of human race to make the world a better place by ending poverty, fighting against climate change, combating injustice and inequality, and preserving biodiversity.

Figure 14: Sustainable Development Goals (source: https://sustainabledevelopment.un.org/sdgs)

The implementation of the SDGs requires, beside national statistics, national spatial data infrastructure (NSDI) and the provision of reliable fundamental geospatial data (Scott and Rajabifard, 2016 [208]), also coordinated global monitoring and modeling of many factors (Lu et al., 2015 [140]): social, economic and environmental, realized through the power of Earth Observation data and techniques together with an increasing computing and storage resource provision. "Science, technology and innovation, and in particular ICT, have been identified as crucial to implementing the Sustainable Development Goals" were the words of the president Mr. Mogens Lykketoft during the 70th session of the United Nations General Assembly in 2015.

The key question in this context is "how can geospatial information be implemented and integrated, at a policy level, to contribute more holistically to measuring and monitoring the targets and indicators of SDGs" (Scott and Rajabifard, 2016 [208]). Geospatial information and Earth Observations are able to provide new and consistent data sources and methodologies to integrate information from various sources, fill data gaps and/or improve the temporal and spatial resolutions of data.

The success of SDGs also requires monitoring and evaluation procedures, standards and metrics to measure the progress towards the targets at local, national, regional and global level (Lu et al., 2015 [140]). Five priorities are emphasized by Lu et al. (2015) [140] for how scientific community should participate in the development of the SDGs:

- **Design diverse metrics** – practical indices for tracking progress on each SDG. These metrics can have the roots in existing methodologies such as environmental impact assessment, natural asset valuation, cost-benefit analysis, and life cycle costing. For these metrics to be properly defined, the goals have to be measurable, comparable and achievable.
- **Establish monitoring mechanisms** – decide which values (such as water and energy consumption, emissions and health impacts, etc.) have to be tracked and what systems are needed/able to do this (how and by whom as well). This step requires global collaboration between governments and scientific bodies not only to establish these monitoring mechanisms but also to implement them, especially in developing countries.
- **Evaluate progress** – choose appropriate criteria to judge the progress towards the goals, based on accepted principles of good practice or governance. The evaluation of the

performance and implementations of SDGs should be done every 3-5 years through a peer-review mechanism established through the UN platform.

- **Enhance infrastructure** – for a better global coverage and for better storage, analysis, processing and sharing of data. This can be achieved through better exploring the capabilities of Earth observation, ground-based monitoring and information processing. Geographic Information Systems (GIS) should be use to host and share data, while image processing, simulation and decision making tools should be used to support sustainability planning, management and enforcement. Capacity building is also an important aspect, especially in developing countries for better observing, data mining and statistics.
- **Standardize and verify data** – to avoid collecting wrong or useless information. The cooperating authorities and agencies should agree on definitions, specifications, methodologies and formats to collect data and also on quality control services for data coming from different sources. All SDG data must be open access and released as soon as possible.

Sustainable development will provide a tangible political "trigger" to foster and accelerate the development and adoption of legal, technical, geospatial and statistical standards; openness and exchange of data and metadata; interoperability of data and information systems; and integration of statistical and geospatial information (both management and exchange) (Scott and Rajabifard, 2016 [208]).

3.6.6. Digital Earth

Digital Earth (Digital Earth [56], Goodchild et al., 2012 [89], Mahdavi-Amiri et al., 2015 [144]) is a global initiative to construct a comprehensive virtual representation of the planet. This initiative is a collaborative effort between Earth sciences, space sciences and information sciences to monitor and forecast natural and human phenomena. The International Society for Digital Earth (ISDE) (http://www.digitalearth-isde.org/) is a non-political, non-governmental and not-for-profit international organization, having as main goal the promotion of academic exchange, science and technology innovation, education, and international collaboration towards Digital Earth.

The vision of Digital Earth was created back in 1998 by a speech of then Vice-President Al Gore, who communicated a vision for the future and the way citizens would interact with global information resources to better comprehend the complexity of our planet and our interactions with it: "a multi-resolution, three-dimensional representation of the planet, into which we can embed vast quantities of geo-referenced data" (Gore, 1998 [90]). This initiative involves a national and international effort to plan and build a cooperative use, internet-based infrastructure to use large amounts of geo-referenced data and information resources, Earth science data, and cultural and historic data (Nebert, 2005 [161]).

The creation of a digital representation of the Earth and its associated data is a complex and difficult task due to a series of factors such as the incredible size of geospatial data, differences between data sets, complexity of globe representation and visualization, etc. (Mahdavi-Amiri et al., 2015 [144]).

One of the main challenges of Digital Earth is to construct the organizational structure able to enable citizens, industry, academia, and government interaction in developing this initiative. A strong public-private partnership to link industry and other non-government organizations with government must also be established (Nebert, 2005 [161]).

Digital Earth can be seen as an initiative for a 3D representation of the Earth for integration, analysis and visualization that has emerged to facilitate solutions to data integration and analysis (Mahdavi-Amiri et al., 2015 [144]). In such a system, data are assigned to locations on the 3D Earth (approximated using either a sphere or an ellipsoid), through a variety of techniques. The Earth can be discretized into cells (using either latitude – longitude parameterization or a refined polyhedron projected to the sphere), and each such cell represent a particular region with a unique index (address), used for fast data access and/or hierarchical or adjacency queries (Mahdavi-Amiri et al., 2015 [144]). Discretization of the Earth into multi-resolution hierarchy of indexed regular cells are known as Discrete Global Grid Systems (DGGSs) and represent the backbone of the Digital Earth systems.

There are several identified technology development areas that need to support the development of such an initiative like Digital Earth (Nebert, 2005 [161]):

- Computational Science:
    - high-speed computing for modelling and simulations;
    - integration and overlaying of diverse sources of geo-referenced information;

    - interactive 3D visualization;
    - display and navigation;
    - computation of information products on demand.
- Mass Storage – distributed active archives, with real-time access of large, multi-resolution data sets.
- Satellite Imagery – 1 meter to 1-kilometer resolution for the planet.
- Broadband Networks – high-speed networks and public access nodes for transmission, interaction and collaboration.
- Interoperability – using well defined standard and protocols.
- Metadata – for automatic data documentation, discovery and preservation.

The Digital Earth initiative is strongly influenced by the continued progress of national, regional and the global SDI initiatives as well as other geospatial data programs.

3.6.7. EarthCube

EarthCube (https://www.earthcube.org/, Gill et al., 2014 [83]) is a joint initiative between the National Science Foundation (NSF) Directorate for Geosciences (GEO) and the Division of Advanced Cyber Infrastructure (ACI), which began in 2011. EarthCube is an evolving and dynamic virtual community of more than 2500 contributors, including earth, ocean, polar, planetary, atmospheric, computer, and social scientists, data and information professionals.

The goal of this initiative is to enable geoscientists to address the challenges of understanding and predicting a complex and evolving solid Earth, hydrosphere, atmosphere, and space environment systems. This goal should be achieved through the development of a common cyber-infrastructure for supporting collecting, accessing, analyzing, sharing and visualizing all forms of data and resource.

Increased access to and use of advanced technological and computational capabilities should be used to obtain enhanced knowledge and understanding of, and ability to predict the Earth System. EarthCube supports standards for interoperability, promotes the usage of advanced technologies to improve and facilitate interdisciplinary research, and helps educate scientists in the emerging practices of digital scholarship, data and software management and open science.

All these activities foster a sustainable future through a better understanding of our complex and changing planet.

3.6.8. Future Earth

Future Earth (http://www.futureearth.org/) is a ten-year initiative for global sustainability research, launched at the Rio+20 UN conferences in 2012. This initiative proposes components such as: observing networks, high-performance computing, Earth-system models, theoretical frameworks, data-management systems, and research infrastructures, needed to track human dimensions and societal changes (Lu et al., 2015 [140]).

The Future Earth research initiative will develop the knowledge for responding effectively to the risks and opportunities of global environmental change and for supporting transformation towards global sustainability.

Future Earth aims to coordinate new, interdisciplinary approaches to research on three themes: Dynamic Planet, Global Sustainable Development and Transformation towards Sustainability. The goal of this initiative is also to provide a global platform to deliver ([75]):

- **Solution oriented** research for sustainability, satisfy human needs for food, water, energy, and health through correlation between environmental change and development challenges;
- **Effective interdisciplinary collaboration** – between and across natural and social sciences, humanities, economics, and technology development, to maximize the scientific output;
- **Timely information for policy makers –** through knowledge generation that will support existing and new global and regional integrated assessment;
- **Participation** – of different stakeholders – policy makers, funders, academics, business and industry – to co-design and co-produce research agendas and knowledge;
- **Increased capacity building** – especially in developing countries, in science, technology and innovation.

Members of the Governing Council of Future Earth include:

- International Council for Science (ICSU);
- International Social Science Council (ISSC);

- Belmont Forum of funding agencies;
- United Nations Educational, Scientific, and Cultural Organization (UNESCO);
- United Nations Environmental Program (UNEP);
- United Nations University (UNU);
- World Meteorological Organization (WMO).

3.7. Personal Contributions

- Heuristic approach on the technical details of SDI components:
    - Standards implementations;
    - Testing and working with different GIS and OGC services: WMS, WFS, WCS, WPS (PyWPS), TJS.
- Published Papers:
    - ***Rodila, D.,*** *and Gorgan, D. (2012). Geospatial and Grid Interoperability through OGC Services Gridification, in International Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS), Vol. 5/6, December 2012, pp. 1650 – 1658, ISSN: 1939-1404, http://dx.doi.org/10.1109/JSTARS.2012.2217115.*
    - *Mihon, D., Colceriu, V., Bacu, V., Allenbach, K.,* ***Rodila, D.,*** *Giuliani, G., and Gorgan, D. (2013). OGC Compliant Services for Remote Sensing Processing over the Grid Infrastructure, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 32–40, ISSN 2158-107X.*
    - *Gorgan, D., Bacu, V., Mihon, D., Stefanut, T.,* ***Rodila, D.,*** *Cau, P., Abbaspour, K., Giuliani, G., Ray, N., and Lehmann, A. (2012). Software platform interoperability throughout enviroGRIDS portal, in International Journal of Selected Topics in Applied Earth Observations and Remote Sensing - JSTARS, Vol. PP/99, pp. 1-11.*
    - ***Rodila, D.,*** *and Gorgan, D. (2011). A Mediation Approach in Geospatial Web Services Gridification, in ICCP2011 – IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 25-27, 2011, pp. 541-548, http://dx.doi.org/10.1109/ICCP.2011.6047928.*
    - ***Rodila, D.,*** *Bacu, V., Ardelean, V., Borlea, C., and Gorgan, D. (2011). Geospatial Web Services Gridification in enviroGRIDS", in European Geosciences Union -*

General Assembly EGU 2011, Vienna, Austria, April 03-08, 2011, (abstract and presentation), http://meetingorganizer.copernicus.org/EGU2011/EGU2011-11469.pdf.

- *Gorgan, D., Bacu, V., Manca, S., Giuliani, G., **Rodila, D.,** Stefanut, T, Mihon, D., Abbaspour, K., Rouholahnejad, E., van Griensven, A., Kokoszkiewicz, L., and Cau, P. (2011). Tools and Applications in enviroGRIDS Project for Spatial Data Processing of Black Sea Catchment Basin, EGI User Forum, Vilnius, 11-15 April, https://www.egi.eu/indico/contributionDisplay.py?contribId=154&sessionId=16&confId=207.*
- ***Rodila, D.,** and Gorgan, D. (2010). Integration of Spatial Data Infrastructures with Grid Environment, in SYNASC 2010 – 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, September 23-26, 2010, pp. 269-277, http://dx.doi.org/10.1109/SYNASC.2010.63.*
- ***Rodila, D.,** Gorgan, D., and Bacu, V. (2010). The Interoperability between OGC Services and Grid Environment in EnviroGRIDS Project, in 3PGCIC 2010 – International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Fukuoka, Japan, November 4-6, 2010, IEEE Computer Press, pp. 387-392,ISBN: 978-0-7695-4237-9, http://dx.doi.org/10.1109/3PGCIC.2010.65.*
- *Gorgan, D., **Rodila, D.,** Bacu, V., Giuliani, G., and Ray, N. (2010). OGC and Grid Interoperability in enviroGRIDS Project, in European Geosciences Union - General Assembly EGU 2010, May 02-07, 2010, Vienna, Austria (abstract and presentation), http://meetingorganizer.copernicus.org/EGU2010/EGU2010-13457.pdf.*
- *Gorgan, D., **Rodila, D.,** Bacu, V., Giuliani, G., Ray, N., Charvat, K., and Lehman, A. (2010). Geospatial and Grid infrastructures interoperability in enviroGRIDS, 5th EGEE User Forum, April 12-16, 2010, Uppsala, Sweden (2010). Book of Abstracts, http://indico.cern.ch/contributionDisplay.py?contribId=161&confId=69338.*

Chapter 4: Environmental Applications

4.1. Introduction

Environmental Sciences is a multidisciplinary field that integrates physical, biological and information sciences to study together the systems, the problems and the solutions of the environment. In the beginning of Environmental Sciences, in the 1960s, the scientific community was more focused on disciplines, trying to develop knowledge in particular fields (such as geology, ecosystems, hydrology, etc.) but in the 1980s it became more and more obvious that these disciplines are strongly connected and the scientific community started to study them as interacting elements in a single big system (Dozier and Gail, 2009 [58]). After this shift, it was easier to understand complex, system-oriented phenomena that link concepts from different fields (climate change involves atmospheric science, biology, human behavior, etc.) but also to understand and make a better use of the collected data (such as these coming from satellite observations). The growing understanding of these complex processes lead also to the development of new models. The knowledge gathered mainly for scientific understanding, begins to be used more to support practical decisions and actions, redirecting the Environmental Sciences to environmental applications. The role between basic science and applications is emphasized by the societal needs. After collecting and analyzing the gathered information, the community needs also a more fundamental, process-based understanding of the phenomena – a science of environmental applications. This science is guided more by societal needs than by scientific curiosity, focusing more on specific actions as well as on their consequences (Dozier and Gail, 2009 [58]).

Applications that are used to solve different environmental issues use specific data as input and produce outputs that are useful for the Earth and environmental community at large can be labeled as "environmental sciences applications" (or simply "environmental applications" hereafter). Since the 1990s, the number and diversity of environmental applications have increase dramatically. Many software systems were developed to integrate data coming from various thematic areas such as agriculture and soil science, ecology, terrain modeling, hydrology, land use/land cover, population distribution, education and health planning, energy resources, etc. The specificity of the majority of these environmental applications is the requirement of

large computational and storage resources due to the massive amount of input and/or output data that comes typically from a combination of high spatial and temporal resolutions. Other reasons for high performance requirements include also the utilization of compute-intensive algorithms, the execution of large number of scenarios, the urgent need of responses, etc. Different parallel and distributed infrastructures, such as Grids, Clouds, and High Performance Computing (HPC) systems can satisfy the necessary requirements for running these applications (Nativi et al., 2013 [159]). Some examples of environmental applications taking advantage of the capabilities offered by parallel and distributed infrastructures are those using parameter estimation, model calibration (Vrugt et al., 2006 [232], gSWAT ([99], Bacu et al., 2011b [17]), Web Processing Service on the Grid (e.g., Giuliani et al., 2012 [87]) and numerical weather prediction (Maity et al., 2013 [145]), satellite images workflows over the Grid (GreenLand [96]).

4.2. Hydrological Models

The hydrological cycle has a great significance in different environmental systems. It involves basic sciences such as physics, chemistry and biology and it connects geology, ecology, atmosphere and society (Savenije, 2009 [207]). Studying the hydrological cycle, both regarding its present state as well its possible future changes, it's therefore an important task that was strongly influenced by spatially distributed modeling. Flood and drought forecasting, water management, prediction of the impact of natural and human induced changes in hydrological cycle are just a few examples in which distributed hydrological models can be very useful. A schematic representation of the hydrological cycle is presented in (SWAT, 2009 [221]) and illustrated in Figure 15.

Hydrological models are therefore important tools for simulating the behavior of catchments in space and time, providing important information not only to scientists but also to decision makers (Viviroli et al., 2009 [231]). As many other environmental applications, these models have to simulate a large variety of physical processes which leads not only to a high complexity but also to a high degree of parameterization (Silvestro et al., 2013 [213]).

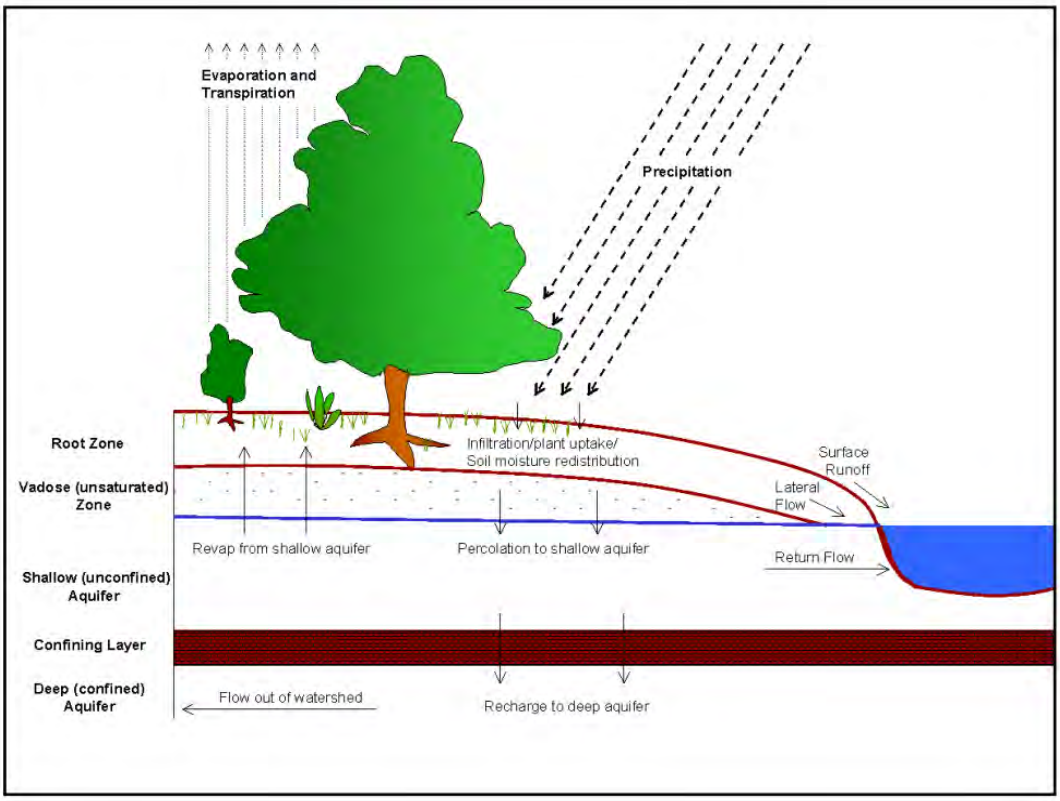

Figure 15: Schematic Representation of the Hydrologic Cycle (SWAT, 2009)

Hydrological models are data driven, working with different types of complex data that can be retrieved in different manners, from different sources, with different policies and coming under different format standards. Collecting all the various types of quality data, needed for a hydrological model, can be a challenging task and can limit the model development. On top of this, the availability of software tools needed in the development process is another challenge that has to be taken under consideration by a specialist in hydrology. With all these challenges, the hydrological models have evolved a lot, on one side because of the exponential development of the computation capacity and on the other side because of the progress of Earth Observation techniques, which made huge amounts of data readily available. All these factors helped the scientific community to have a better understanding and representation of the studied phenomena.

4.3. Remote Sensing Applications

Remote Sensing (RS) is a technique of collecting data about an object or phenomenon from distance, without making physical contact with the object. Remote sensing is not limited to

digital, satellite based sensor systems but includes also the photogrammetric field and hence conventional analogue sensors, including aerial photography. A more detailed definition of remote sensing is given by Jensen (2000) [116]: "non-contact recording of information from the ultraviolet, visible, infra-red, and microwave regions of the electromagnetic spectrum by means of instruments such as cameras, scanners, lasers, linear arrays, and/or area arrays located on platforms such as aircraft or spacecraft, and the analysis of acquired information by means of visual and digital image processing".

Based on the definition, the field of remote sensing is associated with large amounts of visual data, coming from a broad range of sensors, which vary in their spectral, spatial, and temporal characteristics (Basaeed et al., 2012 [21]). Beside traditional sources and collection methods of data, satellite remote sensing remains one of the largest sources of data collection (Cavallaroa et al., 2014 [39]).

More than 1000 manmade satellites currently orbit our planet (UCS Satellite Database - http://goo.gl/wSaVJI), some near the edge of the Earth's atmosphere and others tens of thousands of kilometers above us. A small number of these satellites play a critical and quickly expanding role in monitoring the Earth's surface and atmosphere to track environmental conditions, and a number of new Earth-Observing missions are planned for the next decade (including Sentinel 5) (Seltenrich, 2014 [210]). These new satellites will offer higher resolution imagery, requiring more robust and precise algorithms to process the data they deliver. Taking advantage of satellite and airborne sensors to observe, measure, and record the radiation reflected or emitted by the Earth and its environment, remote sensing can significantly enhanced the information available from traditional data sources.

Medium to high-resolution multi-spectral images are available not only for governmental agencies but also for research and academic centers, industries and citizens, from open data initiatives, every week and in some regions even twice per week, assuring a data continuity strategy (Karmas et al., 2015 [118]). This data deluge is also strongly influenced by the ever-increasing technological progress and technological capabilities of the Web. Data is growing at an unprecedented rate, becoming not only voluminous but also increasingly complex, and accessible in a variety of formats. This brings new important opportunities (such as the monitoring of Earth surface changes with greater frequency then ever) but also new challenges especially regarding the access, analysis, processing, and sharing of these data. Even though EO

data have become freely available, due to the incredible sizes (e.g., petabytes of data), once data are stored in archives, they become hard to transfer, mostly because of bandwidth limitations (Nikolaou et al., 2014 [165]). If users manage to solve this problem, massive computing resources are still needed to store and then process this data (Karmas et al., 2015 [118]). Harnessing the full potential of these EO datasets and getting meaningful information requires not only massive computing resources, but also specialized algorithms and dedicated tools, which have to be brought to data instead of moving the data to the processing centers (Karmas et al., 2015 [118], Evangelidis et al., 2014 [68]).

Extraction of information from satellite – remote sensing data needs techniques and processes which are time consuming: image downloading, mosaic, re-projection, extraction of information of interest, classification, software based enhancement, etc. Tumwizere and Jeong, (2012) [226] describe the main steps in a remote sensing system data flow: 1) data source, 2) data collection, 3) geometric correction, 4) projection/mosaic, 5) information extraction, 6) data analysis, and 7) end user utilization.

Remote sensing enables large-scale observations of areas that sometimes can be inaccessible or difficult to access using conventional methods of data retrieval. A satellite can capture images of an area in minutes while the data collected in the field could take much more time, even years. Remote sensing has been widely used in resource inventory, land use change monitoring, urban planning, monitoring of urban environment regarding air and water pollution, green spaces, traffic analysis, population estimation, agricultural analysis, environmental monitoring, and many other applications (Seltenrich, 2014 [210]). Through the development of remote sensing technology, such as increasing of spatial land spectral resolution, 3D laser scanning, data mining, advanced image processing technology, remote sensing is expected to be used more and more to support a sustainable environment.

4.3.1. Landsat Mission

We have focused our research mainly on working with Landsat satellite imagery, and recently especially on Landsat 8 data. The Landsat archive is probably one of the most authoritative repositories of freely available remotely sensed data, mostly because of the impressive historical archive and the ease of access. Data from Landsat satellites provides the longest continuous record as seen from space of the Earth's surface, starting with the launch of

Landsat 1 in 1972 through Landsat 8 in March 2013 (Santos et al., 2014 [206], Landsat News, 2014 [129]). Landsat 9 is planned to be launched in 2023, maintaining this incredible continuous stream of data. Landsat is one of the most popular Earth Observations programs for observing the Earth surface and it's used in various areas such as monitoring, detecting, and classifying land surface changes.

Landsat data has a long-term historical record of the entire globe, offering high quality data and becoming a vital reference in almost all areas by offering a global image archive with an unmatched value (Wulder et al., 2012 [244]). As Marcia McNutt, then-USGS Director, said: "Landsat is valued all over the world as the 'gold standard' of land observation" (Landsat Science, 2015) and will undoubtedly lead to incredible future insights on the Earth System. In 2008 all new and archived Landsat data held by the USGS have been made freely available over the Internet following a tremendous increase in scientific investigations and applications using Landsat data (Wulder et al., 2012 [244]). Before the free and access policy, a daily average of 52 scenes of Landsat data were distributed but since 2008, this number has dramatically increased to reach a value of 5'700 scenes (Ryan, 2016 [204]). In May 2015, Amazon Web Services (AWS) announced that it would host Landsat 8 imagery on its publicly accessible Simple Storage Service (S3). At that moment, AWS has made available over 80,000 Landsat 8 scenes (approx. 85 Tb of data) and hundreds of other scenes are being added daily since then (Landsat Science, 2015). The free and open access policy of Landsat data was a brilliant example on how to maximize the return on the large investments in satellite missions (Wulder et al., 2012 [244]). The immense benefits of similar Open Data initiatives was emphasized also by Barbara Ryan, the Secretariat Director, Group on Earth Observations (GEO) by pointing out the economic benefits brought back by these initiatives: "The economic value of geospatial data lies in its utility" (Ryan, 2016 [204]). According to GEO, more than 12 million Landsat images have been delivered across 186 countries enabling users to access multiple-year scenes for the same locations (EarthZine, 2014 [59]).

Landsat 8 satellite images the entire Earth every 16 days in an 8-day offset from Landsat 7. Landsat 8 images consist of nine spectral bands, with a 30 m spatial resolution for Bands 1 to 7, and 9 and 15 m resolution for Band 8 (panchromatic). It includes two new spectral bands: a deep blue band (Band 1) designed for coastal/aerosol studies, and a shortwave infrared band (Band 9) for cirrus (thin type of clouds that forms high in the sky) detection. The thermal bands

10 and 11 are useful for providing more accurate surface temperatures and are collected at 100 m resolution (http://landsat.usgs.gov/).

Landsat 8 is acquiring around 550 images/day (approx. 60% more scenes per day compared to Landsat 7 (Roy et al., 2014 [202]), having the ability to image more frequently in persistently cloudy areas to improve data collections in areas of critical importance for climate studies (Landsat News, 2014 [129], Roy et al., 2014 [202]). It has improved high capacity on-board recording and satellite to ground transmission capabilities, compared to previous Landsat systems. With all this, the strength of the Landsat is not necessarily the high resolution (as there are other instruments that provide much higher resolution than Landsat) but the impressive historical archive and the ease of access.

4.3.2. Landsat Applications

Based on recent studies (Santos and Gonçalves, 2014 [206], Landsat News, 2014 [129]) the target applications of Landsat 8 focus on natural processes such as volcanic eruptions, glacial retreat, floods, forest fires, and other natural disaster impacts, and on human induced processes such as urban expansion, crop irrigation, and forest clear-cutting. Even though Landsat in general is more oriented to land than sea, Landsat 8 provides an impressive picture of band combinations of coastal zones through the new Band 1 (coastal/aerosol band), which allows a closer investigation of coastal waters. Pan-sharpening continues to be available with Landsat 8 and allows the creation of a single high-resolution color image (useful to improve classification accuracy) by merging high-resolution panchromatic band of 15 m and lower resolution multispectral imagery of 30 m (Santos and Gonçalves, 2014 [206]). Landsat 8 is also used in agriculture for local and global decision making: monitoring crops, forecasting crop production, monitoring droughts and water use, computing vegetation indexes (SAVI, NDVI), etc. Possible problems could be detected and explored with more detailed resolution images (Santos and Gonçalves, 2014 [206]). Change detection techniques can also be applied using Landsat 8, such as for coastline changes but also for rapid detections such as clear-cuts and burnt areas. Santos and Gonçalves, 2014 [206] also made a preliminary analysis on remote sensing services to quickly produce land use and land cover (LULC) maps from Landsat 8 data. The conclusion is that rapid land cover mapping is feasible using automated processes and pre-defined training data but some adaption effort is needed to support the new features of Landsat 8 (16 bit images).

According to USGS, there is an expansive range of customers (from academics to foresters and urban planners to agricultural managers) that use Landsat satellite imagery. This data has quickly found its way into a wide range of applications, which involve scientific discovery, managing and monitoring resources in different domains such as: economic and environmental quality, national security, public health and human well-being etc. (Roy et al., 2014 [202]).

The benefits of Landsat data can be observed in different areas, including (Roy et al., 2014 [202]):

- Water resources analysis and management;
- Agriculture and forest analysis and management;
- Homeland security;
- Infrastructure analysis;
- Disaster management;
- Climate change science;
- Wetland protection;
- Monitoring land cover changes.

4.4. Environmental Challenges Overview

There are different types of challenges that limit our capacity to understand the Earth system, its components and their interactions. Among these challenges we have identified:

- Environmental data is growing at an unprecedented rate, in volume, complexity, variety;
- Data scarcity;
- Data standards application;
- Once data are stored in archives, they become hard to transfer, mostly because of bandwidth limitations;
- Increasing need of services to access, store, analyze, process and share big environmental data in a standardized way;
- Software requirements and availability to extract meaningful information out of raw data;
- Computational and storage capacities.

All of these challenges can be seen at different levels, depending on the scale of the context. The envisaged solutions are the provision of massive computing and storage resources but also specialized algorithms and dedicated tools. A methodology and framework able to efficiently and automatically acquisition and process data is needed therefore.

4.5. Conclusions

The increase global annual Landsat data volume requires high performance supercomputers, capable of providing petabyte data storage and processing solutions. Recent technological developments have the potential to provide researchers and policy makers with an *"unprecedented capacity to access, analyze, and integrate higher level products derived from the multi-petabyte scale archive of global Landsat data" (*Roy et al., 2014 [202]*).*

Consequently, data acquisition services are essential, especially for satellite images and applications in the area of remote sensing. There are unprecedented opportunities for efficiently and automatically producing near-real time Landsat 8 monitoring products but the first step for all this is the automatic data acquisition. Any remote sensing application has to perform this step before being able to perform any geo-processing concerning data analysis and/or data processing tasks. This task usually involves several steps such as data discovery, validation and downloads.

The choice of the appropriate parallel or distributed infrastructure depends on the application features, data model, and processing requirements of the environmental application. To run on one or several of these distributed or parallel infrastructures (i.e., a heterogeneous computational environment), the application has to be modified to have a particular structure or to use particular programming interfaces for accessing the resources of the infrastructures. This is typically done without knowing too much details about the final infrastructure(s) on which the application will run.

4.6. Personal Contributions

- Identification of major environmental challenges related to environmental applications, from an engineering point of view;
- Identification of general characteristics of environmental applications, mainly from hydrological and remote sensing fields.

Chapter 5: Distributed Systems

5.1. Introduction

According to (Tanenbaum and Steen, 2006 [223]), a distributed system is a collection of independent computers that are presented to the users as a single compute resource and provides a single system view. A more complete definition of a distributed system is given by (Baker et al, 2002 [20]) as: "*A type of parallel and distributed system that enables the sharing, selection and aggregation of geographically distributed autonomous and heterogeneous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality of service requirements*". The development of distributed systems was strongly influenced by the development of computer networks in the late 1970s and early 1980s (Andrews, 1999 [11]) and as the use of high-speed broadband network increases, the field of computing is constantly changing as well.

Up to nowadays there are a few technologies that have emerged in this area and in the following we will present some of the most important ones. An adapted taxonomy of the current technologies is also presented in Figure 16.

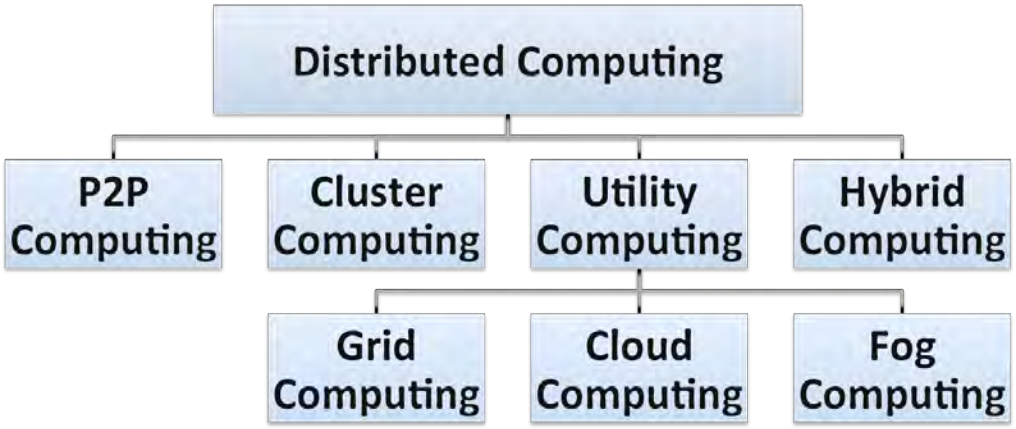

Figure 16: Taxonomy of Distributed Computing.

Before starting the presentation in detail of some of the most important distributed systems, we consider it is important to also shortly mention the Multicore systems.

A Multicore is a processing system with two or more independent cores integrated on the same chip. Multicore architectures have evolved because it was difficult to make single-core clock frequencies even higher. These architectures explicitly exploit thread level parallelism, i.e. server can serve each client in a separate thread. Multicore processors are Multiple Instructions Multiple Data (MIMD), meaning that different cores execute different threads, operating on different parts of memory. Most of the time Multicore is also a shared memory multiprocessor i.e. all cores share the same memory but they can also have distributed or mixed memory architectures. The additional performance offered by this platform is exploited only by parallelized applications. The execution of other types of applications can even be slow down.
Older computers have just one CPU, but newer computers have multi-core processor chips and many CPUs. With single-CPU, single-core computers, it is also possible to perform parallel processing by connecting the computers in a network. However, this type of parallel processing requires very sophisticated software called distributed processing software. The main advantages of the system is that: it does not have the disadvantages of the Grid and Cloud, it has all the advantages of using a PC with different operating systems, the user is in full control of the job being processed, it can be stopped and restarted at any time, it is much simpler to use, not needing Grid or Cloud certificate, permission, etc., and depending on the PC, the processes can go faster up to the number of available CPUs. There are actually not many disadvantages in PPS, except that we need a powerful computer to take the full advantages of the system. With the advancement of new technologies, this is now available at a reasonable cost (Rouholahnejad et al., 2011 [200]).

A major challenge in Multicore processing is the software development. The performance speed up depends on how good is the multi-threading of the parallel source code. The major characteristic of a good parallel code should be correctness, efficiency, scalability and portability.

We also consider that it is necessary to emphasize here the difference between Distributed and Parallel. Distributed Computing it's a type of Parallel Computing although the latest mostly refers to processing in which different parts of a program run simultaneously on two or more processors that are part of the same computer. Both types require that the program is

divided is section that can run simultaneously but Distributed Computing also requires that these sections run in different environments (and not on the same machine) (Kaur, 2015 [122]).

Although Peer to Peer (P2P) networks are among the first distributed systems while the most important class of distributed systems contains the distributed computing systems that are used for high performance computing tasks (Tanenbaum and Steen, 2006 [223]): cluster, Grid (mid-1990s) and Cloud (2007). All these systems belong to the Utility Computing class, which is based on a service-provisioning model where users (consumers) pay providers for using computing power only when they need to. In other words, Utility Computing aims to show how computing needs of users can be fulfilled by IT industry and is based on the model used by conventional utilities (telephone, electricity, gas, etc.) (Kahanwal and Singh, 2012 [117]). A detailed comparison of all these distributed systems or a group of them and their specific functionalities are discussed in literature (Foster et al., 2008 [73], Sadashiv and Kumar, 2011 [205], Mateescu et al. 2011 [146], Rings et al., 2011 [191], Rings and Grabowski, 2012 [192], Hajibaba and Gorgin, 2014 [103]).

The trend in distributed systems is changing fast due to the ever-increasing needs and technological progresses and new computing paradigms start to develop such as Hybrid Computing (also known as Jungle computing) (2010 - 2011) and Fog Computing (2012).

5.2. Peer to Peer (P2P)

Peer-to-Peer (P2P) network is one of the first distributed systems (Hajibaba and Gorgin, 2014 [103]). In this type of distributed systems every node acts both as a client and a server, providing part of the system resources and no peer machine has a global view of the entire P2P system. Peer machines are simply clients connected to the Internet, acting autonomously to join or leave the system. There is no master-slave relationship among the nodes and no central coordination or central database (Kahanwal and Singh, 2012 [117]). An architecture of such a system is shown in Figure 17.

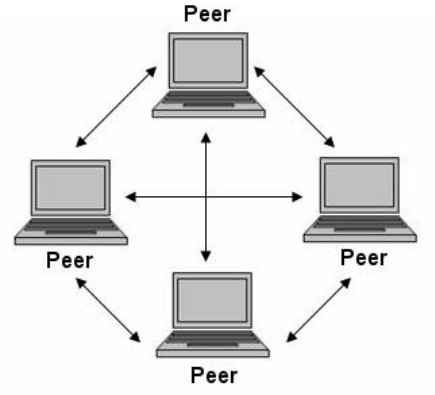

Figure 17: Peer to Peer (P2P) Network

5.3. Cluster

5.3.1. General Description

A cluster can be seen as a set of loosely connected or tightly connected nodes, which communicate with each other and work together to achieve a common goal. The components of a cluster are usually connected to each other through a fast local area network (LAN), each node running its own instance of an operating system. In these systems, nodes are frequently homogenous in both hardware and operating system. The computing nodes are orchestrated usually by a middleware – a software layer that sits atop the nodes, which gives the users the impression of a large cohesive single computing unit instead of several working nodes (a single system image concept). Computer clusters emerged to improve performance and availability of a single computer and they have become popular also due to the availability of low cost microprocessors, high-speed networks and a large set of software for high performance distributed computing. A computer cluster may be a very fast supercomputer or it may be just a simple 2-node system, which connects 2 personal computers together. Traditional owner centric clusters are usually excellent in handling capability workloads with large datasets in a well-managed, secure environment. Capacity is fixed and there is usually no support for virtualization and resource sharing (Mateescu et al., 2011 [146]).

The major advantages of a computing cluster are:

- **Cost efficiency** – the cluster is cost efficient for the amount of power and processing speed it can produce.

- **Processing speed** – multiple high-speed computers work together to provide a unified faster processing.
- **Improved network infrastructure** – different LAN topologies used within a cluster create a highly efficient and effective infrastructure.
- **Flexibility/elasticity** – you can easily enhance the existing specifications or add extra components to the system to meet the size and time requirements of your workload.
- **High availability of resources** – the failure of a machine in a cluster does not affect the processing of the other machines.
- **Run jobs anytime anywhere** – the submission of jobs is easily done using simple APIs or management tools.

5.3.2. Cluster Types

Clusters can be of different types and a classification of these has been made in previous researches (Kahanwal and Singh, 2012 [117], Kaur, 2015 [122]):

- **High Availability Clusters** (also known as Failover Clusters): groups of computers that support server applications and can be reliably utilized with a minimum downtime. These clusters immediately detect hardware/software faults and restart the application on another system (they harness redundant computers in groups or cluster that provide continue service) and this is called failover.
- **Load Balancing Clusters**: required by applications with large volumes of client requests or having high demands on security and redundancy.
    - **Software-based**: involves installing special software on the servers, which dispatches or accepts requests from the client to the servers, based on different algorithms (simple round-robin for example or more complicated ones). Examples: Microsoft Network Load Balancing for web farms and Microsoft Component Load Balancing for application farms.
    - **Hardware-based**: consists of a specialized switch or router with software to give it load-balancing functionality. By having both functionalities into a single device, reduces the extra hardware but it is also more complex to program and to troubleshoot.

- **High Performance Computing Clusters**: used to increase computing throughput, used most of the time for complex applications with strict time constrains (example: run different jobs with different parameters or data sets). The cluster manages the resources needed for each job and handles the jobs execution.

5.3.3. Architecture

The architecture of a cluster computing environment, as presented by (Kahanwal and Singh, 2012 [117]) is shown in Figure 18.

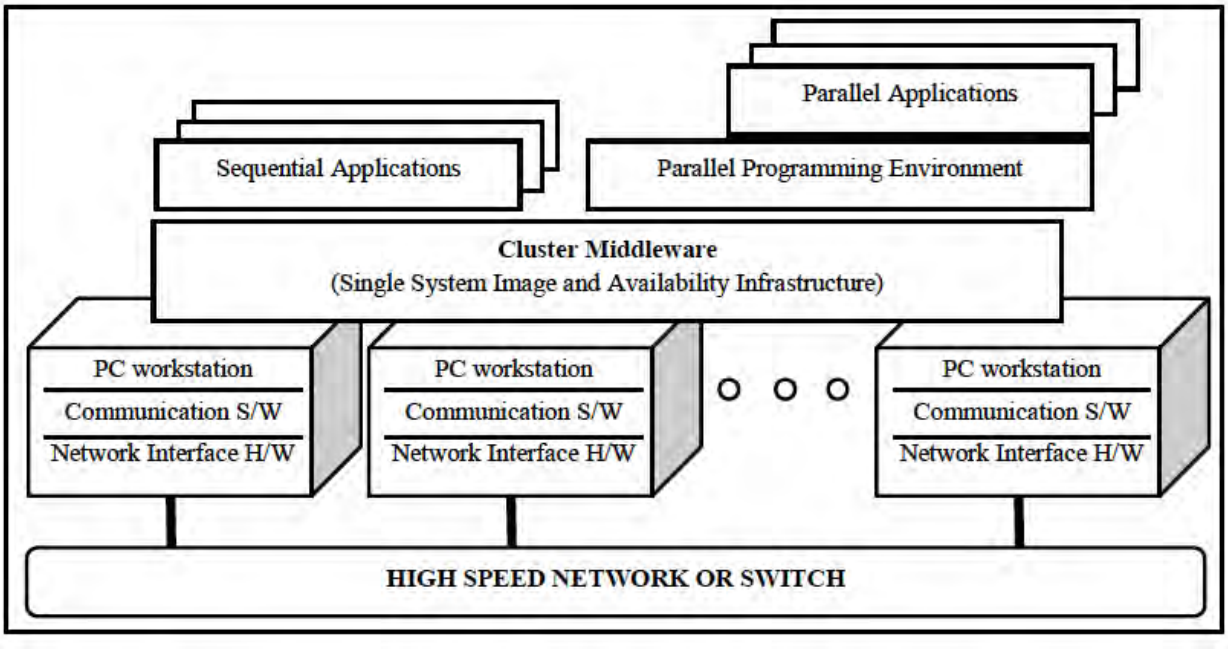

Figure 18: Cluster Architecture (Kahanwal and Singh, 2012)

The components of a cluster system include (Kahanwal and Singh, 2012 [117]):

- High Performance Computers;
- Micro-kernel based operating systems;
- High speed networks or switches;
- NICs (Network Interface Cards);
- Fast Communications Protocols and Services;
- Cluster Middleware (hardware, Operating System kernels, applications and subsystems);
- Parallel Programming Environment tools (compilers, parallel virtual machines, etc.);
- Sequential and Parallel applications.

5.3.4. SLURM

Slurm (http://slurm.schedmd.com/) is an open source workload designed for Linux clusters of all sizes and it provides three important functions:

- Allocates exclusive and/or non-exclusive access to resources for users for a certain amount of time;
- Provides a framework for starting, executing, and monitoring parallel jobs on a set of allocated resources (computing nodes);
- Arbitrates contention of resources by managing a queue of pending jobs.

Besides these there important functions, Slurm also provides a set of important properties, such as:

- **Scalability** – can operate in a heterogeneous cluster (up to tens of millions of processors);
- **Performance** – can accept 1000 job submissions per second and can fully execute 500 simple jobs per second;
- **Free and Open Source** – the source code is freely available under GNU General Public License;
- **Portability** – initially was written for Linux but after it was ported to other systems;
- **Power Management** – the jobs can specify their desired CPU frequency, the power use by job is recorded and idle resources can be powered down until needed;
- **Fault Tolerant –** highly tolerant of system failures, including node failures executing control functions;
- **Flexibility** – allows various interconnects, authentication mechanisms, schedulers, etc. through a plugin mechanism;
- **Resizable jobs** – jobs can grow and shrink on demand; time and size limit ranges can be specified in the job submission;
- **Status jobs** – status-running jobs exists to help identify load imbalances and other anomalies.

5.4. Grid

5.4.1. General Description

A **Grid** is a distributed paradigm that enables remote sharing, selection and aggregation of geographically autonomous resources dynamically at runtime, within a Virtual Organization (VO), based on different criteria: availability, capability, performance, quality of service, etc. Members of a VO can work together in projects, even across enterprises boundaries without the need to belong to the same affiliation. A VO defines what computational and data resources are shared but also who is allowed to access this resources (Hoeing, 2010 [107]). As opposed to the cluster systems, Grid systems frequently involve distributed management by multiple administrative entities (such as VO) (Bonomi et al., 2012 [30]).

Grid infrastructures offer seamless, secure and on-demand access to geographically distributed computing, communication and information resources (Costan, 2010 [48]). The term Grid actually originates from the idea of making access to computational power as easily as to the electrical power grid. Grid systems are used for the execution of large tasks that require high computing power and/or large data storages. It is mainly designed to solve problems that are too big for a supercomputer while being flexible to solve smaller problems as well (Kaur, 2015 [122]). With all this, Grid computing has not been accepted as default technology in the business domain, mainly due to missing features like quality of service agreements and the extreme focus on high performance compute jobs instead of general information system provisioning (Hoeing, 2010 [107]). This is the point where Cloud computing entered in the scene as a new paradigm.

A Grid has a variety of resources based on diverse hardware and software structures, computer languages and frameworks but enables exposing these heterogeneous resources with a unified interface, allowing users to access multiple resources in a unified manner. The amount of resources is still constant within a non-virtualized Grid and there is a limited interoperability between different Grid software stacks.

The Grid middleware offers services to allocate compute resources, to access data resources, to monitor executing jobs, to provide security, etc. All these functionalities are offered in modern Grid middlewares via SOAP-based services (Hoeing, 2010 [107]). The security in such environments is normally assured using personal certificates, such a X.509 certificates, issued by a trusted institution. These certificates allow the communication with Grid services and

also facilitate the traceability of Grid activities of each user. A user can also delegate right to another Grid component, facilitating thus the execution of operations in a workflow.

Infrastructures such as TeraGrid (Katz et al., 2010 [121]), EGEE (Enabling Grids for e-Science) (Ferrari and Gaido, 2011 [70]) and DEISA (Distributed European Infrastructure for Supercomputing Applications) (Gentzsch et al., 2011 [79]) integrate high-end computing and storage systems via high-speed interconnects, and support traditional, batch-queue-based computationally/data intensive high-performance applications.

Depending on their usage, Grid systems can be classified as follows (Figure 19):

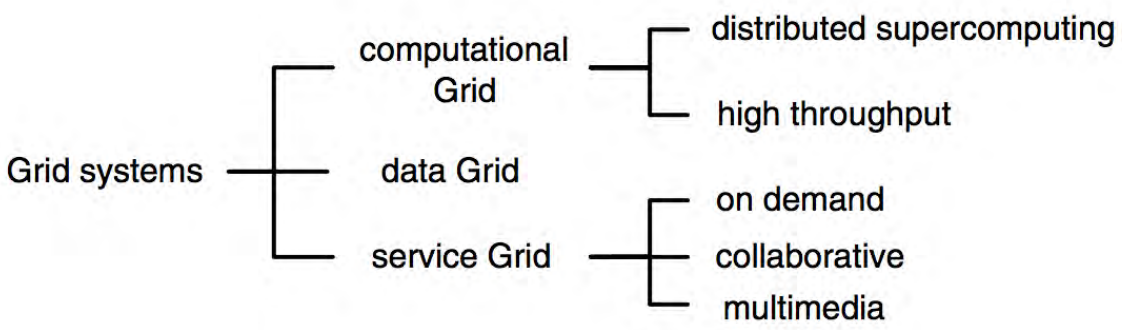

Figure 19: Grid Systems Taxonomy - Krauter et al., 2002

- **Computational Grid** (a system with a higher aggregate capacity than any of its constituent machine):
  - *Distributed Supercomputing* (executes the applications in parallel, on multiple machines, to reduce the completion time of a job);
  - *High Throughput* (increases the completion rate of a stream of jobs arriving in real time).
- **Data Grid** (provides an infrastructure for synthesizing new information from data repositories – digital libraries or data warehouses. Special purpose data mining applications, which correlate information from multiple different high volume data sources, can be use cases for these systems).
- **Service Grid** (provide services that are not provided by any single machine):
  - *On Demand* (dynamically aggregates different resources to provide new services. Ex: dynamically increase the fidelity of a simulation by allocating more machines);

- *Collaborative* (connects users and applications into collaborative workgroups and enable real time interaction between humans and applications through a virtual workspace);
- *Multimedia* (provides infrastructure for real time multimedia applications – required the support of QoS across multiple machines while the deployment on a single machine of a multimedia application can be done without QoS).

The resource management in a Grid system involves managing the basic Grid elements. A Grid computing system must contain a Computing Element (CE), a number of Storage Elements (SE) and Worker Nodes (WN). A CE provides the connection with other Grid networks and dispatches the jobs on the WN through a Workload Management System (WMS). The storage of input and output data needed for job executions is done in the Storage Element while the WNs are the servers that offer the processing power (Garlasu et al., 2013 [76]). To be able to access the Grid resources, a user needs certificates issued by the VO. The administrative part of a VO comprises a Workload Management System (WMS), which keeps track of all the available CEs, a Virtual Organization Membership System (VOMS) and the Logical File Catalog (LFC).

5.4.2. Architecture

A Grid architecture, as defined in literature (Rings and Grabowski, 2012 [192]) is presented in Figure 20. Local resources are entities of different types that fulfil job requests and are usually deployed on a private network. These resources include computing, storage, sensor, and services (Krauter et al., 2002 [126], Rings and Grabowski, 2012 [192]). The Grid core services use specific protocol and interfaces to access the local resources from a public network and they include services for information, data, execution, and resource management. These Grid services are used by Grid schedulers to schedule jobs over several Grid infrastructures, or by Grid portals and Grid applications (Rings and Grabowski, 2012 [192]).

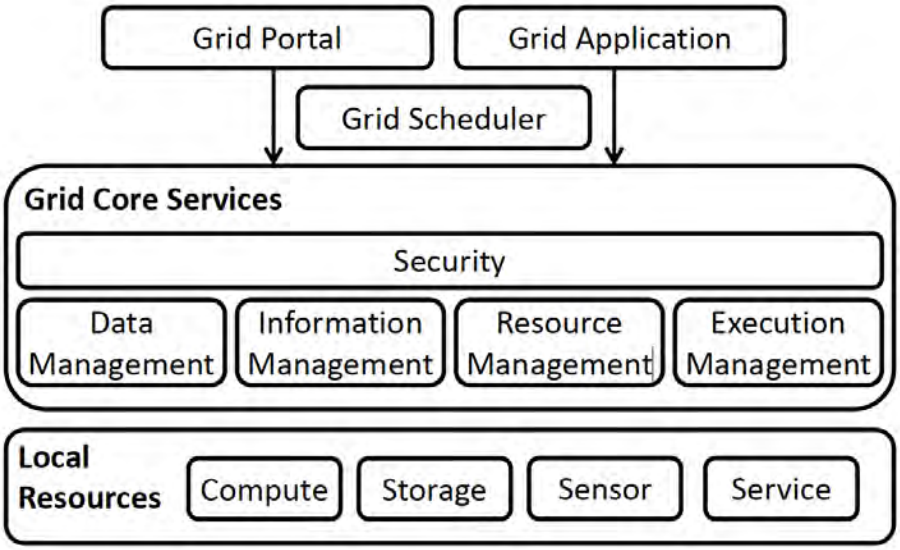

Figure 20: Conceptual Model of Grid Computing (Rings and Grabowski, 2012)

5.4.3. gLite Middleware

A Grid middleware provides secure access to diverse resources that are managed in a centralized manner (Rings and Grabowski, 2012 [192]). The gLite ([93]) middleware was produced by the EGEE (Enabling Grid for E-SciencE) (EGEE [60], Ferrari and Gaido, 2011 [70]) project and subsequently developed by the EMI (European Middleware Initiative) project (EMI [61]). This middleware computer software is an integrated set of components designed to enable resource sharing and it was used by LHC (Large Hadron Collider) (LHC [135]) CERN experiment as well as other scientific domains. The gLite middleware offers just a command line interface therefore is quite hard for non-specialists to use it. GANGA (Moscicki et al. 2009 [154], Harrison et al. 2013 [104]) and DIANE (Distributed Analysis Environment) (Moscicki, 2003 [153]) are two software tools that can facilitate the accessibility of users to the Grid infrastructure through flexible programming interfaces as well as graphical user interfaces. DIANE is used in conjunction with GANGA to provide a more efficient usage of the Grid resources. Both of these tools will be described in detail in the following sections.

5.4.4. Scheduling, Execution and Monitoring Tools

5.4.4.1. GANGA

GANGA (Moscicki et al. 2009 [154], Harrison et al. 2013 [104]) is a Python front-end tool used for defining and managing Grid jobs and it was initially designed as an interface for Grid for two of the seven particle physics detector experiments at CERN – ATLAS and LHCb. A job defined with GANGA is usually composed of several blocks such as: application, backend, input dataset, output dataset, splitter and merger. The application refers to the software component that has to be executed while the backend specifies the used computing system. The input data set is the application input and the output dataset represent the generated data. The split and merge component are optional and can be used to divide the job in parallel sub-jobs and to combine their results.

The main scope of Ganga was to facilitate different users to create, execute and monitor processes in the Grid infrastructure. It allows the motorization of processes even in independent sessions so the user can find the execution information even after log out/log in. Once a process has been launched for execution, the user can no longer modify it. A launched process can be in one of the following states: submitted, running, completed, failed or killed. Ganga offers support for Grid certificates management both in the form of classic Grid proxy certificates as well as for VOMS extended certificates. The tool offers support for proxy certificate creation as well as useful notations regarding the certificates state.

Ganga offers an abstract level over the already existing Grid technologies for execution and management of jobs in Grid infrastructure: Globus, Condor, Unicore or gLite. The offered functionalities can be accessed by the user either through command line, Python scripts or in a graphical user interface. The architecture of this tool, as presented in (Moscicki et al., 2009 [154]) is illustrated in Figure 21.

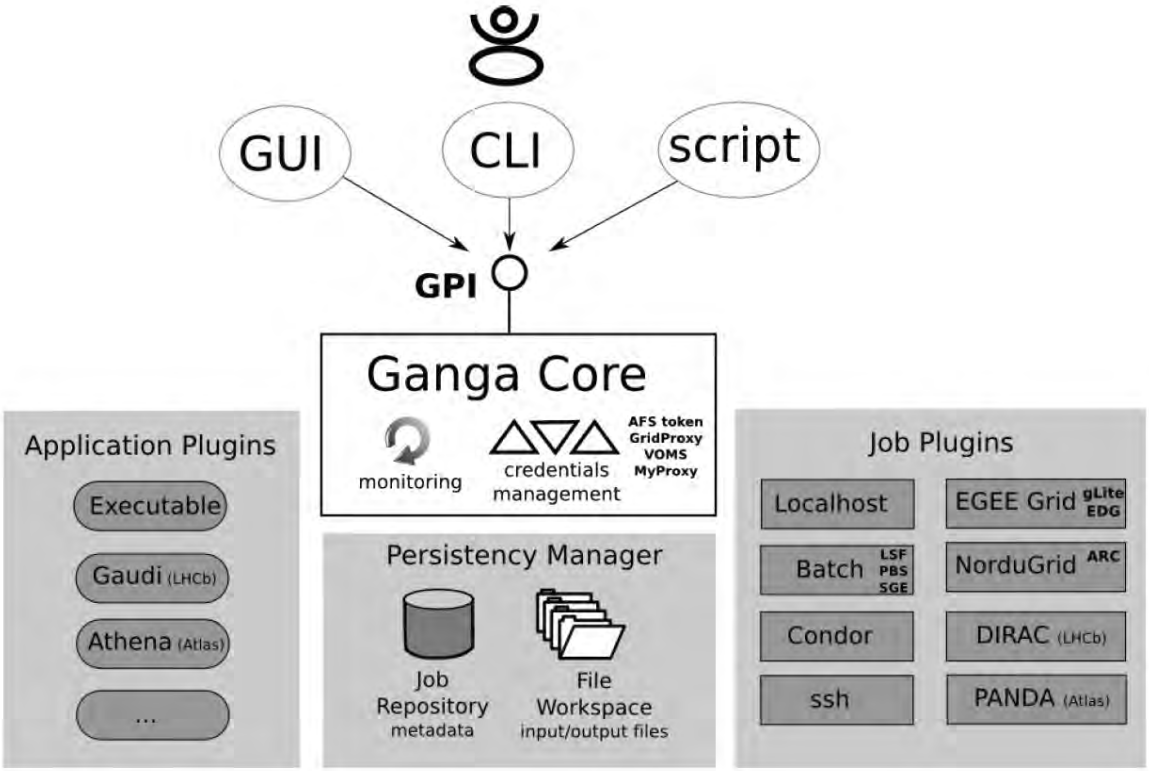

Figure 21: Ganga – Architecture (Moscicki et al., 2009)

5.4.4.2. DIANE

DIANE – Distributed Analysis Environment (Moscicki, 2003 [153]) is another Python tool developed at CERN, which allows the efficient Grid execution of applications. This tool allows the execution of applications on different Grid infrastructures. It uses a master-client model over the Grid services and the architecture is illustrated in Figure 22.

The idea of DIANE is not to assign a Worker Node for each sub-task in an application, as this would considerably increase the execution time or occupy a lot of resources, but to start a certain number of Worker Nodes, which will be responsible for all application sub-tasks. The Master component is responsible for assigning the tasks for each worker and to monitor the execution. Each DIANE Worker will execute one or more tasks, depending on several factors.

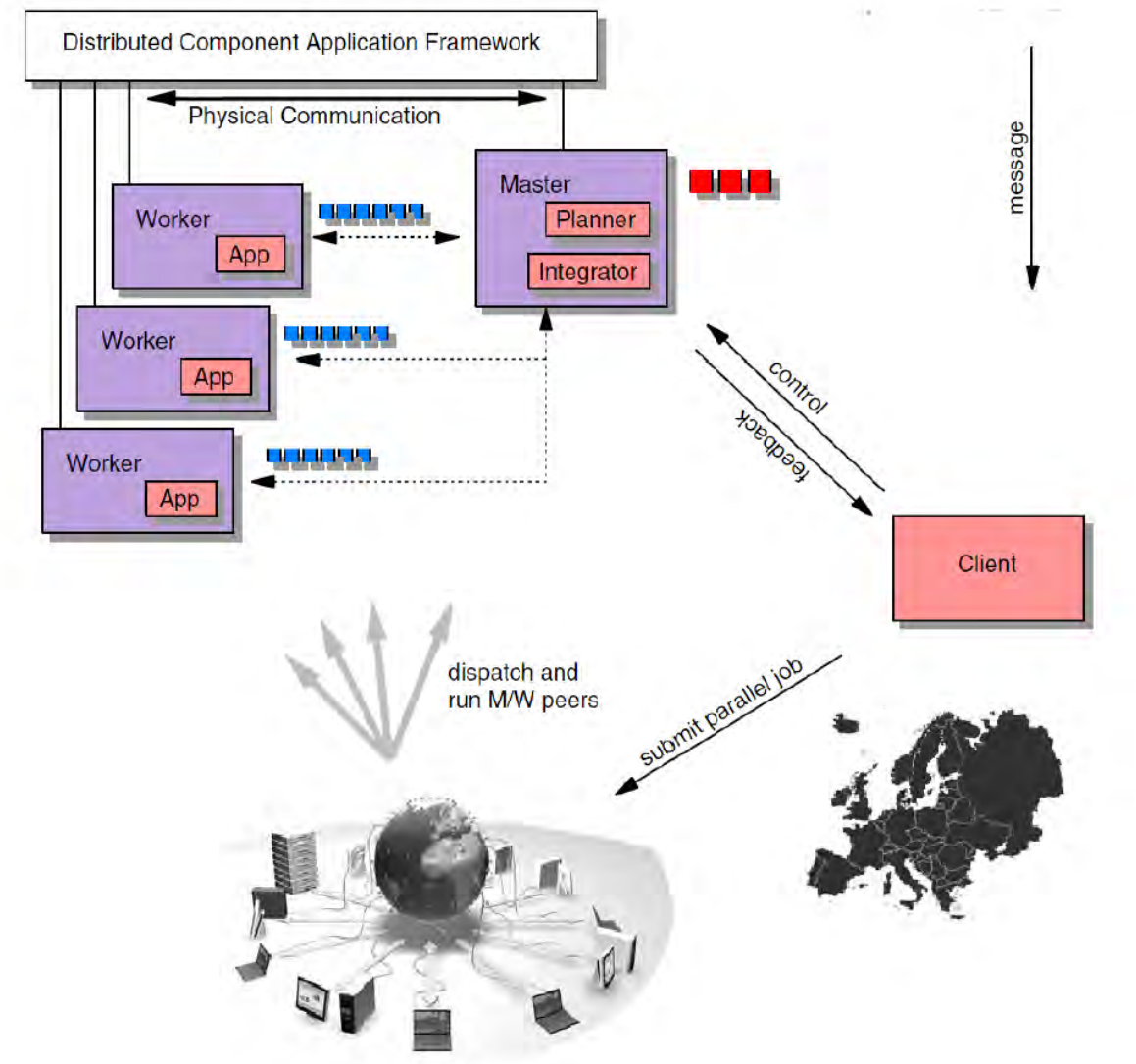

Figure 22: DIANE - Architecture (Moscicki, 2003)

Grid advantages (Taylor et al., 2004 [224]):

- Provides components for reliable data management (Grid-FTP);
- Resource allocation and management (GRAM – Grid Resource Allocation Manager);
- Information discovery and look-up (GIS – Grid Information Service);
- Authentication and security (GSS – Generic Security Service);
- Infrastructure monitoring;
- Virtualization through Open Grid Service Architecture (OGSA);
- Extensive set of resource management components;

- Facilitates access to geographically distributed data sets;
- Environment for a secure, scalable parallel and distributed set of resources.

From a user point of view, especially in the GI community, the main advantages offered by the Grid infrastructure can be summarized in:

- **Security** - the geospatial data is not shared in a distributed environment in a secure mode because the OGC Web services do not provide such security mechanisms. The Grid environment can facilitate these security mechanisms needed for transferring and processing heterogeneous geospatial data within a distributed environment.
- **Data management** - the Grid infrastructure offers functionalities for management of large amounts of data, functionalities that are strongly needed in the Geospatial domain due to the multiple heterogeneous sources of data.
- **Computational power** – needed by Geospatial applications, which usually work with data coming from different locations and in different formats, requiring special processing resources, most of the time available only on remote sites. The Grid environment is able to provide processing resources and high performance computing in a distributed environment (Di, 2004 [54]).

Grid disadvantages (Taylor et al., 2004 [224]):

- Lack support for some of the network and data management functionality;
- Lack support for application level multicast (most of the Grid application only need reliable data transfer);
- Grid provides support for data partitioning and replication but not for data indexing and retrieval.

Applications running on the Grid:

- Memory intensive application that need a message passing interface (MPI) may not be able to take advantage of the Grid infrastructure as the worker node do not communicate and the connection between them might not be fast enough (gigabit Ethernet) as needed for intense MPI applications;

- Applications requiring licensing are also hard to migrate on Grid infrastructures as it can be complicated to pay the license for all the nodes in the Grid and it can be complicated to schedule your work only on these who have the license installed;
- Political challenges associated with sharing resources across different administration domains may also restrict the use of some application on a Grid infrastructure (Taylor et al., 2004 [224]);
- Grid standards need improvements to have a larger set of applications taking advantage of its functionalities but the improvement of standards is usually done only when a significant number of industry players and government agencies recognize their need in this infrastructure and its standards (Taylor et al., 2004 [224]).

5.5. Cloud

5.5.1. General Description

Cloud computing was one of the most explosively expanding technologies. It is a commercially driven technology that offers services to access computational and storage resources, platforms and software solutions. This paradigm has many definitions and views, depending on the used context. A combination of these definitions is given by (Hajibaba and Gorgin, 2014 [103]):

"*Cloud is a parallel and distributed system consisting of a shared pool of virtualized resources (e.g. network, server, storage, application, and service) in large-scale data centers. These resources can be dynamically provisioned, reconfigured and exploited by a pay-per-use economic model in which consumer is charged on the quantity of cloud services usage and provider guarantees Service Level Agreements (SLA) through negotiations with consumers. In addition, resources can be rapidly leased and released with minimal management effort or service provider interaction. Hardware management is highly abstracted from the user and infrastructure capacity is highly elastic*."

The on-demand resource provisioning feature of Cloud computing (due to virtualization) enables capacity resizing, workload migration, better availability and better performance. With all this, Cloud interoperability limits resource sharing across providers and does not excel either in security and built-in workload management services (Mateescu et al., 2015 [146]).

5.5.2. Cloud Delivery Models

Cloud computing offers its benefits through three types of services or delivery models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). These delivery models are illustrated in Figure 23 (Hajibaba and Gorgin, 2014 [103]) and described in the following.

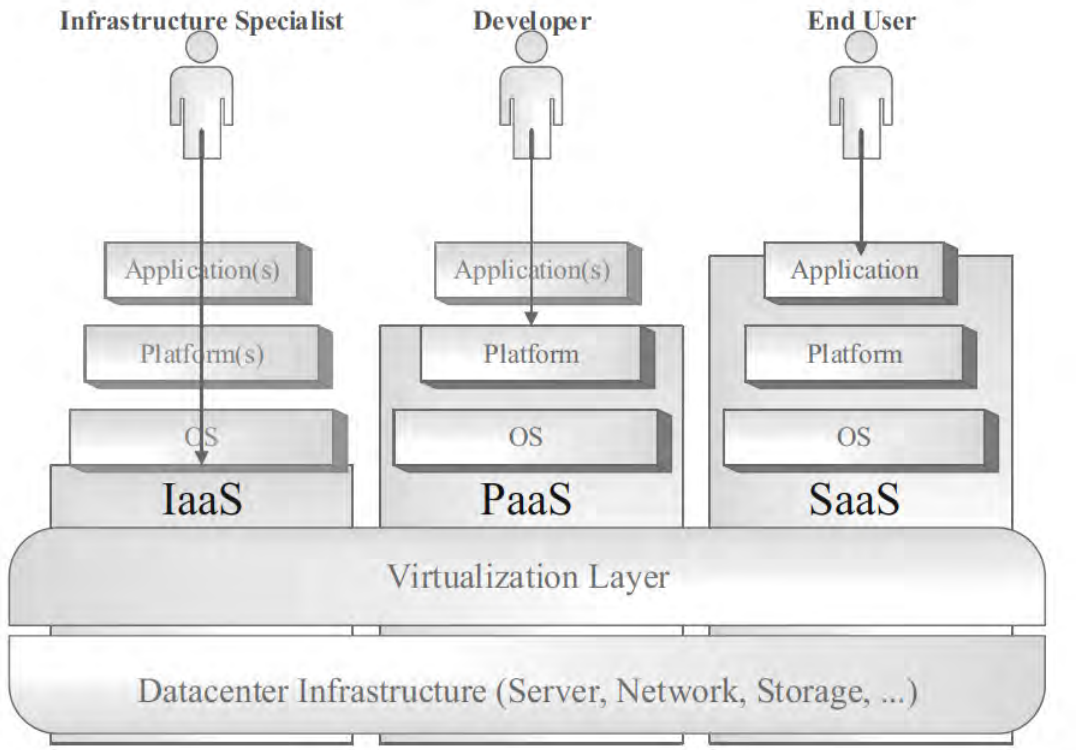

Figure 23: Cloud Delivery Models: IaaS, PaaS and SaaS (Hajibaba and Gorgin, 2014)

5.5.2.1. Infrastructure as a Service (IaaS)

An Infrastructure as a Service (IaaS) offers hardware related services using the principles of Cloud computing. This layer includes virtualized resources (storage, processors, and networks) that are used to deploy and run arbitrary software (Rings and Grabowski, 2012 [192]). Some authors (Kahanwal and Singh, 2012 [117]) also incorporate at this level some other services such as Communication as a Service (CaaS) and Data-storage as a Service (DaaS). Leading vendors that provide IaaS (Kaur, 2015 [122]): Amazon EC2, Amazon S3, Rackspace Cloud Servers and Flexiscale.

5.5.2.2. Platform as a Service (PaaS)

Platform as a Service (PaaS) offers a development platform on the Cloud. The platforms provided by different vendors are usually not compatible. At this level, services for automated resource management, fault tolerance, dynamic provisioning and load balancing are deployed (Rings and Grabowski, 2012 [192]). These services are used within a running environment in a transparent manner via an API. Important vendors in this context (Kaur, 2015 [122]): Google Application Engine and Microsoft Azure.

5.5.2.3. Software as a Service (SaaS)

Software as a Service (SaaS) offers a complete software solution, hosted on the Cloud and accessible on pay-per-use basis. This layer provides Web interfaces for users to access applications without the need to install local software (Rings and Grabowski, 2012 [192]). Examples of providers (Kaur, 2015 [122]): Salesforce.com offering the online Customer Relationship Management (CRM) space, Googles gmail and Microsoft hotmail, Microsoft online version of office – BPOS (Business Productivity Online Standard Suite).

5.5.3. Cloud Deployment Models

A Cloud can be deployed using different models: Private, Public, Hybrid and Community. These models are illustrated in Figure 24.

5.5.3.1. Private

In private Clouds, the computing infrastructure is dedicated only to a particular organization and not shared with others. They are more expensive and more secure as their access is restricted and can either be externally hosted (by a third party specializing in Cloud infrastructures) or on premise (Kahanwal and Singh, 2012 [117], Kaur, 2015 [122]).

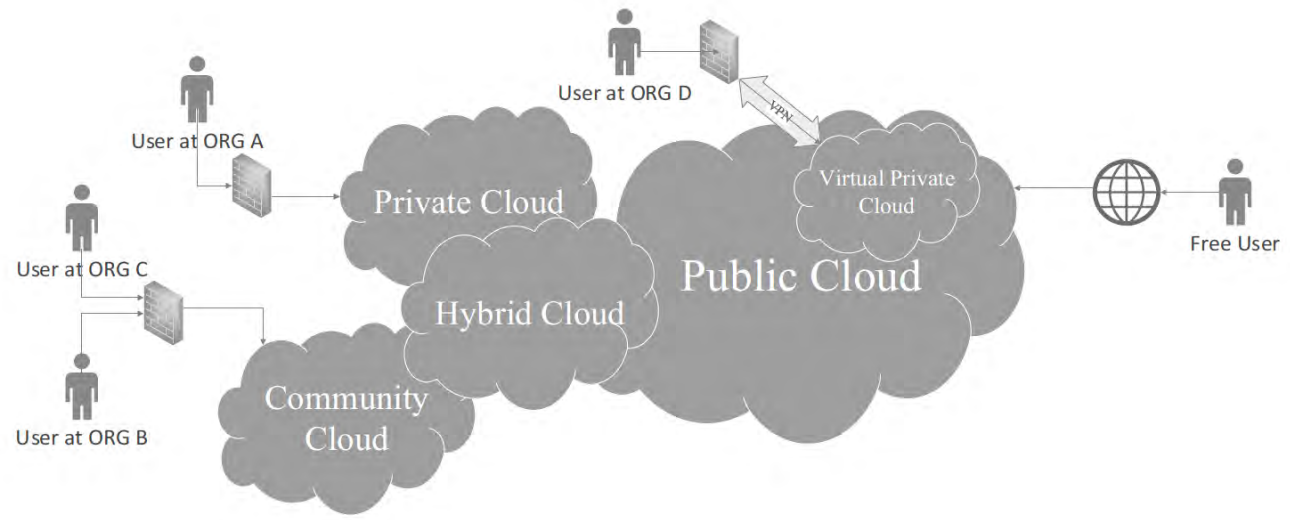

Figure 24: Cloud Deployment Models (Hajibaba and Gorgin, 2014)

5.5.3.2. Public

A Cloud vendor hosts the computing infrastructure of a public Cloud and it's usually shared between any organizations, or to general public. The resources are accessible via the Internet and the customer has no visibility or control over where the resources are located. The clients can choose the security level they need and negotiate service levels agreements (Kahanwal and Singh, 2012 [117], Kaur, 2015 [122]).

5.5.3.3. Hybrid

A hybrid Cloud is formed from at least a private Cloud and a public Cloud and it typically used by organizations that host critical applications on private Clouds and applications with softer security constrains on public Clouds. In this model, the combined clouds retain their identities but are combined together through standardized technology (Kahanwal and Singh, 2012 [117]). A hybrid Cloud can also be used in the context of a so-called Cloud Burst. In this situation, organizations use their own computing infrastructures for normal usage but access a public Cloud for high/peak load requirements (Kaur, 2015 [122]). General public does not have access to this type of Cloud.

5.5.3.4. Community

A community Cloud involves sharing the Cloud infrastructures among organizations belonging to the same community or having a shared concern or interest (government organizations for example). This type of cloud can be managed by the organization or by a third party and can be accessed by both the general public and the organizations forming the community.

5.5.4. Federated Cloud

As customer's requests for computational resources are getting higher and higher, public Cloud providers are force to increase the scalability more and more. Even so, sometimes only one public Cloud is not enough for satisfying all demands, introducing the idea of Cloud Federation or Federated Clouds (Caron et al., 2013 [38]). In such a Cloud Federation, the providers put their resources together to gain scalability. The problems that arise in this context are most of the time related with the appropriate selection of a Cloud among the federation of heterogeneous Clouds, based on computational power, availability, etc. An interesting solution for brokering is RightScale (RightScale [189]). This commercial solution allows you to manage applications that are spread between different Cloud Providers, using a single management interface, which interacts with a series of providers, such as Amazon Web Services, Windows Azure, Rackspace, CloudStack and OpenStack. Solutions on the academic side, based on Cloud brokering, but with less advanced functionalities then commercial ones (Caron et al., 2013 [38]), include RESERVOIR (Rodero-Merino et al., 2010 [193]), CLEVER (Tusa et al., 2011 [227]). These projects all try to leverage multi-Cloud resources while hiding the complexity of managing them. (Caron et al., 2013 [38]) describes an improvement of the DIET (Caron and Desprez, 2006 [37]) architecture. They enable DIET, which is a hierarchically structured component (scalable middleware), to benefit from on-demand resources when handling client service requests. The improved architecture upgrades DIET to a multi-Cloud middleware designed to interact with multi-Cloud platforms, while hiding the complexity and the heterogeneity that lies behind.

Cloud computing is a computing paradigm which has brought tremendous convenience for the processing of large-scale datasets but the main identified Cloud disadvantages are the need of high speed internet connection needed and the security issues.

5.5.5. Cloud Providers

During our research, we have been working with two main Cloud providers, i.e. Open Stack and Microsoft Windows Azure.

5.5.5.1. Open Stack

Open Stack Cloud was designed to easily scale out, with a modular architecture (Figure 25), which is based on a growing set of core services and a set of shared services, which provide an integrated cloud management.

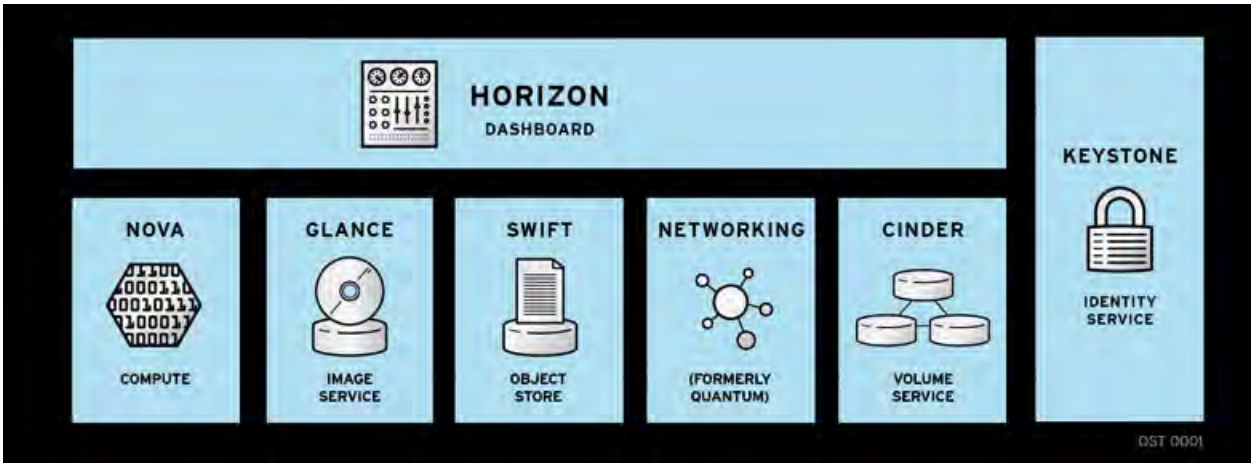

Figure 25: OpenStack Architecture

The core services include the "Compute", "Storage", "Networking" and "Dashboard" components while the shared services are "Identity" and "Image".

The OpenStack Compute component (Nova) is used to provision and manage large networks of virtual machines and includes features for managing virtualized commodity server resources (CPU, memory, disk, network interfaces, etc.), managing local area networks (Flat, Flat DHCP, VLAN DHCP, IPv4 and IPv6 networks, etc.), managing virtual machine images (store, share, import, query, etc.), assigning floating IP addresses to VMs, providing security (role based access control - RBAC), etc. OpenStack compute is compatible with tools such as Hadoop ([100]) and High Performance Computing (HPC) applications.

Hadoop is an open source platform that consists of the Hadoop kernel, Hadoop Distributed File System (HDFS [101]) Map Reduce and several other instruments. Hadoop is currently one of the most mature, accessible and popular implementation of the MapReduce

programing model. This tool has been widely adopted both in scientific communities and in enterprises to solve Big Data related problems. Together with other associated systems, it provides a scalable, large-scale solution to data storage, analysis and mining.

Hadoop Distributed File System (HDFS) (HDFS [101]) was developed from the need to store Big Data on multiple machines. HDFS is a block-structured distributed file system, which was designed to hold huge amounts of data, offering scalable and easy to use functionalities to operate with it. It also offers a great portability (as it is written in Java) as well as a great reliability (all files are replicated on two or more Data Nodes). HDFS is based on the principle that moving the computation to the data is cheaper than moving the data where the processing takes place, especially in the cases where the input data has a huge volume/size (Garlasu et al., 2013 [76]).

OpenStack Storage (Swift & Cider) provides both object and block storage. Object storage is a distributed storage system for static data (virtual machine images, backups, archives), which is written to multiple disk drives spread throughout the Cloud (LaBarge and McGuire, 2012 [128]). It also provides persistent block storage devices. Examples of usage include high performance storage for databases, servers that need access to raw block level storage, expandable file systems, etc. (LaBarge and McGuire, 2012 [128]).

The OpenStack Networking component (Quantum) is an API driven system for managing Cloud networks and IP addresses (static, DCHP or floating IP addresses). It also provides several networking models (such as flat or VLANs) and allows users to create and manage their own network (OpenFlow and SDN – software defined networking technology).

OpenStack Dashboard (Horizon) is implemented as an extensible web-based application and allows both administrators and normal users to manage and control Cloud resources (compute, storage and networking). It is used to create users and projects, to assign users to the created projects and to control the resources allocated to the projects.

OpenStack Identity (Keystone) maintains a database of users, which are mapped to the services they are allowed to access. It provides a common authentication system across the Cloud, which can be of multiple forms, including username and password, Amazon Web Services style login and token-based systems. It also provides facilities to define permissions for different types of resources and to set common policies across users and services.

OpenStack Image (Glance) provides services for disk and server images (discovery, registration, delivery). The administrators can create template images, which can be further instantiated by the users. The images can be stored in different formats such as Raw, VHD (Hyper - V), VDI (Virtual Box), qcow2 (Qemu/KVM), VMDK (WMware) and OVF (VMware, others) (LaBarge and McGuire, 2012).

5.5.5.2. Windows Azure

Windows Azure is an open and flexible global Cloud platform, which provides researchers with the power and scalability of Cloud computing for collaboration, computation and data-intensive processing. Windows Azure is deployed around the world, in different regions (places where you can choose to place and run your applications): 4 regions in North America, 2 regions in Europe and 2 in Asia, each regions with huge datacenters, hosting 10s or hundreds thousands of servers. These regions are rapidly expanding around the world. The user can run his/her application in different regions, even simultaneously, by redirecting the traffic to the more convenient region.

Compute models available in Windows Azure are Virtual Machines, Web Sites and Cloud Services. All three allows you to build scalable, reliable applications in the Cloud but it depends on the user requirements which one is more suitable.

Virtual Machines (IaaS):

- Enable you to be admin on the box;
- They are durable (if you reboot the VM it will be still there, with all the changes and data you stored to disk);
- SSH or remote desktop connection to run any workload;
- Private networking possibility: deploy virtual machines in the Cloud and group them together so they are part of their own network. You can also then connect it back to your corporate network (if you have one) and establish a VPN secure tunnel to link your machines running in your own corporate environment up to your virtual machines in the Cloud – making them look like they're all part of one connected network;
- Huge flexibility both on the compute and networking sides;
- They have drivers (backed up in the Windows Azure storage) which are triply replicated, providing reliability;

- Continuous storage geo-replication: when a user saves something in the storage system, it is automatically replicated to another data center, in the background (if this option is not deactivated).

Web sites (specific PaaS):

- Another flexible and very fast deployment compute model in Windows Azure;
- Are managed services that can be used to run Web sites and Web APIs on the internet;
- Are deployed in seconds using several flexible deployment options such as FTP, Git or TFS (Team Foundation Service);
- Focus is on building and deploying HTTP based applications and not on VM, servers or infrastructure;
- Allows users to use any tool and any OS;
- Machine instances can be registered and scaled out as needed to additional VMs.

Cloud services (full PaaS):

- Another model used for building applications in Windows Azure Cloud;
- Enable a broader set of workloads then Web sites while providing more automated management then VMs;
- Enables users to build highly scalable applications and services;
- Support not only Web based deployments, but also multi-tier architectures where you might have a combination of front ends, middle tiers, as well as virtual machines running as part of your solution;
- Supports automated application management, making it easy to deploy, scale out, isolate, and recover from any type of hardware failure. As well as support for automated updates;
- A Cloud service is in essence a container of related service roles: Web role (Web server instances) and Worker role (VM that manage computation and data).

Windows Azure provides also applications building blocks for: Big Data, Database, Storage, Traffic, Caching, Messaging, Identity, Media, CDN and Networking. These are all managed services that provide a lot of value. Any of these services can be used with a VM, a Web Site or a Cloud Service, giving enough flexibility to consume them based on your needs.

The storage system used in Windows Azure is a highly available, scalable and secure file system, in which you can store any type of data. Users can expose the storage through HTTP

URLs and make it either public or private. A new storage can be created in a few minutes and the continuous geo-replication is enabled by default. The storage services include: Blob service (binary and text data), Queue service (stores messages that may be accessed by a client), Table service (for structured storage for non-relational data) and Windows Azure drivers (for mounting an NTFS volume accessible to code running in your Windows Azure service).

The Service Bus is another managed service, which provides secure messaging, relay and queue capabilities. It securely integrates cloud-based solutions with on premise environments and it enables a very loosely coupled architecture. The Service Bus can be used from any OS (whether it's a VM, Web site or a Cloud Service) and with any supported programming language due to the existing cross platform libraries.

5.6. Fog Computing

5.6.1. General Description

Fog computing is a new paradigm which extends the Cloud computing and enables computing at the source of the data as it's pushing the frontier of computing away from centralized nodes to the edge of a network. This highly virtualized platform provides compute, storage and networking services between end devices and traditional Cloud Computing Data Centers. To achieve this, it enables a new breed of applications and services (Bonomi et al., 2012 [30]).

The Internet of Things (IoT) (also known as Internet of Objects) is a network of individual networks connected together with security, analytics and management. It covers a wide range of technologies and envisions a variety of "things" or objects, which are physically or virtually interrelated and are able to communicate with each other to deliver a new class of applications and services (Atzori et al., 2010 [14]). IoT has as main objective to smartly connect intelligent devices supporting wireless communication with existing networks and participating to computational tasks, using network resources (Madsen et al., 2013 [142]). The IoT has the ability to gather, analyze and distribute data that we can turn into information, knowledge and ultimately wisdom (Evans, 2011 [69]).

The IoT model is based on the H/M/P structure (Madsen et al., 2013 [142]):

- H – Hardware: sensors, actuators, embedded communication hardware;

- M – Middleware: providing on demand storage and data analysis tools;
- P – Presentation: visualization and interpretation tools;

By extending Cloud computing services to include smart sensors and intelligent devices, Fog computing has a big influence in the advancement of Internet of Things (IoT) (Madsen et al., 2013 [142]).

Some of the main characteristics of Fog Computing (Bonomi et al., 2012 [30], Hajibaba and Gorgin, 2014 [103]) make this paradigm appropriate for a number of critical (IoT) services and applications:

- Proximity of data to end-users;
- Low latency, location awareness and edge location;
- Wide spread (dense) geographical distribution;
- Hierarchical organization;
- Mobility;
- Huge number of nodes;
- Large scale sensor networks;
- Predominance of wireless access;
- Strong presence of streaming and real time applications;
- Heterogeneity;
- Interoperability and federation;
- Integration with the Cloud;
- Support for online analytics.

5.6.2. Architectures

In a Fog infrastructure, the data, processing and the applications are concentrated in devices at the network edges, being thus closer to the end-user, rather than being centralized like in Cloud. A Fog infrastructure multi-tier architecture is presented in (Hajibaba and Gorgin, 2014 [103]) and illustrated in Figure 26.

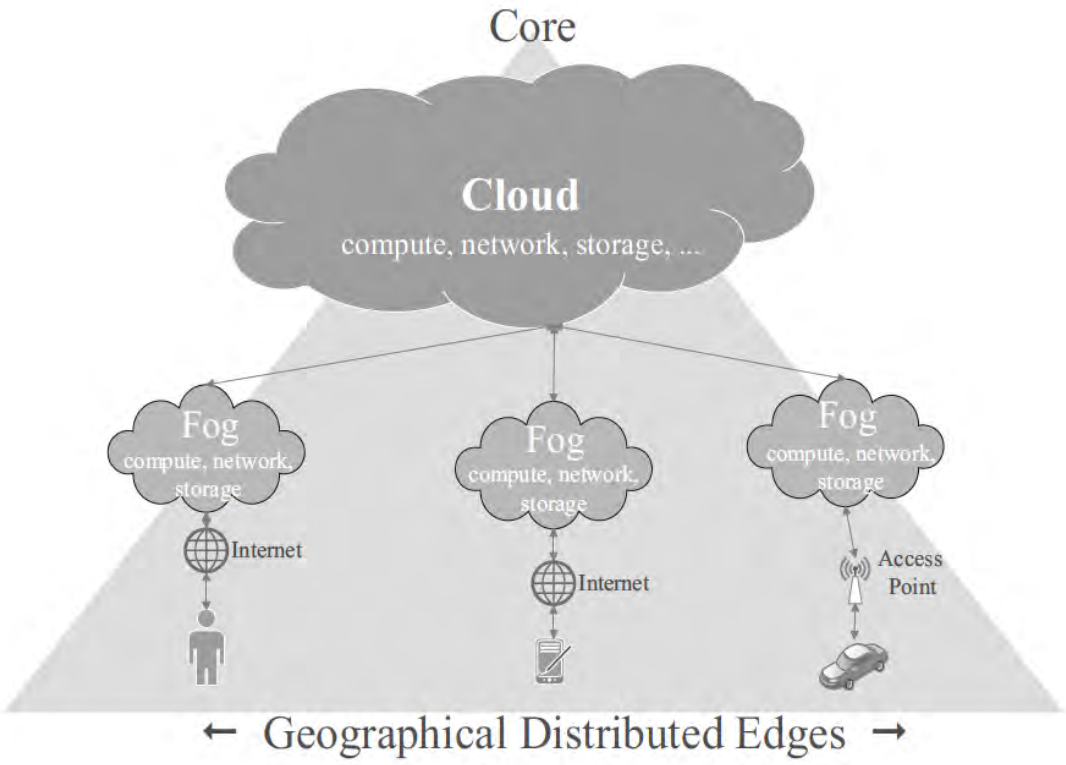

Figure 26: Fog Computing Architecture (Hajibaba and Gorgin, 2014)

Fog collectors of data generated by smart devices (vehicles, sensors, mobiles, etc.) are located at the border of the network and represent the first tire in the architecture. This first tire is designed for M2M (machine to machine) interaction and provides the following functionalities (Madsen et al., 2013 [142]):

- Collects and process data from devices using a volatile memory (memory that requires power to maintain the stored information);
- Sends control commands to the actuators;
- Filters the local data and sends the rest of the data to upper tiers, responsible with virtualization and reporting, and into the Cloud.

Figure 26 shows how theoretically Fog is locate below the Cloud and it is similar with an optimized transfer medium in which the Cloud services, compute, storage, workloads, applications and Big Data can be provided at the edge of the network in a truly distributed way

or as (Hajibaba and Gorgin, 2014 [103]) defines it: "FOG brings computing from the core to the edge of the network". This approach enables analytics and knowledge generation at the data source. The idea of distributing the Cloud data and placing it closer to end-users has many advantages: eliminates service latency, improves QoS and also eliminates other data transfer issues. All this make the Fog computing paradigm a good candidate for Big Data, real time analytics, mobile computing and data streaming applications (Hajibaba and Gorgin, 2014 [103]).

The term of Fog was chosen because fog is a cloud closer to the ground as Fog computing is similar to Cloud computing but closed to the data. The aim of Fog computing is to bring the hardware and software virtualization from Cloud to earth, closer to the user (Bonomi et al., 2012 [30], Hajibaba and Gorgin, 2014 [103]). The idea of Fog Computing is not to replace the Cloud but to enhance it by including smart sensors and intelligent devices. It isolates the user data, which are exclusively located at the edge of the network and allows the users to connect analytics, security functions and other services directly to the Cloud.

Fog computing is still a new paradigm, under development and needs some time to become mature enough to be used in production systems but the opportunities are huge.

5.7. Hybrid Computing

5.7.1. Concept Description

The scientific applications of high-performance and distributed computing are widely spread and the most popular parallel and distributed platforms among scientists are Multicore processors, clusters, Grids and Clouds systems. Due to the integration of many-core technologies, these infrastructures are undergoing revolutionary changes, providing orders-of-magnitude speed improvements. Although each of these systems is straightforward to program, due to many existing platform specific tools, creating applications able to run on an optimum combination of such systems becomes difficult. The programming complexity of the applications, running on such heterogeneous and hierarchical platforms, has vastly increased also because of data distribution, need of scalability and software heterogeneity (Maassen et al., 2011 [141], Seinstra et al., 2011 [209]). These issues lead most of the times to the need of simultaneously using multiple platforms at the same time. In many realistic scientific research areas, domain experts are actually forced to concurrently use multiple infrastructures. All of

these infrastructures are undergoing many changes due to the integration of core technologies, providing speed improvements but becoming also more heterogeneous, complex and hierarchical.

With platforms becoming more complex and heterogeneous, the programming of the applications is even harder and the efficient mapping of these applications to a suitable and optimum platform is challenging and complex as well. The problem becomes even harder to solve when the programmer has to take into considerations also the selection of an optimum computing platform among the available ones or even a combinations of simultaneous usage of several computing platforms, depending on the specific applications. The need to use multiple computing platforms for running an application is due to cases in which the reservation of a sufficient number of computing nodes in a single platform is impossible but also due to the distributed nature of the input data, heterogeneity of the software, ad-hoc availability of the resources etc. (Seinstra et al., 2011 [209]).

The idea of a hybrid-computing environment (defined as the use of multiple diverse distributed and highly non-uniform high performance platforms and systems simultaneously, to achieve peak performances) is not new and it was born mostly because of the urgent need of scalability, data distribution, heterogeneity in software and hardware resources – all at once – required by many scientific problems of great complexity. This idea was discussed in previous works as well (Mateescu et al., 2011 [146], Belgacem and Chopard, 2015 [25], Borgdorff et al., 2011 [31], Borgdorff et al., 2013 [32], Seinstra et al., 2011 [209]) even though it appears under different names such as *Hybrid High Performance Computing*, *Elastic Cluster*, *Jungle Computing*, *Heterogeneous Distributed Computing*, etc.

In essence, a hybrid-computing environment is illustrated in Figure 27. Such an environment will become even more a key component for the research agenda of the coming years.

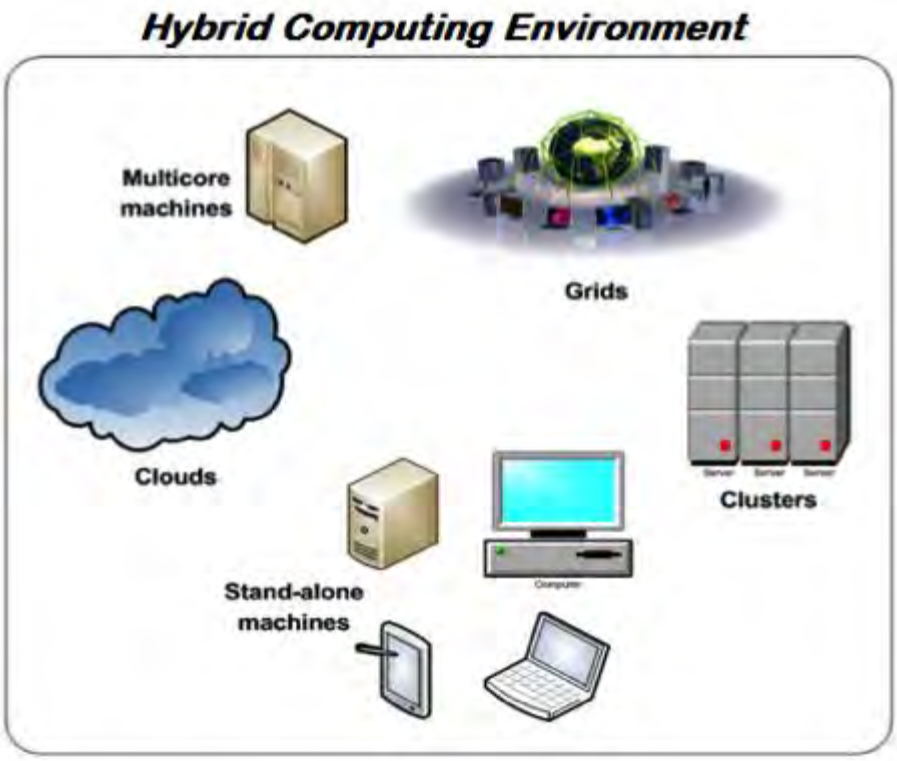

Figure 27: A Hybrid Computing Environment

5.7.2. Initiatives

An interesting approach of distribution among several supercomputing facilities is proposed in (Borgdorff et al., 2011 [31], Borgdorff et al., 2013 [32]) where the authors describe the concept of Distributed Multiscale Computing (DMC). The proposed approach allows different components of a multiscale application to be distributed on several supercomputing facilities and run in a tightly and/or loosely coupled way. This gives the users the possibility to access at once computing resources that are available at different computing centers. The DMC also allows running parts of an application on the most appropriate hardware such as GPU.

A hybrid High Performance Computing (HPC) infrastructure architecture, also called Elastic Cluster, is introduced by (Mateescu et al., 2011 [146]) in which three different types of resources are combined and the benefits of these technologies are strengthen: owner-centric HPC (traditional), Grid computing and Cloud computing.

The idea of Jungle Computing (defined as a new distributed computing paradigm) presented by (Seinstra et al., 2011 [209]) consists of all compute resources available to an end-

user, including clusters, supercomputer, clouds, grids, desktop grids, as well as stand-alone machines and even mobile devices.

Software frameworks have emerged as well in this direction, allowing users to execute external application over different computing resources such as GC3Pie [78], Ibis [108], etc.

GC3Pie (Maffioletti and Murri, 2012 [143]) is a Python framework that aims to orchestrate the execution of external commands over different computing resources (such as a Sun/Oracle/Open Grid Engine cluster, the Swiss National Distributed Computing Infrastructure SMSCG, OpenStack Cloud, ARC-based computational grid, etc.). It is a flexible framework that allows the implementation of command line driver scripts (in the form of Python object classes) that can be customized easily by overriding specific object methods. GC3Pie also conceptualizes the executed applications but using plain programming language (i.e., you describe your application using a set of Python classes which can be extended and specialized). The tool was designed to coordinate the execution of independent applications meaning that it is used to steer the computation, not to perform it.

Ibis software system for high-performance and distributed computing is presented in (Seinstra et al., 2011 [209]) as an implementation of a Jungle computing system able to simplify the programming and the deployment of applications on such an environment. Another platform designed to run on such a Jungle computing system is called Constellation and a specific usage and description of this lightweight software is mention in (Maassen et al., 2011 [141]). The scope of Constellation is to run applications efficiently on complex combinations of distributed and heterogeneous computing hardware.

5.8. Distributed Applications

Parallel and distributed applications are programs meant to run over parallel and distributed resources, having mechanisms to coordinate the communication between their components.

Some confusion may come up when we talk about concurrent processing, parallel processing and distributed processing. Although these terms have common properties, they are somehow different. In parallel processing, all the processors share the same memory and can easily exchange massages among them. In distributed processing, each processor has its own memory, and the information exchange between processor is done through messages. In the

past, distribution use to mean different geographic locations but today it can also refer to independent processes running on the same physical machine but in different virtual spaces.

Considering the parallel programming model, we have to take into account two major aspects: the problem decomposition and the interaction between processes. The interaction between processes can usually be done either through shared memory or through messages. In a shared memory model, the processes share a common address space where they can read and write asynchronous, requiring special mechanism for memory consistency (such as locks, and semaphores). In a message interaction model, the parallel processes communicate through messages either synchronous or asynchronous. The problem decomposition in a parallel model can be done either at the process level or at the data level or both. The parallelism at the process level is usually called MISD (Multiple Instructions Single Data) or MPSD (Multiple Processes Single Data), the parallelism at the data level is labelled as SIMD (Single Instruction Multiple Data) or SPMD (Single Process Multiple Data) and both types of parallelism are usually referred to as MIMD (Multiple Instructions, Multiple Data) or MPMD (Multiple Processes, Multiple Data).

Parallel applications are considered to be distributed in one of the following cases:

- The processing of the application is distributed among several processing stations;
- The processed data are distributed – this can happen due to several reasons such as: security policies, access rights, large data transfer costs, etc.;
- The users of the application are distributed.

Distributed applications can be further classified as following:

- Batch Processing applications:
    - Singular jobs:
        - HPC jobs (MPI);
        - HTC jobs (BOT, …, Image rendering);
        - Many-Task computing;
        - Service workflows.
    - Periodical jobs:
        - Data centric (Data warehouse, …, Reporting);
        - Administrative (Cron jobs).

- Interactive applications:
    - Compute intensive (Multiplayer online games);
    - Data centric (OLTP, …, OLAP).

5.8.1. Message Passing Interface (MPI)

The Message Passing Interface (MPI) (Snir et al., 1998 [216], Gropp et al., 1999 [97]) is an efficient programming model successfully used in the high performance computing community for years. It provides highly optimized communication operations but it can only be used with a high-end and high-bandwidth network. Therefore, MPI allows efficient parallel programs to run on cluster nodes and multicore machine interconnected with a high speed network connection but not on Cloud platforms, which lack of low latency high bandwidth network capabilities. Attempts to implement MPI on Cloud platform still exists (Agarwal et al., 2014 [6]).

This paradigm requires high skills in parallel programming because the parallelization and the communication operation are explicitly handled and codded by the programmer. As examples of MPI implementation, we can specify OpenMPI (www.open-mpi.org) and MPICH (www.mpich.org).

5.8.2. Bags – of – Tasks (BoT)

Bag-of-tasks (BOT) model refers to a parallel computation composed of independent jobs that form a single logical computation. A successful BOT implies the successful termination of all jobs. BOTs are traditionally the most common type of parallel applications using Grid resources (Silberstein et al., 2009 [212]). This model is in general well suited for parameter-sweep applications, where tasks are independent and operate on different input files.

Development and Execution of Environmental Applications

Chapter 6: Development and Description of Different Environmental Applications

In this chapter we will present a set of environmental applications that we have developed and/or we have used as use cases within different national and international research projects.

6.1. Research Projects

6.1.1. MedioGRID (UTCN)

MedioGRID (Parallel and distributed graphical processing on Grid structure of geographical and environmental data) (http://cgis.utcluj.ro/projects/mediogrid) was a national research project funded by the Ministry of Education and Research by the Excellence Research Programme (2005-2008, contract 19CEEX-I03). The main goal of the project was to accomplish a pilot program to process the images acquired in real time from meteorological and resource satellites, in order to extract the meteorological and environment parameters that characterize the atmospheric and terrestrial state.

MedioGRID was an academic and research Grid network that connected, over a wide geographic area, servers and workstations located at each of the seven partners from Cluj-Napoca, Timisoara and Bucharest. The main objectives of the project were: to develop a Grid structure to support the parallel and distributed processing of huge data (geographical and environmental); to develop algorithms for Grid based processing of satellite images; to develop and experiment environment supervising applications with data extracted from satellite images; and to model and visualize the virtual geographical space.

Within this project, the first version of gProcess platform (http://cgis.utcluj.ro/applications/gprocess) was developed, supporting the development and the execution of satellite image processing over the Grid infrastructure. We also performed experiments on the development and execution of applications in Earth and Environmental Sciences domain through the GreenLand application (http://cgis.utcluj.ro/applications/greenland).

6.1.2. GiSHEO (UTCN)

GiSHEO (On demand Grid services for high education and training in Earth Observation) (http://cgis.utcluj.ro/projects/gisheo) is a research project funded by European Space Agency through the ESA-PECS Programme (01 January 2008 - 31 December 2010).

The main goal of the project was to set-up and to develop a reliable resource for knowledge and associated instruments for higher education and training, using existing distributed information on Earth observation and Grid technology. This will enable higher exploitation of the potential of ESA database information and synergies orientated towards education and training in Earth observation.

6.1.3. enviroGRIDS (UTCN -UNIGE)

enviroGRIDS (Building Capacity for a Black Sea Catchment Observation and Assessment System supporting Sustainable Development, Grant Agreement n 226740) (http://www.envirogrids.net/) is an 7th Framework Program for EU Research (FP7) project, which ran from 2009 to 2013. The aim of the project was to build capacities in the Black Sea region to use new international standards to gather, store, distribute, analyze, visualize and disseminate information on past, present and future states of the region to be able to assess its sustainability and vulnerability.

The Black Sea Catchment region is a huge geographical area and a very complex environment. The development of a hydrological model for this watershed implies therefore highly interconnected and continuously evolving interactions at many spatial and temporal scales (Gorgan et al., 2013 [92]). Among the main objectives of the enviroGRIDS project was also the development of a high-resolution model for the entire Black Sea Basin (BSB) able to evaluate the impact of land use and climate change on the water resources. The development, calibration and execution of the SWAT hydrological model for the BSB region were successfully performed.

Within this project, we have developed and integrated environmental applications such as: gSWAT (http://cgis.utcluj.ro/applications/gswat) and GreenLand (http://cgis.utcluj.ro/applications/greenland).

6.1.4. enviroPAD (UTCN-UNIGE)

enviroPAD (Efficient Development and Execution of Environmental Applications on Parallel and Distributed Infrastructures) (https://www.unige.ch/envirospace/projects/enviropad/) was an 18-month project run from July 2013 to December 2014 and it was funded by the CRUS/SCIEX program (www.sciex.ch). The main goal of this project was to study and experiment a general solution/methodology for simultaneously running environmental applications on different parallel and distributed environments such as Multicore, cluster, Grid, Cloud, etc. The simultaneous execution on multiple environments is a complex and challenging task as it involves not only the interoperability of the applications with different parallel and distributed environments but also the interoperability and the coexistence of these environments. The research in this project was interdisciplinary as it requires the involvement of different leading experts from both the Environmental community (provided by enviroSPACE team and other colleagues at UNIGE and UNEP) and the Computer Science field (provided by the Technical University of Cluj Napoca, with inputs from the UNIGE teams). The developed methodology would allow users to easily port environmental applications to run on multiple environments and to benefit from all the advantages exposed by these platforms. Both the environmental and computer sciences communities will profit from the outputs of the project.

6.1.5. UNEP Live (UNIGE-UNEP/GRID-Geneva)

UNEP Live platform (http://www.uneplive.org/) is the main tool for environmental reporting and assessment for UNEP. The project has as main goal to facilitate the exchange and sharing of latest data, information, assessments and knowledge amongst member countries, research networks, communities of practice, indigenous people, and society with the scope of keeping the environment and the emerging issues under review.

EDAP application was developed and integrated in this project.

6.1.6. LiMES (UNIGE-UNEP/GRID-Geneva)

LiMES (Live Monitoring of Earth Surface) (http://limes.grid.unep.ch/) is a project currently under development at UNEP/GRID-Geneva, which started in 2015 and has as main goal to obtain the automatic monitoring of landcover changes using satellite imagery.

Within this project we have developed the LiMES application (http://limes.grid.unep.ch/).

6.2. gProcess

The gProcess platform (http://cgis.utcluj.ro/applications/gprocess) is an interactive toolset supporting the flexible description, instantiation, scheduling and execution of the Grid processing. The gProcess platform is a collection of Grid services and tools providing the following basic functionality:

- Visual manipulation based interactive description of the Grid based satellite image processing by pattern workflow like directed acyclic graph (DAG);
- Development of hypergraphs as complex composition of basic operators, Grid and Web services, and subgraphs;
- Pattern workflow instantiation for particular satellite images;
- Satellite data management, access and visualization;
- Workflow based Grid execution;
- Process execution control and visualization;
- Optimal execution for appropriate mapping of the processing over the Grid resources. The optimal processing is achieved in terms of code optimization, total execution time, and data communication costs over the Grid.

6.3. SWAT Hydrological Model

6.3.1. General Presentation

SWAT (Soil Water and Assessment Tool) (http://swat.tamu.edu/) is a physically based, open source hydrological model used for simulating different physical processes and predicting the impact of water and land management, sediment and agricultural chemical yields in large, complex watersheds, with varying soils, land uses, and management conditions. It was developed in the early 1990s as a continuation of the modeling experience done by the USDA Agricultural Research Service (ARS) and it is meant to assist water resource managers in assessing the impact of management on water supplies in watersheds and river basins (Arnold et al., 1998 [13]). The goal was to have a model that 1) is computationally efficient, 2) allows considerable spatial

detail, 3) requires readily available inputs, 4) is continuous-time, 5) is capable of simulating land-management scenarios, and 6) gives reasonable results. On top of offering all that, SWAT is also an open source model, which helped to quickly become popular worldwide and to gain a large user community.

SWAT is a basin scale, continuous time model that operates on a daily time stamp (Gassman et al., 2007 [77]). It divides watersheds into sub-basis and further on into smaller units called Hydrological Response Units (HRUs). Sub-basins are spatially connected with river network but the HRUs have no spatial connection. HRUs are considered as cells within which the hydrologic process can be treated as homogeneous (Arnold et al., 1998 [13]) (homogeneous land use, management and soil characteristics). The sub-division in HRUs is performed in a stochastic manner by considering the unique combinations of land use, soil, and slope, without having a specific location in the sub-basin. This type of sub-division was necessary because of the large degree of heterogeneity within a watershed, which can come from differences in climate, topography, soil and geology but also from the boundaries separating soil types, geologic formations or land covers. The sub-basin analysis is divided into 8 different areas: hydrology, weather, sedimentation, soil temperature, crop growth, nutrients, pesticides, and agricultural management.

The data needed for SWAT modeling is usually divided into spatial and no-spatial data. The spatial data is formed mostly from raster maps for Digital Elevation Model (DEM), land use maps and soil maps, and from vector data such as river geometry (digital stream network). The non-spatial data (hydrological and weather data) consists from meteorological variables like precipitation, temperature, wind speed, solar radiation, and relative humidity on daily or sub-daily time steps. Hydrological and weather data usually come as tables (or arrays of points). All these data are needed to simulate soil water content, surface runoff, nutrient cycles, energy, soil temperature, mass transport, land management etc., at HRU level and then aggregate them at sub-basin level (Abbaspour et al., 2007 [2]).

6.3.2. SWAT Calibration

Like for most of the other hydrological models, obtaining meaningful results from SWAT implies some vital aspects: sensitivity, calibration, and uncertainty analysis (Gassman et al., 2007 [77]). Sensitivity analysis in the process related with selecting the input parameters that have the

greatest impact on the SWAT output. In the literature there are reported a large number of sensitivity analysis methods.

Manual calibration involves comparing measured and simulated values and then using expert judgment to determine which values to adjust, how much adjustment they need and in the end determine when reasonable results have been obtained (Gassman et al., 2007 [77]). Automated techniques on the other side use Monte Carlo or other parameter estimation schemes able to automatically detect what is the best choice of values for a set of parameters, usually based on a large set of simulations (Gassman et al., 2007 [77]).

A calibration of the SWAT model is therefore a repeating process of changing the parameter values, followed by a model run, until a certain evaluation criteria is met. This process is illustrated in Figure 28.

Both sensitivity analysis and calibration techniques can be evaluated with a wide range of graphical and statistical procedures and can be performed either manual or automatically (Gassman et al., 2007 [77]).

As SWAT input parameters are physically based, they are allowed to vary within a realistic uncertainty range during calibration. Uncertainty is defined in the literature as "the estimated amount by which an observed or calculated value may depart from the true value." The main sources of uncertainty are reported as: model algorithms, model calibration and validation data, input variability, and scale (Gassman et al., 2007 [77]). Some examples of these uncertainties presented in literature (Abbaspour et al., 2007 [2]): effects of wetlands and reservoirs on hydrology and chemical transport, occurrences of landslides and large constructions (roads, dams, tunnels, bridges) affecting water quality and quantity, unknown wastewater dischargers in the water streams, agricultural activities, etc.

Monte Carlo simulation, first order error or approximation (FOE or FOA), Latin Hypercube (LH) simulation with constrained Monte Carlo simulation, mean value first order reliability, generalized likelihood uncertainty estimation (GLUE) are a few of the uncertainty analysis reported in the literature.

SUFI-2 (Sequential Uncertainty Fitting Version 2) is an uncertainty analysis algorithm based on a semi-automated inverse modelling procedure (Abbaspour et al., 2004 [1]). Using this algorithm, the uncertainty of the input parameters are depicted as uniform distributions, while model output uncertainty is quantified by the 95% prediction uncertainty (95PPU), calculated at

the 2.5% and 97.5% levels of the cumulative distribution of output variables obtained through Latin Hypercube sampling (Abbaspour et al., 2007 [2]). In this algorithm as parameter uncertainty increases, the model output uncertainty increases. If for a single parameter value we obtain a single model response, the propagation of uncertainty in a parameter leads to uncertainty in prediction described by 95PPU. The algorithm starts by assuming a large parameter uncertainty, meaning that the measure data initially falls within the 95PPU, then decreases the uncertainty in steps based on specified rules.

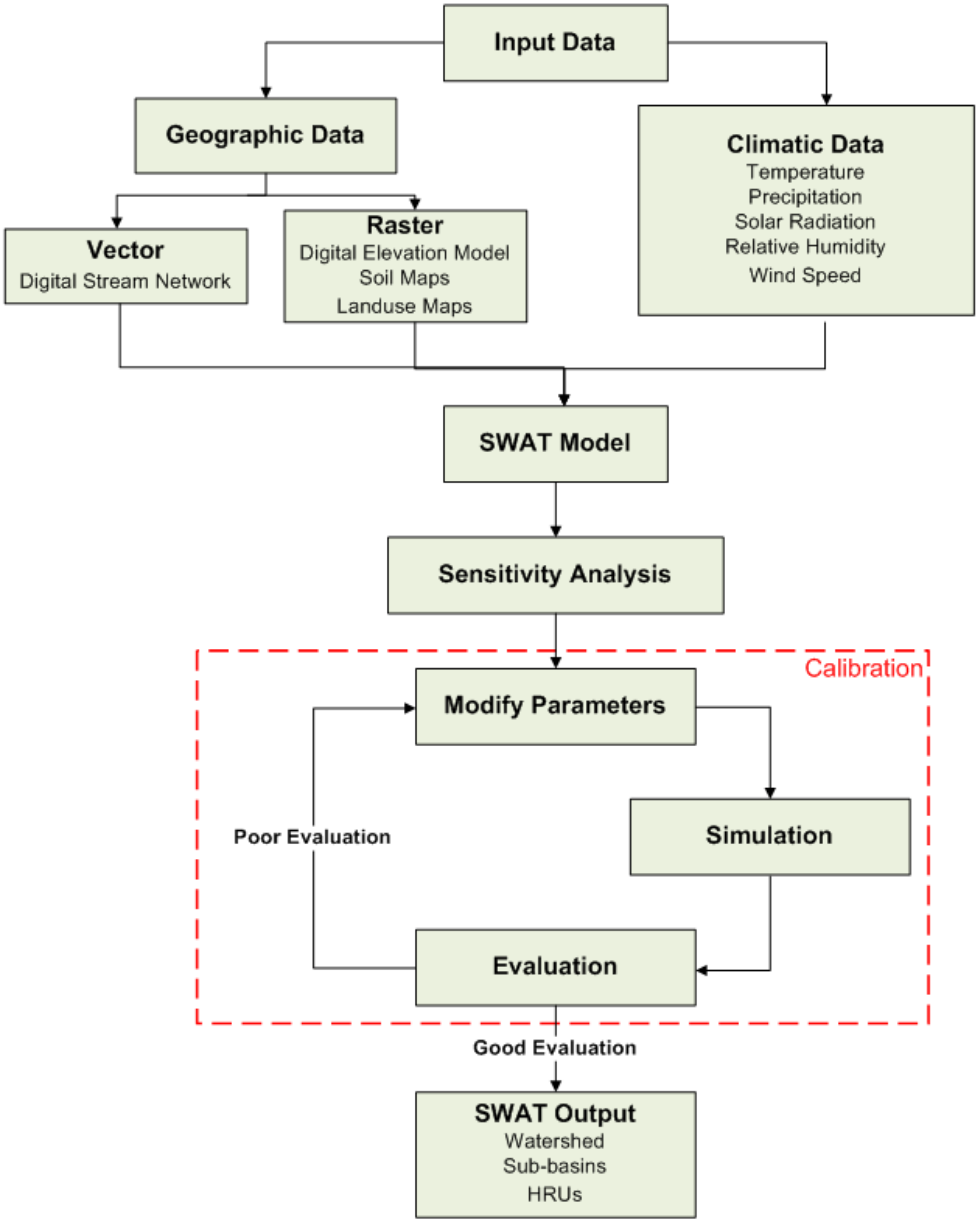

Figure 28: SWAT Flow Chart

Algorithm steps (Abbaspour et al., 2004 [1], Abbaspour et al., 2007 [2]):

1) Define an objective function. The final parameter ranges are always conditioned on the form of the objective function. A combination of different types of functions can be possible to yield a "multi-criteria" formulation.
2) Establish physically meaningful absolute minimum and maximum ranges for the parameters being optimized. The assumption is that all parameters are uniformly distributed within a region bounded by minimum and maximum values.
3) Optional (but highly recommended) "absolute sensitive analysis" for all parameters in the early stages of calibration (keep all parameters constant to realistic values while varying each parameter within the range assigned in step 1).
4) Initial uncertainty rages are assigned to parameters for the first round of Latin Hypercube (LH) sampling: b_j : [$b_{j,min}$ < /= b_j </= $b_{j,max}$], j=1,m. Appropriate ranges can be selected by using the sensitivity analysis in step 3.
5) Perform LH sampling and obtain n parameter combinations, n = number of desired simulations (approx. 500 - 1000). Run the simulation program n times and save the simulated output variables of interest, corresponding to the measurements.
6) Evaluate the simulations by calculating the objecting function g.
7) Evaluate each sampling round through a series of measures and identify the relative significance of each parameter b_i. This gives estimates of the average changes in the objective function resulting from changes in each parameter, while all others parameters are changing. We therefore have here a "relative sensitivity" based on linear approximation, which also depends on the ranges of the parameters (only provides partial information about the sensitivity of the objective function to model parameters). The ranking of sensitive parameters may change in every iteration.
8) Calculate measures for assessing the uncertainties. Calculate the 95PPU for all variables in the objective function by the 2.5% (X_L) and 97.5% (X_U) of the cumulative distribution of every simulated point. Assess the goodness of fit by the uncertainty measures.

9) Further sampling rounds with updated parameter ranges (as the parameter uncertainties are initially large). Each subsequent iteration will have narrower parameter ranges, while ensuring that the updated parameter ranges are always centered on the top p current best estimates (p is a user defined value). In the final step, parameters are ranked according to their sensitivities and highly correlated parameters are also identified. The highly correlated parameters with the smaller sensitivities should be fixed to the best estimates and remove from other sampling rounds.

Discretization of the watershed into HRUs results in the generation of a huge number of SWAT input files in ASCII format. Further on, each sub-basin needs four files to describe the sub-basin, weather, water use and water quality parameters. Each HRU needs six files to store information on chemistry, ground water, topography, management, routing, and soil properties (Abbaspour et al., 2007 [2]). The execution of SWAT hydrological models therefore involves a large set of input and output data, a high number of simulations, needed to be performed for model calibration to be able to obtain meaningful results, and a large number of parameters to be calibrated. Distributed hydrological models are especially difficult to calibrate because of different factors such as: time constrains, difficulties in parameterization, non-uniqueness and uncertainties in the conceptual model, model inputs, as well as lack of knowledge on parameters (Abbaspour et. al, 2010 [3]). All these challenges and issues imply the necessity of large storage and computational resources that can be offered by different distributed systems. This need is also implied by emergency environmental disasters, which need rapid modelling and analysis. The solutions in such cases is either to run the models with fewer simulations and obtain less than optimum solutions, or to run the models in large scale distributed systems to obtain meaningful and rapid results.

6.3.3. Black Sea Catchment Use Case

This use case was developed and partially executed in the frame of the enviroGRIDS project. The aim of the project was to build capacities in the Black Sea region to use new international standards to gather, store, distribute, analyze, visualize and disseminate information on past, present and future states of the region to be able to assess its sustainability and vulnerability.

The Black Sea Catchment region is a huge geographical area and a very complex environment. The development of a hydrological model for this watershed implies therefore highly interconnected and continuously evolving interactions at many spatial and temporal scales (Gorgan et al., 2013 [92]). Among the main objectives of the enviroGRIDS project was also the development of a high-resolution model for the entire Black Sea Basin (BSB) able to evaluate the impact of land use and climate change on the water resources. The development, calibration and execution of the SWAT hydrological model for the BSB region were successfully performed.

The Black Sea Basin has a total area of 2.3 million km^2 and drains rivers of 23 European and Asian countries (Albania, Austria, Belarus, Bosnia, Bulgaria, Croatia, Czech Republic, Georgia, Germany, Hungary, Italy, Macedonia, Moldova, Montenegro, Poland, Romania, Russia, Serbia, Slovakia, Slovenia, Switzerland, Turkey, and Ukraine) into the Black Sea, the most important rivers in the basin being Danube, Dnieper, Don, Kuban, Kizilirmak, and Sakarya. The Black Sea Basin is presented in Figure 29.

Figure 29: Illustration of Black Sea Basin

The sub-basins were delineated with a threshold area of 100 km^2, resulting 12,982 sub-basins and each of this was split into HRUs having unique combinations of slop, land use classes and soil types, resulting a number of 89,202 HRUs (Rouholahnejad et al., 2014 [201]). The simulation period was 1970-2006 using 3 years of initialization (1970-1972), 24 years for calibration (1973-1996) and 10 years for validation (1997-2006). Even though the model runs on a daily time step, monthly outputs were used for the calibration and validation of the model (build using SWAT2009).

Sensitivity analysis, calibration, validation, and uncertainty analysis were performed for water quantity, water quality, and crop yield (Rouholahnejad et al., 2014 [201]). SUFI-2 (Abbaspour et al., 2004 [1], Abbaspour et al., 2007 [2]) algorithm was used for calibration and uncertainty analysis. This is a tool used for sensitivity analysis, multi-site calibration and uncertainty analysis, capable of analyzing a large number of parameters and measured data from many gauging stations simultaneously offering therefore a high level of parallelization.

Using this algorithm, all sources of uncertainty are mapped to a set of parameter ranges. These ranges are initially assigned to calibrating parameters based on sensitivity analysis, knowledge and literature. After this, sets of Latin Hypercube (LH) samples are drawn from the parameter ranges and the objective function is calculated for each parameter set. The uncertainty is quantified using the 95% prediction uncertainty (99PPU). The goodness of the model performance is evaluated considering the calibration and the uncertainty level, based on some defined rules regarding on how good the measured data is (what percent it falls in a certain uncertainty band) and how well the model matches with the observations. Two measured are used at this step, referred to as P-factor and R-factor. The P-factor provides a measure of the model's ability to capture uncertainties (what percent of the measure data falls in the 95PPU uncertainty band). This index should have a value of 1 for perfect scenario. The R-factor is a measure of the quality of calibration and indicates the thickness of the 95PPU band. In an ideal case, this index is close to 0 (Abbaspour et al., 2007 [2], Rouholahnejad et al., 2014 [201]).

The model input and output files (which can easily be thousands of files for high resolution models such as the one for the BSB) are stored in a TxtInOut directory. All the initial files are copy in a first phase in the BACKUP directory. These files will remain unchanged during the calibration process to have in the end a reference.

SUFI-2 algorithm:

- Pre-processing phase (SUFI2_pre.sh) in which a Latin Hypercube (LH) procedure draws samples from the parameter ranges (SUFI2_LH_sample.exe). The parameter sets thus sampled are independent, allowing an easy parallelization. Theoretically all samples can be run in parallel, depending on the number of available computing resources. If all simulations were run in parallel, the theoretic time of an iteration would be the time of the longest simulation.
- Execution of SWAT_Edit.exe program, which copies the set of sampled parameters from par_val.txt into their appropriate locations in the SWAT input files. Depending on the number of model input files, this process can take a long time (open each file, make changes, save file, close file).
- Execute the SWAT model (SWAT2009.exe) and extract the outputs of interest from the SWAT output files (output.sub, output.rch, …)
- Post-processing phase (SUFI2_post.sh), in which a number of programs are executed:
    - SUFI2_goal_fn.exe – calculates the objective function. SUFI2 algorithm normally allows seven different functions (mean square error, summation and multiplicative forms of men square error, Chi square, Nash-Sutcliffe, weighted r^2, and ranked sum of square error aimed at fitting the frequency distributions (Rouholahnejad et al., 2014 [201])). Each function implies another set of results meaning that the final parameter ranges are always conditioned on the chosen objective function. The use of "multi-objective" functions is also possible, where different variables are included in the objective function.
    - SUFI_95ppu.exe – calculates the 95%prediction uncertainty. SUFI2 maps uncertainties on the parameters in the model by fitting all measurements in the 95PPU (making the uncertainty band as small as possible) (Rouholahnejad et al., 2014 [201]).
    - SUFI2_new_pars.exe – calculates updated parameters for the next iteration. The assumption is that all parameters are uniformly distributed within a region bounded by minimum and maximum values (defined by the user). The absolute parameter ranges should be as large as possible but

still physically meaningful (usually based on literature and professional judgment).

This process repeats until a satisfactory value of the objective function or model evaluation is achieved. The steps of this algorithm, as adapted from (Rouholahnejad et al., 2014 [201]), are illustrated in Figure 30. After the calibration is finished, the outputs file from each parallel processing directory are collected and merged in the SUFI2.OUT directory.

From a computational point of view, we can simplify the SWAT calibration process as the execution of a variable number of iterations till the calibration criteria is satisfied. Each such iteration consists of a number of independent simulations (as shown also in Figure 30). After each iteration, the user can change the chosen objective function to assess the impact of the function on the model's results. This simplified view is illustrated in Figure 31. The simplified steps executed within an iteration are:

- Pre-processing – this step is executed only once per iteration and at this phase, the parameters are generated within a defined range for each simulation, using LH algorithm. For each generated set of parameters, we have a corresponding simulation. In general, the number of simulations should be quite high, around 500-1000.
- Execution – actual model execution.
- Post-processing – This phase is executed at the end of the simulations and it's meant to process the output data for the next iteration.

For each model calibration, we need the following directory structure:

- Model inputs – all the model input files plus configuration files;
- Calibration outputs – holding the calibration outputs;
- BACKUP – a copy of the model inputs which is not affected during the calibration process;
- Executable – holds the model executables which are dependent on the model development.

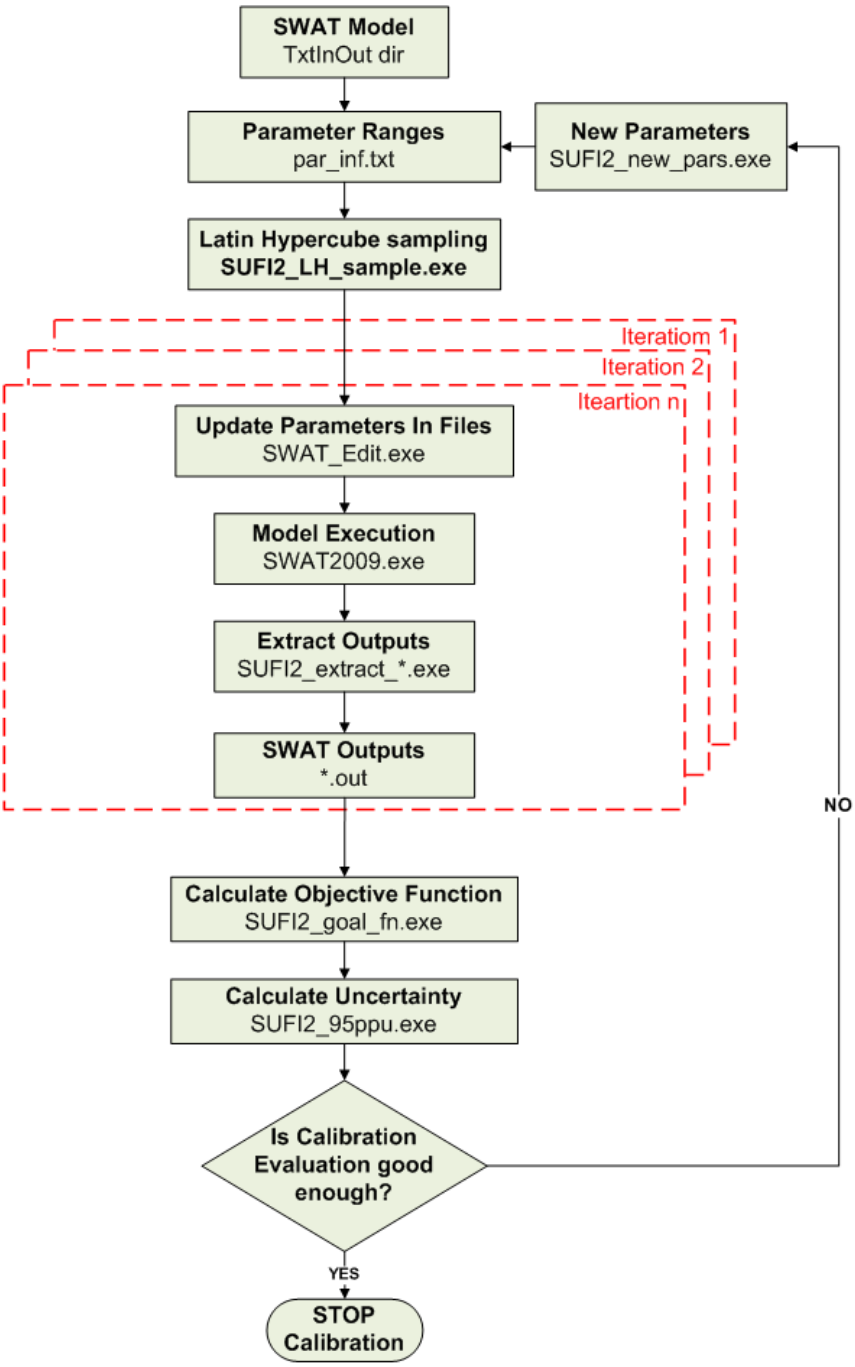

Figure 30: SWAT Calibration Flow

The calibrated model for the Black Sea Basin was used further on to calculate water resources component: blue water, green water flow, and green water storage, but also to assess the impact of land use and climate change on present and future water resources at high spatial and temporal resolution. The overall picture and detail conclusions can be found in final

enviroGRIDS reports as well as in (Rouholahnejad et al., 2014 [201]). The model outputs could be used to establish environmental goals, taking well-informed decision and develop monitoring strategies.

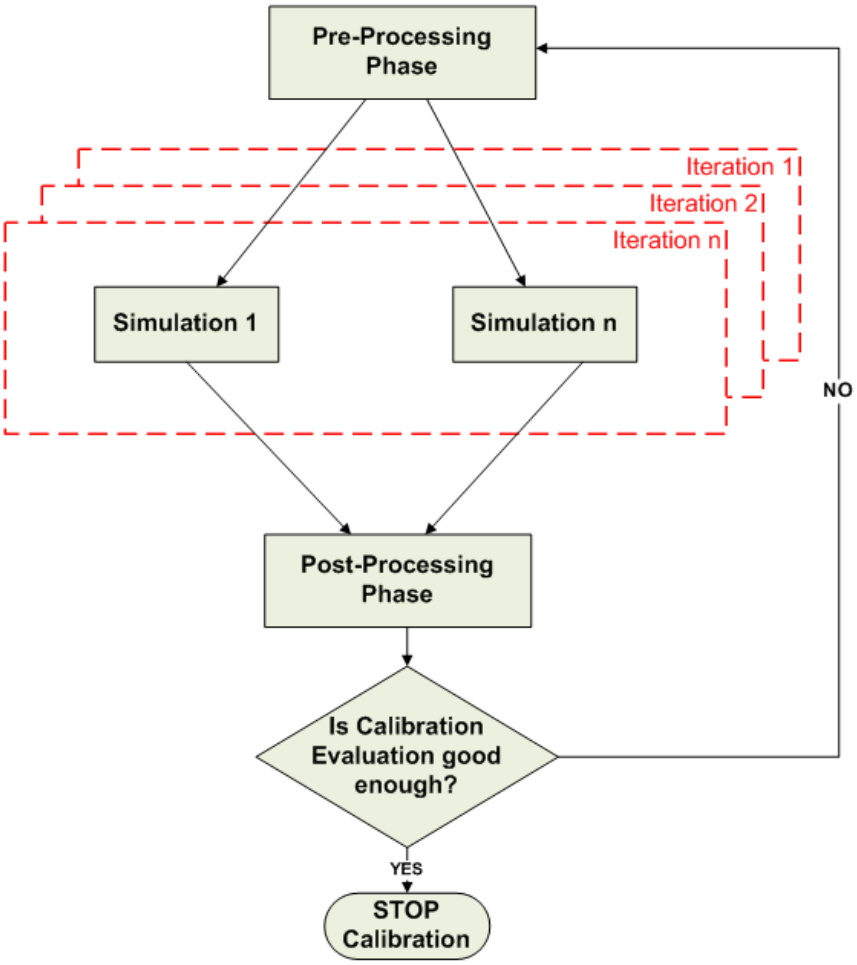

Figure 31: Simplified SWAT Calibration

6.4. gSWAT

gSWAT application (http://cgis.utcluj.ro/applications/gswat) was developed in the framework of enviroGRIDS project and allows the user to calibrate SWAT models in a flexible and interactive manner by taking advantage of the Grid infrastructure. The graphical user interface links the user to the Grid infrastructure in a flexible and transparent way. It uses intuitive user interaction techniques that allow different categories of users to use the application.

gSWAT is a Web application that may be used by specialists in calibration process of the SWAT models or by students who are learning the calibration process. For the calibration process, which is computational intensive and needs a lot of storage space, the Grid infrastructure offers a distributed execution environment and storage space as well for the outputs generated by SWAT.

6.5. GreenLand

The GreenLand (http://cgis.utcluj.ro/applications/greenland) is a Grid based application that offers scalability when dealing with large number of users and large processing data volume. The application has as main objective the generation and execution of workflows, based on satellite images, over the Grid infrastructure. In this case, each workflow could be represented as a DAG (Direct Acyclic Graph). The nodes of the graph could take the form of a simple operator or a sub-workflow. Multiple nested levels are allowed within the GreenLand application. Due to the computing and storage capabilities offered by the Grid infrastructure, the workflows execution times are significantly improved in comparison with standalone/cluster processing.

6.6. EDAP – Environmental Data Acquisition and Processing

6.6.1. Introduction

EDAP – Environmental Data Acquisition and Processing – is an environmental application, which was initiated in 2015 and it is still under development at UNEP/GRID-Geneva (United Nations Environmental Program - Global Resource Information Database – Geneva - http://www.grid.unep.ch). EDAP is used to automatically download environmental data (environmental variables) from different providers (World Bank – WB, Food and Agriculture organization of the United Nations – FAO, etc.) as well as from custom user input files, to process the data and to deliver it in different standard output formats. Currently, EDAP is used as an alternative way to provide data for UNEP Environmental Data Explorer – EDE (http://geodata.grid.unep.ch/). EDE is the authoritative source for data sets used by UNEP and its partners in the Global Environment Outlook report as well as other integrated environment assessments and it gives access to a broad collection of harmonized environmental and socio-economic statistical data at different aggregation levels, together with a considerable selection of

geospatial data (such as maps, graphs) usually at global and regional scales. Its online database holds more than 500 different variables (http://geodata.grid.unep.ch/extras/datasetlist.php), covering themes like Freshwater, Population, Forests, Emissions, Climate, Disasters, Health and GDP. EDE serves most of its data to the UNEP Live platform.

In the following we will present first a brief description of the EDE and the associated data flow, to give the context in which EDAP is integrated. After that we will present in detail the main objective of EDAP, the architecture and its components, the data flow and main functionalities.

6.6.2. Context – Environmental Data Explorer (EDE)

Environmental Data Explorer – EDE provides access to a large collection of harmonized environmental data and socio-economic statistical data sets, and this data are always furnished with associated metadata. EDE is not just a simple re-distributer of data (taken from different providers) but it also provided a value added data information.

The main features of EDE are (De Bono, 2016 [51]):

- Provides data at national level for 237 countries (when available) – these data are never modified and respect the original value provided by the original source;
- Makes available aggregated indicators at regional and global scale following a proven complex protocol in order to estimate the eventually missing values for a country and consequently allow a more realistic aggregation process;
- Follows main international standards (ISO19115, OGC);
- Human interaction: data are selected, checked by real operators;
- Continually updated to provide the latest data.

EDE manages selected environmentally oriented indicators from more than 60 different sources such as (in the order of importance and amount of data retrieved): FAO – FAOSTAT, World Bank, IEA, UNFCCC, JRC-PBL, UN Population, UNSD, FAO-Aquastat, CRED, FAO-FRA, OECD, UNESCO, WHO, FAO-Fishstat, etc. The data included in EDE are always retrieved directly from the original sources, to avoid any possible modification and to guarantee the most up to date information. In most of the cases, the data are downloaded manually from these data providers, they are formatted using different templates, depending on their original

structure and they are inserted into a PostgreSQL database, where they are processed automatically, using different SQL functions and procedures.

The schematic representation of the EDE data flow is presented in Figure 32 and has five main steps: 1) Retrieve data from the source (supervised or automatically); 2) Data processing environment; 3) Offline aggregations and data verification; 4) Publish data and metadata on EDE public server; and 5) Publish metadata on EDE dedicated GeoNetwork. The steps for which EDAP provides an automatic alternative way of retrieving and processing the data are highlighted with red and include basically step 1b and 2 i.e. automatic acquisitions of data from different providers (WB, FAO, etc.) or user predefined input files and automatic processing of the data.

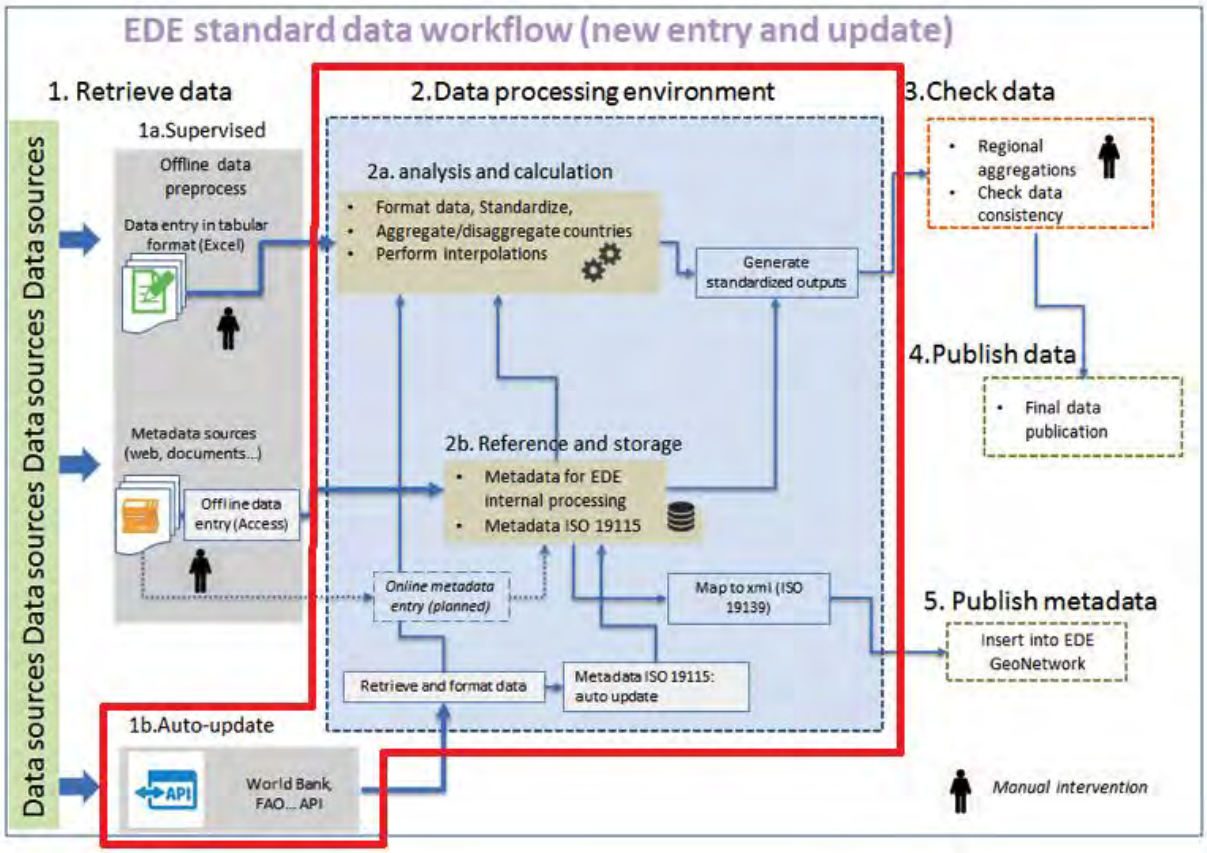

Figure 32: EDE Data Flow and EDAP (adapted from: De Bono, 2016)

6.6.3. Scope and Objectives

The EDE database contains an official list of 237 countries/territories, each associated with their related international acknowledged counties cods (ISO2, ISO3, UN, etc.). When retrieving the data (indicators) from different providers, we have to match the providers supplied standard code with the codes in our database. In some cases, unfortunately the providers do not have up to date standard codes and on top of this there is no general agreement on the countries and departments subdivisions (a country like France can incorporate or not overseas territories, depending on the provider). There are also situations in which different data providers have different measure units for data (indicators), requesting variable recalculation to appropriate standard units for a further processing. All these issues are usually treated manually by specialists at UNEP-GRID Geneva, making the data treatment process a time-consuming task and making the automation of data processing a complex procedure.

The main goal of EDAP is to offer a completely automated flow to access external data (coming from different providers), to integrate the data into the EDE system and to process it to get the meaningful information, which has to be presented in different formats and for different purposes. The main objectives of this project are to reduce as much as possible the human intervention while integrating the accumulated expertise in an automatic workflow for data management; to reduce also the time needed to update the data (indicators) and metadata present in the current EDE system, which serves the UNEP Live platform; to avoid human errors implied by processing large heterogeneous datasets; and to make the data management process accessible to non-specialists.

Currently there are several identified factors, which prevent or make the automation process difficult to achieve (De Bono, 2016 [51]):

- Most of the providers do not have a standard way to share their data i.e. they do not provide a specific Web service and or a specific data structure/format, making it difficult to develop automatic scripts to harvest their data.
- In the case the data providers actually have a Web service interface to download data, there is still no single protocol applied by all providers, and in most of the cases each provider uses its own style. This means that dedicated scripts have to be developed for each particular provider, reducing the code re-usage to the minimum. Some interesting initiatives such as Quandl (https://www.quandl.com/) got our attention. This library

contains modules (developed in Python, R, etc.) allowing unified, but size limited access to hundreds of databases. Although this library is rather oriented towards financial and economic data, they give access to several Open Databases, including some International Organizations such as UN Statistic Division, WHO, FAO, etc.

- The frequency of data update is different for each data providers and this is an important factor that has to be considered. If the data is updated several times during the year with irregular and not always predictable frequency, as in the case of the WDI (World Development Indicators) – World Bank, then automation becomes extremely useful. In case of infrequent updates (such as FAO-FRA which updates the data every 5 years) or indeterminate, the need of automation becomes less of a priority.
- An automatic update and processing only applies to already existing data.

Taking into account the above mentioned constrains and restriction, we have started the development of EDAP considering the most important two main data providers in EDE, i.e. World Bank and FAO. We also provide a third option for retrieving the data from user specific input files (which can be given in 3 different formats, based on the data structure). The third option was introduced dues to the frequent needs the specialists have on introducing particular mix of data sets (maybe collected from many different small data providers) in the system. In the future we also consider the usage of Quandl library to gain access more easily to data coming from small different other data providers.

6.6.4. EDAP System Architecture

The architecture of EDAP system is presented in Figure 33. We can distinguish 2 main components: Data Access and Data Processing.

The Data Access component contains the data providers, the EDE database and the API layer. Currently we have 2 data providers, WB and FAO, and a file system containing excel user input files. In the future we plan to add other smaller data providers into the system (based on the EDE requirements) and one possible solution, to avoid interacting with each separate provider, is to use Quandl, which is a tool delivering financial and economic data from hundreds of sources into different formats via different popular tools (excel, CSV, R, etc.) The API layer contains at the moment a specific connector (API) for each data source. This specific connector connects to

the data source (WB, FAO or the user input files), queries the required data based on specific protocols, handles data exceptions (missing countries, different county codes, missing values, etc.), fetches the data in different Python data structures and exports the data in a standard CSV file format. After this, the data is directly inserted in our PostgreSQL database. At this point, the data can enter the standard data processing flow, which is the same for all data. This process is described in detail in the next section.

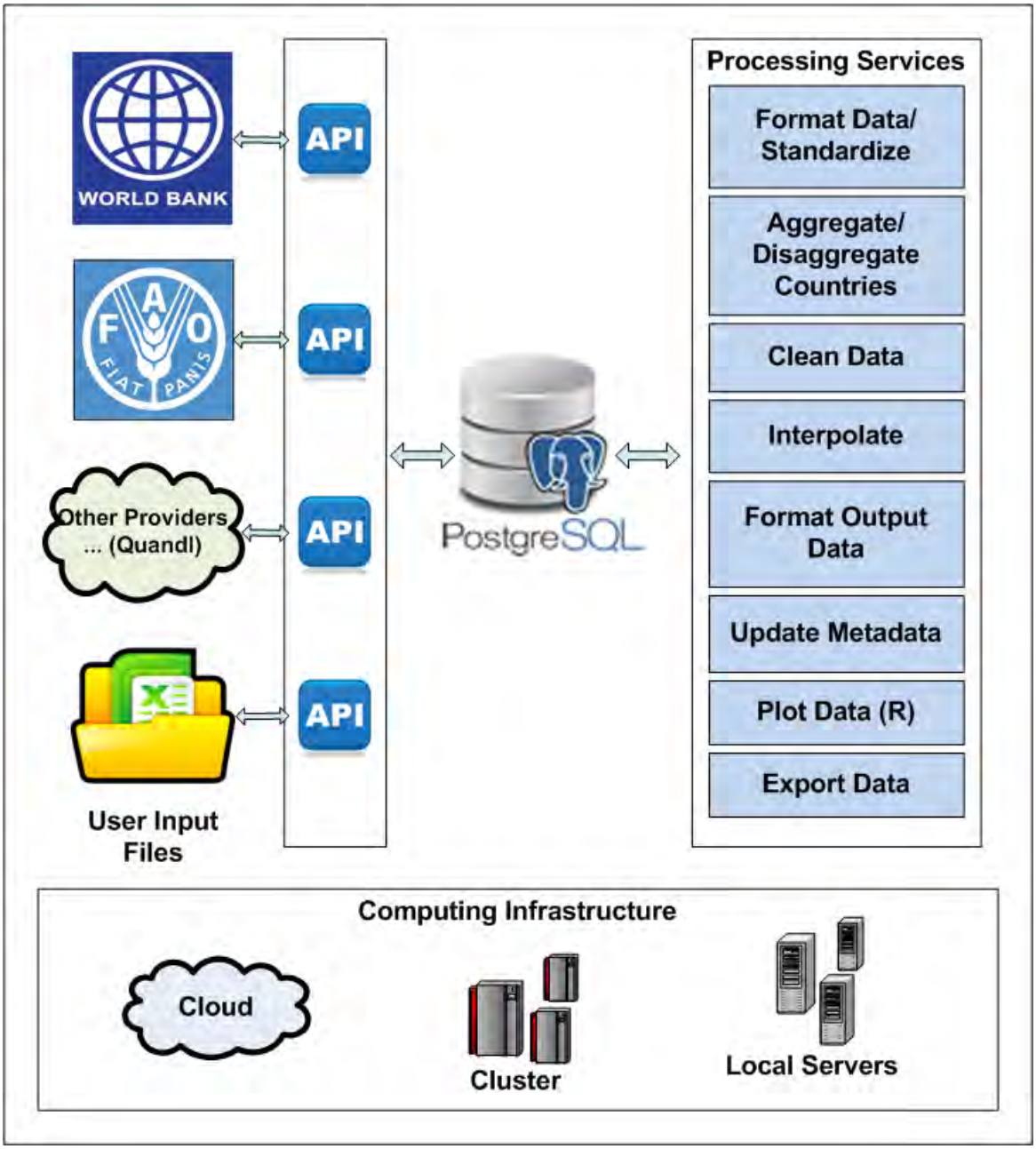

Figure 33: EDAP Architecture

The current version of EDAP is developed in Python and provides the user 3 different executions flows, based on the data provider: WB, FAO – FAOSTAT, and User Input File. The difference in the above-mentioned cases is the acquisition of the data.

In the case of WB, the application uses the wbdata (https://pypi.python.org/pypi/wbdata) python library to connect to the WB site and download the required data. The list of required indicators and required countries for which data is needed from WB is already set up in our database. To be able to request this data from WB some appropriate formatting of the data is needed. When requesting data from WB database, we take the list of available data and we match it with the list of required data. We only perform queries for the resulting set of data after matching.

In the case of FAO, we retrieve the data using the FAOSTAT API (http://fenixapps.fao.org/repository/api/) and REST web-services. Similar with the WB case, the list of required indicators, with their corresponding codes, and the list of required countries/regions are stored in our PostgreSQL database.

The third option given by the EDAP system is to insert the data automatically in the database, using predefined user input files. The system is capable to parse 3 types of excel custom input files and the difference between the 3 types of files is the display and the format of the data.

The Data Processing component incorporates a series of data processing services. These services are used in the data execution flow and offer functionalities such as:

- Format the data based on required standards;
- Handle countries exceptions (aggregation/disaggregation), which can occur depending on the data provider. Especially in the case of FAO, there are several known countries exceptions that have to be handled in the automatic data processing flow;
- Clean empty data entries in the database, which can appear either due to the lack of data from providers or from the lack of data in general. The system allows the user to choose in the configuration file the maximum number of empty data entries that are allowed for one year. If there are more empty values in the database than this number, the whole year is clean for that indicator;

- Interpolate data, currently based on two different interpolation algorithms, presented in the next section;
- Format the output data based on required standards and generate output tables both for initial data and interpolated data. At least one output table is generated for each required indicator;
- Update metadata, which is based on the ISO 19115 standard and has a series of attributes;
- The update process usually involves the modification of just a few attributes;
- Format and plot output data automatically using R. This step is needed to help the output data quality check by the specialists before inserting the output data in other execution flows or other systems (such as EDE);
- Export the output data (created indicator tables), using sql dumps, for integration with other systems (data sharing).

The two main components (Data Access and Data Processing) of EDAP systems are based on top of a computational environment, which can be formed from local server and/or cluster machines and/or a Cloud infrastructure.

In the next section we describe in detail the data processing flow and the associated steps data is following.

6.6.5. Data Processing Flow

Standardized formats for metadata are a must for automated production, processing and exchange of data between national and international organizations. The EDE indicators are described, cataloged and served using the ISO 19115 "Geographic Information – Metadata" standard, from ISO/TC 22. The main issue is that most of the data providers used in EDE do not apply this standard for metadata or any other standard for that matter. This implies, that in the case a new indicator is added in the EDE database, metadata are inserted manually and in most of the cases, the additional information has to be checked in provider's documentation, if available. This is another factor, which makes the automation of data a complex task. After the standard metadata is attached to data, the update of an existing indicator requires only the modification of a few metadata sections, which can be easily done within an automatic flow.

EDAP inherits the requirements and constrains imposed by EDE although it is an independed system and can be used to provide data for other systems too. Two EDE important constrains, applied also in the case of EDAP are:

- National data (values), supplied for 237 countries when available, are not modified in any way as they are released from the original data source. The system only ads, if they are not already available, standard names and county codes. Base on this national data, the system provides functionality to aggregate the values (indicators) at regional and global scale, following a proven complex protocol to estimate the eventually missing values for a country/region and to provide a more realistic aggregation process.
- Data always includes metadata, following the ISO19115 standard. In most of the cases, the metadata is missing and it has to be added by specialists.

Regardless the data source, once the data is in our database, the execution flow is the same for all the cases:

1) Gather and link all the metadata information;
2) Clean the empty fields;
3) Interpolate missing data (if possible);
4) Format data and create Indicators Tables (for initial data and for interpolated data);
5) Update metadata;
6) Plot data using R graphics;
7) Export the data.

6.6.5.1. Gather and link all the metadata information

When gathering the data in the table ALL, we perform a series of LEFT JOIN, having as main table the initial data insertion table. The LEFT JOIN operation will take all the data from the initial data insertion table and will add extra rows for all the extra matches in the LEFT JOINED tables. The extra rows in our case are added from the stat.cnty_completness_score tables because it matches 3 different completeness scores (in general: local, regional and global) for each data, depending on the aggregation type we can have (geo_agg_type). There are exceptions for some data, but in principles it triples the number of rows in the initial data insertion table (stat.data_file or stat.data_wb or stat.data_fao). If we have a huge number of

indicators (a large amount of data) the script can take a lot of time if we triple the nr. of rows in the database. As a solution, we give up 2 columns (geo_agg_type and compltn_score corresponding to each eggregation type) and we added 3 columns: compltn_score_local, compltn_score_regional and compltn_score_global. This way, we keep in the database only one row for each combination of (country, indicator and year) not 3. The scripts are more efficient and the efficiency is proportional with the amount of processed data.

6.6.5.2. Clean empty fields

This operation is done in the application to clean the database of the useless data. For the moment, the application allows the user to configure the level at which we what to clean the data by setting the CLEAN variables value in the configuration file (ede.cfg). This variable represents the minimum number of countries having value for a certain indicators per year, below which we delete the row in the database corresponding to that year and that indicator. This means that the cleaning algorithm will clean all the lines in the database corresponding to indicators and years for which we have values for less than a certain number (CLEAN) of countries.

6.6.5.3. Data Interpolation

Interpolation is a method used for estimating the value of a function between two known values. Linear interpolation is the estimation of a new value by connecting two known values with a straight line. If the known values are (x0, y0) and (x1, y1), then we have the following relation for point (x, y) that we want to interpolate:

$$(y-y0)/(x-x0) = (y1-y0)/(x1-x0)$$

This means that the y value for some point x is:

$$y = y0 + (y1-y0)* (x-x0)/(x1-x0)$$

This is the formula for linear interpolation in the interval (x0, x1), which can also be seen as a weighted average.

The interpolation algorithms used in EDAP is based on the COMPLETE variable, which can be adjusted in the configuration file. This variable represents the number of missing years (for an indicator and a country) still acceptable to interpolate. This means that if we have a gap with a number of years larger than COMPLETE, we leave it like that, no interpolation is done.

Currently we have 2 options of interpolation:

- Interpolate the values for the missing years and setting all missing values to the mean of the last known value and the next know value after the series of missing values.
- Linear interpolation: set the value of the missing years as a weighted average.

6.6.5.4. Format data and create Indicators Tables

We create a table for each indicator that we have in the database using the name specified in the metadata table. The tables are created both for original data and for the interpolated data.

6.6.5.5. Update metadata

The metadata is updated after the creation of the indicator tables and only for the indicators for which the creation of the resulting table was successful. The metadata is updated in the current date and an SQL file is also generated with the used update queries.

6.6.5.6. Plot data using R graphics

R graphics are generated for the resulting data (indicators table) as a way to check the consistency of the obtained data (especially the interpolated data)

6.6.5.7. Export the data

After the specialist has analyzed the consistency of the data in the resulting indicators tables (using the R graphics), the tables are exported (using sql dump) and ready to be integrated in the production database or in other data flows.

6.7. LiMES – Live Monitoring of Earth Surface

6.7.1. Scope and Objectives

Automatizing most of the processes for monitoring large amount of sites has become an essential goal considering the Information Age in which we live and the high velocity, variety and volume of data we experience. The main scope of LiMES project is precisely to automatize the processing of satellite imagery for monitoring landcover changes of several hundreds of sites per year (which can be chosen for example form the UNEP environmental hotspots, from the RAMSAR (RAMSAR [183]) sites, monitoring protected areas or SDGs area of interest). The

whole process of processing satellite images is a complex one and implies several intermediary steps: discovery, download, conversion of raw files into images, actual processing (computation of index, pan-sharpening, image corrections, etc.), and display (Figure 34). All these steps are automated but the main issue is related to the image selection and the quality check. The intention is to reduce to the minimum the human intervention and to facilitate as much as possible helping options for the user.

Currently we use satellite imagery coming from Landsat 5, 7 and 8 but in the future we also plan integration of other satellite data such as Sentinel 2.

LiMES capabilities:

- Allows a fast and easy update of the site through an automated workflow;
- Based on fully open source and OGC standards compliant components and technologies: Python, GRASS, Gdal, OWS, OpenLayer;
- Automatic Processing – based on Python and PyWPS;
- Scalability – allows the processing and monitoring of thousands of sites;
- Can be used for various thematics, such as: Hotspots monitoring, RAMSAR, Protected Areas, and at various geographical scales;
- Flexible – for easily adding other (satellite) data providers;
- Full transparency – all processes as well as all used images (and they reference) are available as metadata.

LiMES next challenges:

- Allow parallel processing of sites – the results will be considerable improved as the number of sites increases;
- Visual compatibility of images;
- Mosaicking;
- Cloud masking;
- Management of no-data in surface statistics;
- Display and result analysis filtered by date:
    - Year – to monitor seasonal variations;
    - Month – to monitor long-term variations.
- Allow more (guided) user interaction in image selection;
- Result Image Ranking by the users.

6.7.2. System Architecture

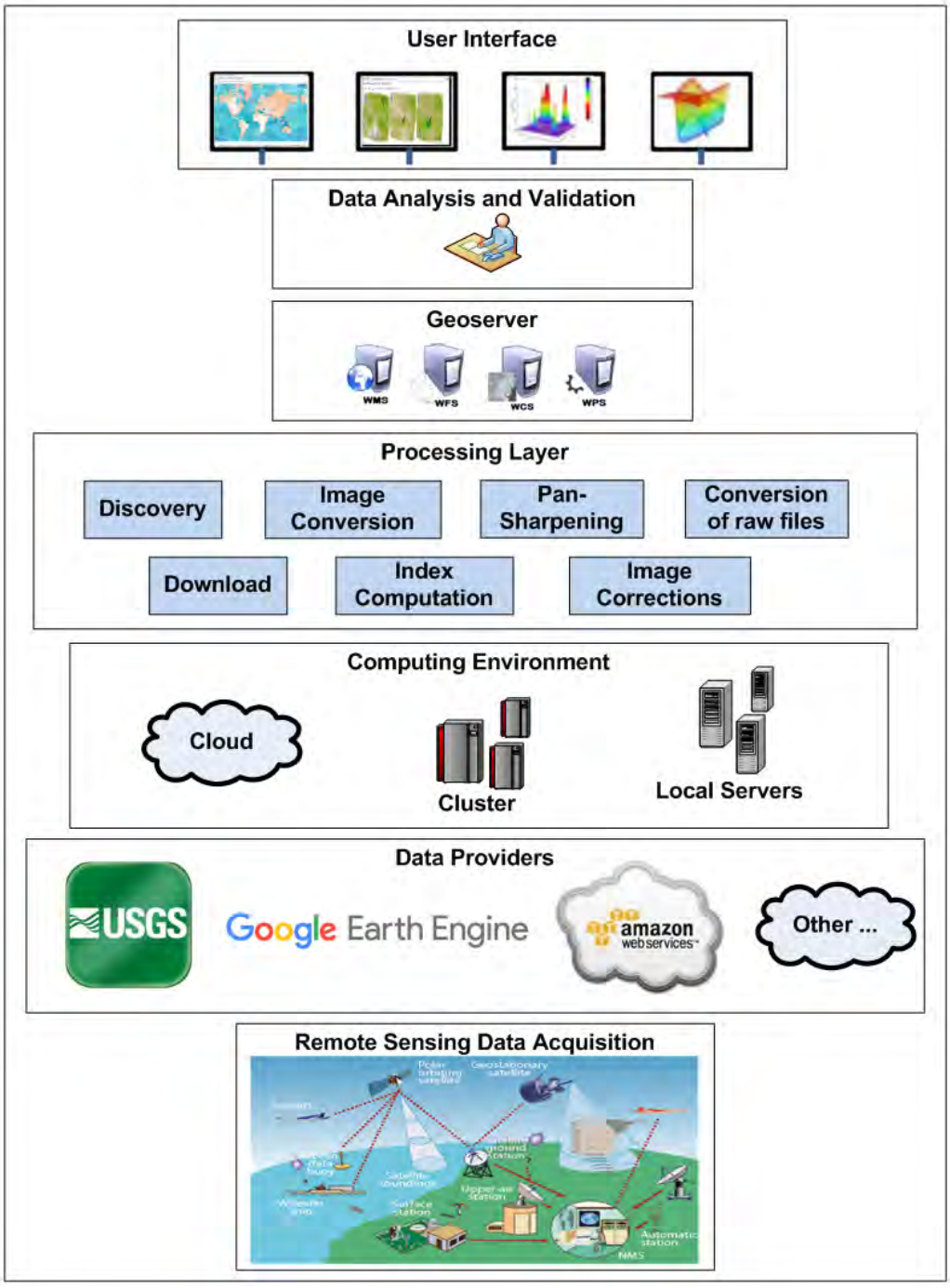

Figure 34: LiMES Architecture

The LiMES architecture is presented in Figure 34 and it's composed of 7 main components: Remote Sensing Data Acquisition, Data Providers, Computing Environment, Processing Layer, Geoserver component, Data Analysis and Validation, and User Interface.

The scope of LiMES is to hide the processing complexity as much as possible while giving the user a large degree of flexibility. All the functionalities in LiMES are offered as python scripts and exposed as WPS processes in a dedicated PyWPS instance installed on our server (UNEP/GRID-Geneva – UNIGE network). The WPS processes' results are either exposed automatically in Geoserver as WMS, WCS or they are further on integrated on the site using javascript and HTML. The publication of results in Geoserver is done automatically using REST API (gsconfig and curl libraries).

6.7.3. Execution Flow

The execution flow of LiMES is presented in Figure 37, where all the main steps of the flow are clearly emphasized and delineated: discovery, download, processing, validation and display.

Within the processing phase, a lot of operators can be applied on images. One example implementation is the calculation of Normalized Difference Vegetation Index (NDVI), following the formula:

$$\text{NDVI} = \frac{(nIR - \text{Re}\,d)}{(nIR + \text{Re}\,d)}$$

Figure 35: NDVI Formula

Where:

- NDVI – Normalized Difference Vegetation Index
- nIR – electromagnetic reflectance in Near Infra-Red (not equal to zero)
- Red – electromagnetic reflectance in Red (not equal to zero)

An example of NDVI calculation is illustrated in Figure 36.

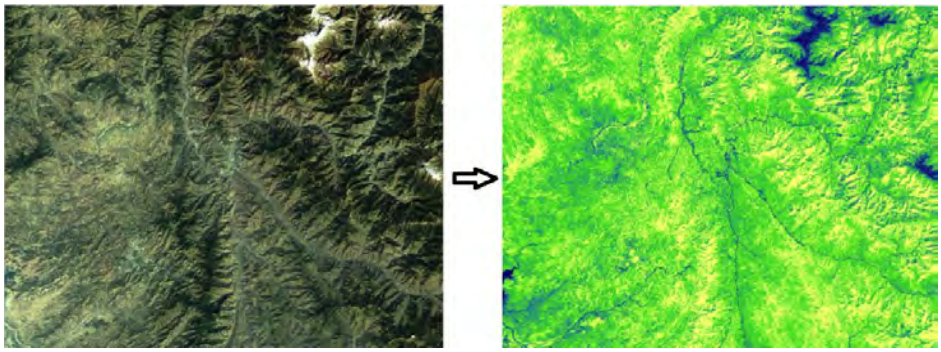

Figure 36: NDVI Example

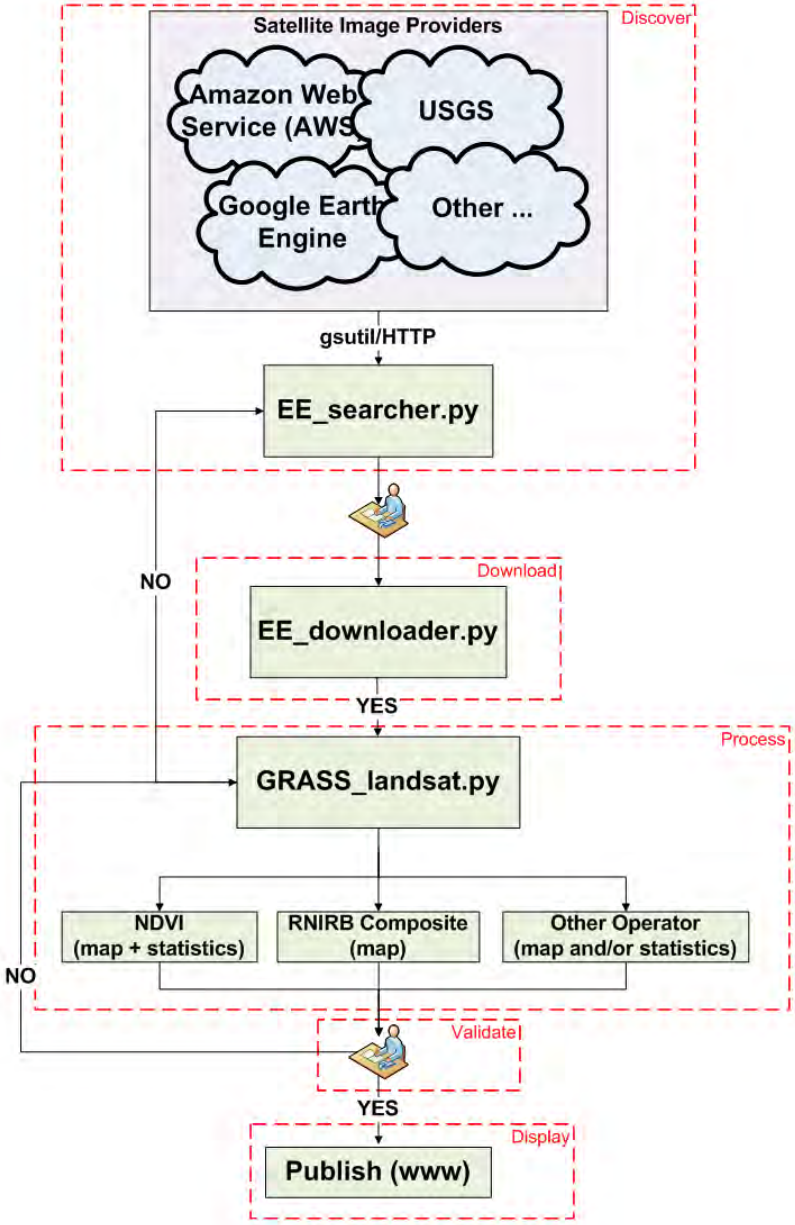

Figure 37: LiMES - Execution Flow

6.7.4. Use Cases

6.7.4.1. Site Overview

The LiMES site (Figure 38) is still under development but it already gives users different options for selecting existing monitoring area and to further apply different processing flows and visualize different details of that area.

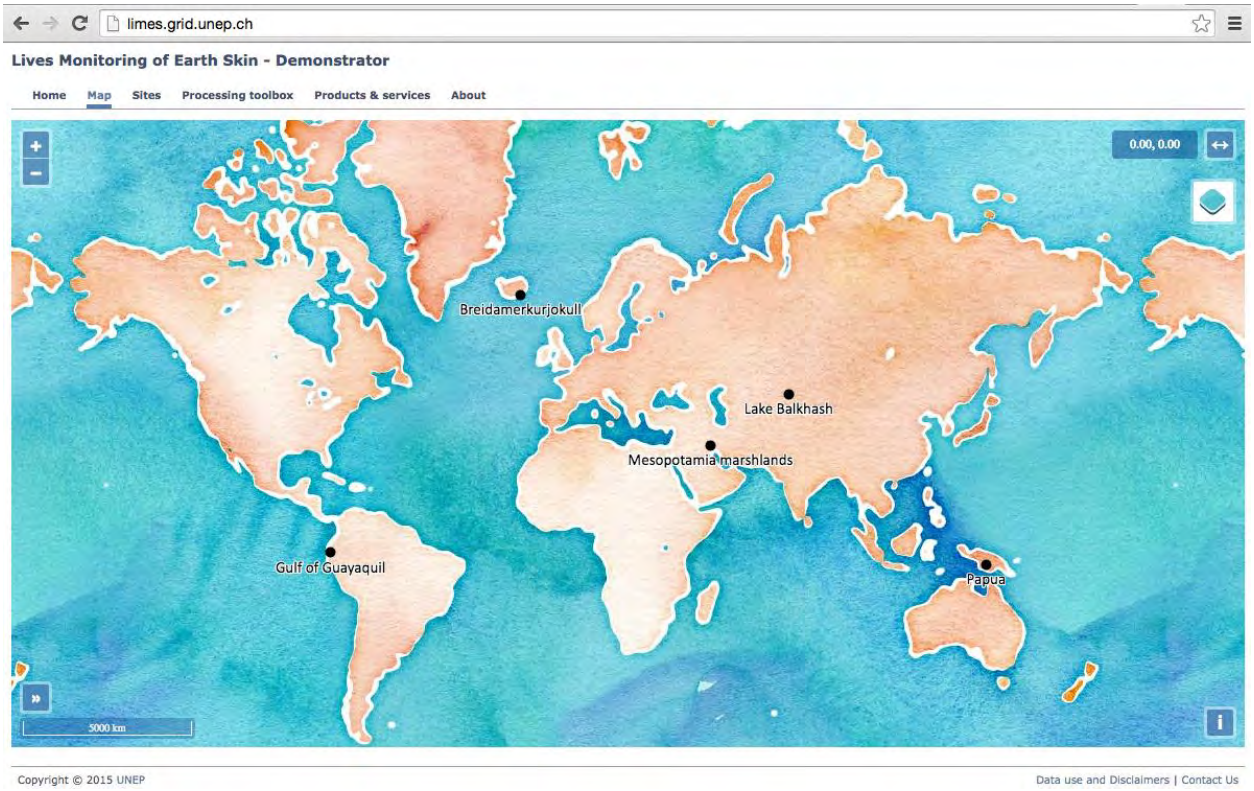

Figure 38: LiMES - Site Overview

Figure 39 presents an example of monitoring site – the area of Mesopotamian marshlands, one of the world great wetlands, covering an estimated area of 15 000 – 20 000 km^2 (5 792 – 7 722 square miles), located at the confluence of the Tigris and Euphrates rivers in Iraq. The marshlands are an important center of biodiversity, play a vital role in the intercontinental migration of birds, and have long supported unique human communities. Upstream damming as well as drainage activities in the marshlands themselves have significantly reduced the quantity of water entering the marshes. Together these factors have led to the collapse of the ecosystem. Re-flooding, by breaching of dykes and drainage canals has begun for restoring the marshlands

and as a result of these activities, vegetation and wildlife have returned to some parts of the marshes.

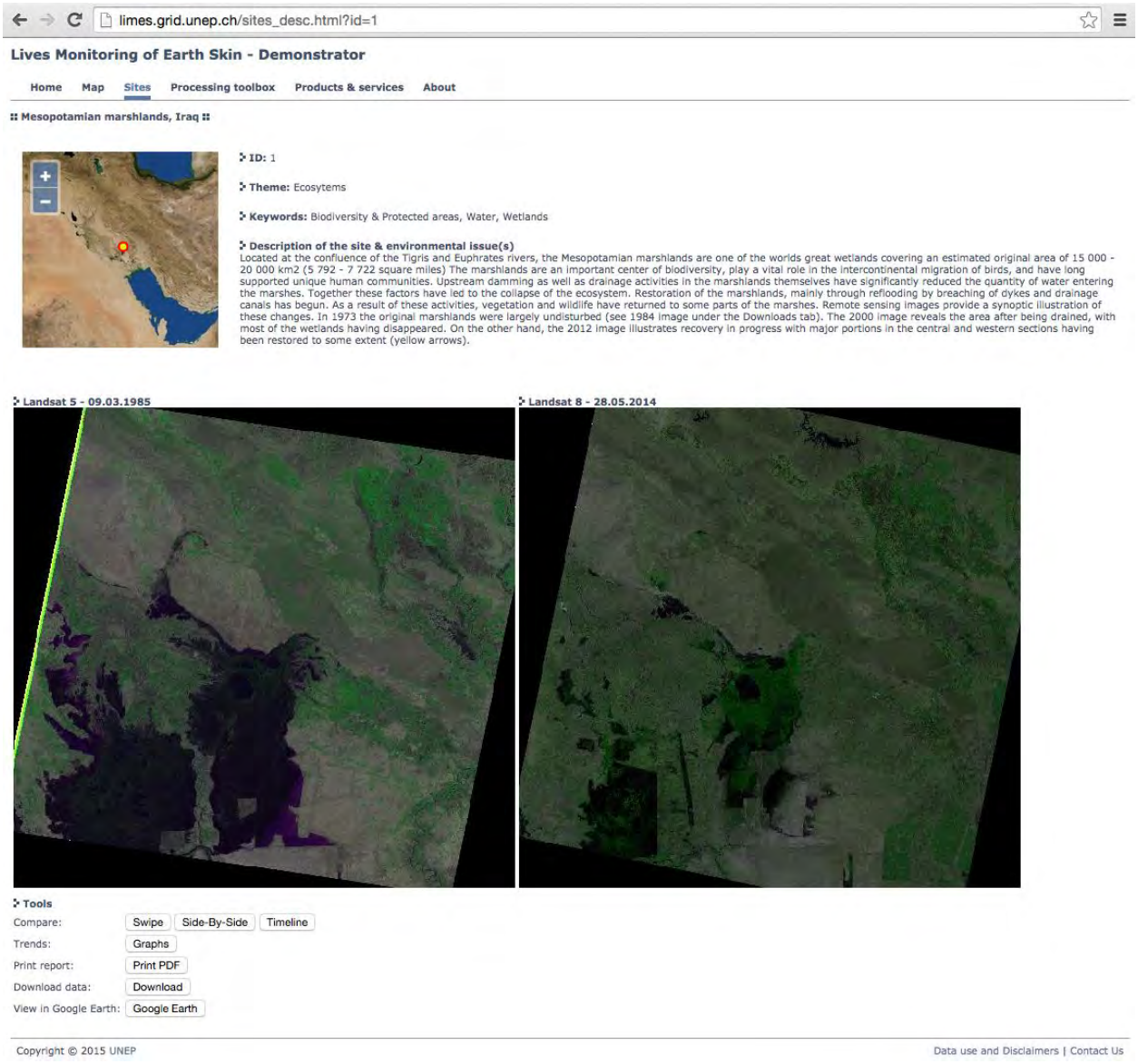

Figure 39: LiMES – Mesopotamian marshlands, Iraq

6.7.4.2. Side By Side Option

The Side by Side option gives users the possibility to visualize in comparison up to three images. The zoom is synchronized so that the users can navigate within the images and the corresponding area in the other images will be synchronized for an easy comparison of the status across time.

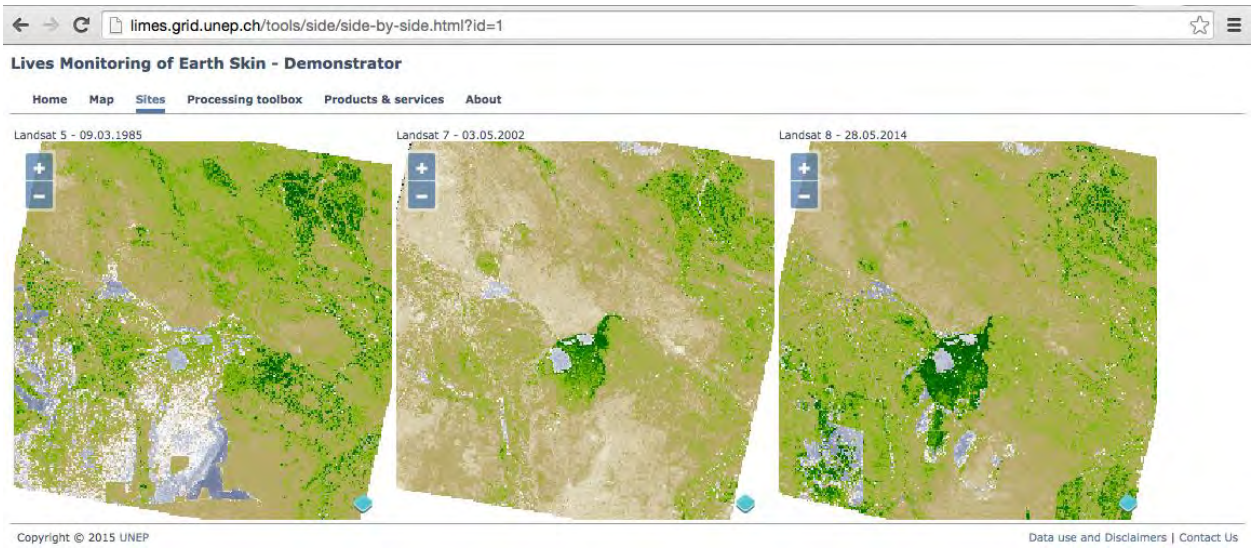

Figure 40: LiMES - Side by Side Option

Figure 40 presents the Side by Side option for Mesopotamian marshlands. The three remote sensing images provide an illustration of the changes in this area. In 1973 the original marshlands were largely undisturbed. In 2000 the area was drained, with most of the wetlands having disappeared. On the other hand, the 2014 image illustrates recovery in progress with major portions in the central and western sections having been restored to some extent (yellow arrows) due to restoration activities.

6.7.4.3. Swipe Option

The Swipe option shows the impacts of different factor over the selected site, comparing the first available image with the last available one.

Figure 41 and Figure 42 show the Swipe option of the same Mesopotamian marshlands area, site presented in previous sections. Using this option, the situation of this area (drained area and restoration activities) is even more emphasized over the time. In this case, only the differences between the past and the current situation are shown, no intermediary steps like in the case of the side by side option.

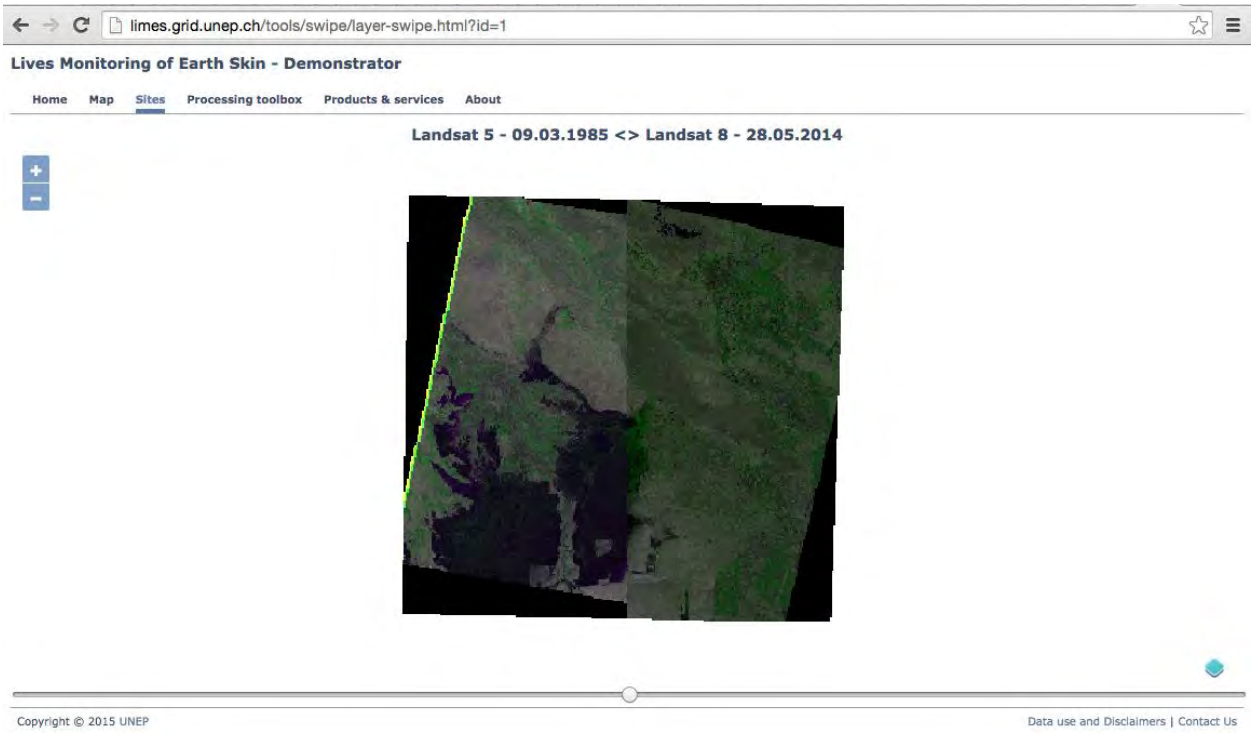

Figure 41: LiMES - Swipe Option

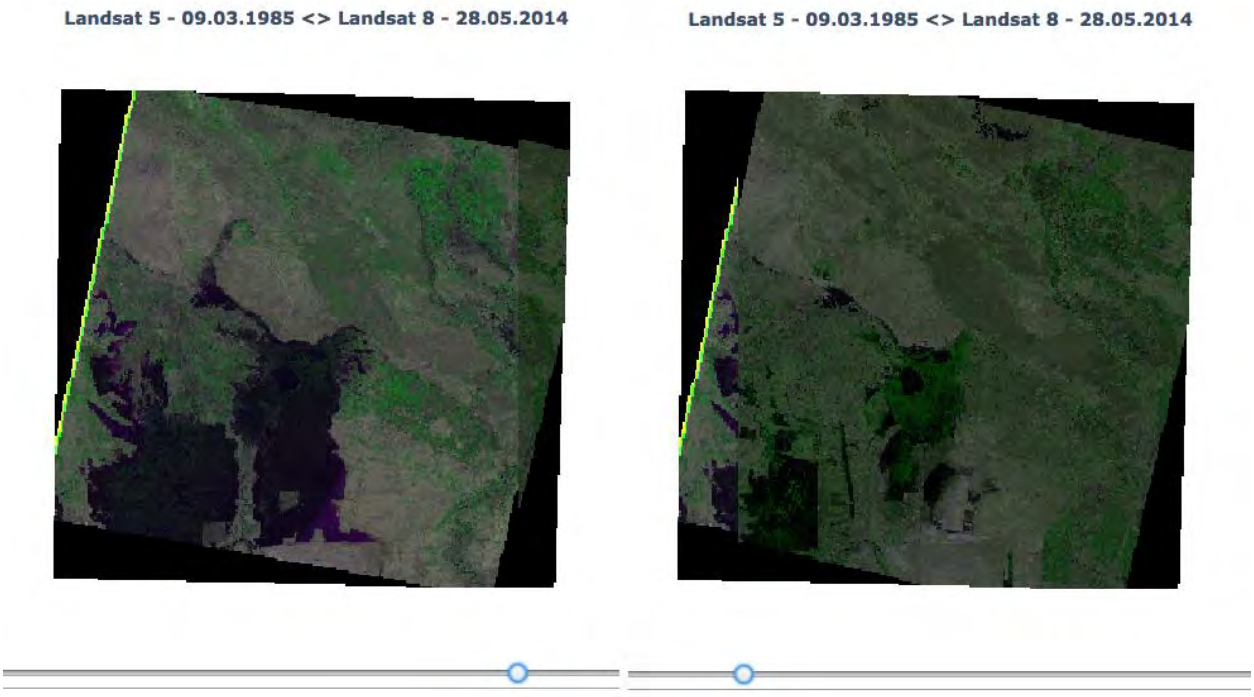

Figure 42: LiMES - Swipe Option – Comparison

6.7.5. Conclusions

The potential applications of LiMES are multiple and include: regularly updating the UNEP hotspots, monitoring the status of RAMSAR and other protected areas, where landcover changes monitoring is required, supporting the development of SDGs for a sustainable development, etc. As the satellite images in LiMES can be served using OGC image standards geospatial services, LiMES can be easily integrated with any interoperable platform that follows the OGC standards, including UNEP-Live.

6.8. Global Flood Model

A streamflow simulation framework, formed by the coupling of a Global Climate Model (EC-Earth) with a hydrological model (Continuum) was used to produce 140 years of streamflow time series on a large number of stations, all over the world (in all the 5 continents) (Silvestro et al., 2016 [215]). The scenario includes a current climate time window (from 1950 to 2012) and a future climate time window (from 2013 to 2094), having a spatial resolution of 1.125 and a time resolution of 3 hours. The time series were than post-processed to improve the knowledge about the impact of a possible climate change on streamflow extreme values.

6.8.1. Continuum Hydrological Model

Continuum (Silvestro et al., 2013 [213], Silvestro et al., 2015 [214]) is a distributed and continuous hydrological model, which aims at balancing the necessity for a complete description of physical processes with the goal of avoiding over-parameterization. This means that a special attention is given to reducing as much as possible the parameterization of the physical processes (so that land information can be extensively used as a constraint to parameter calibration) but in the same time, the model indents to maintain the necessary details of all the terms of the hydrological cycle. The model was designed to be implemented in different contexts but especially on data-scarce environments (with no stream flow data). All the main hydrological phenomena are modeled in a distributed way.

The Continuum model requires five meteorological variables as input: rain, temperature, solar short wave radiation, air relative humidity, and wind speed. These inputs are produced by the EC-Earth model (Moss et al., 2010 [155]), a state-of-the-art Global Climate Model (GCM), developed in the framework of the European Consortium EC-Earth. The EC-Earth model

simulates climate in the period 1960–2012, using reconstructed historical anthropogenic forcings and solar variability, and creates three scenarios for the period 2006–2094, based on the three representative concentration pathways (RCPs) for anthropogenic emissions (Silvestro et al., 2016 [215]). The five input variables generated by the EC-Earth model need to be downscaled (using different techniques, based on the variable) to be further used in the hydrological model from 1.125 deg (EC-Earth) to 0.083 deg (Continuum) spatial resolution. The temporal downscaling was not considered because the EC-Earth output already has a 3 hours fine resolution.

A basin is represented using a regular square mesh based on a DEM, the flow directions are identified based on the directions of maximum slope derived by the DEM and the drainage network considers a representation, which distinguishes between hillslope and channeled flow and it is able to describe hydrodynamic and morphological conditions. Infiltration and subsurface flow exploit land use information and climatology to set the infiltration parameters and the overland runoff is distributed with differentiation between hillslope and channel flow (Silvestro et al., 2016 [215]).

6.8.2. World Scale Scenario

Considering the world scale of the application, the estimation of the model parameters does not allow a detailed calibration for each basin and for each section. Nevertheless, for such a scenario, the objective is not to reproduce with high precision all the characteristics of the hydrographs but rather to estimate the annual maxima daily streamflow (AMDS). The calibration of the hydrological model was done using stations where at least 15 years of daily data were available and the analysis was carried out on a larger number of sections, including these where only monthly streamflow data were available. A detail description of the hydrological model calibration is given by Silvestro et al. (2016) [215].

Once the model was calibrated, a unique continuous simulation was run from 1960 to 2094, generating a sample of 134 AMDS for each station. To be able to estimate possible differences between current and future climate of a given time period, the entire period was divided in three periods:

- Current historical climate HC (1960 - 2012);
- Near future climate NF (2010 - 2060);
- Far future climate FF (2044 - 2094);

In such a large scale scenario, we can find a large number of sources of uncertainty related to the elements of the chai used to generate the streamflow time series, such as: the GCM output, the downscaling procedure, the hydrological model structure and parameterization, etc. All these factors make the statistical analysis and the estimation process about a trend at global scale even more complex. Several approaches to reduce these uncertainties are presented in (Silvestro et al., 2016 [215]).

Model Specifications:

- Developed by CIMA Research Foundation – International Center on Environmental Monitoring (CIMA [42]);
- Large use case: Flood Model GAR (Global Assessment Report);
- 2 main procedures to execute:
    - Generation of a hydrological model input, starting from the output of a climatic model;
    - Execution of the Continuum model.
- 30 domains;
- 1st procedure – 1 year – one domain -> 1.4 GB;
- 30 domains – 150/185 years – 6,5/7,8 TB + output data 1-2 TB.

6.9. Personal Contributions

- Development of a new environmental application (Environmental Data Acquisition and Processing – EDAP).
- Contributions to the development of different functional environmental applications mostly in the remote sensing and hydrological fields: gProcess, gSWAT, GreenLand, LiMES. The personal contributions are mainly oriented to:
    - Overall system architecture;
    - Processing components, using different geospatial technologies;
    - Scheduling and execution on different computing infrastructures;
    - Database design and implementation.
- Analyzing and working with two environmental applications (hydrological models SWAT and GFM - Continuum) in different projects.

- Published Papers:
  - *Giuliani, G., Dao, H., De Bono, A., Chatenoux, B., Allenbach, K., De Laborie, P.,* ***Rodila, D.,*** *Alexandris, N., and Peduzzi, P. (2016). Live Monitoring of Earth Surface (LiMES): a framework for monitoring environmental changes from Earth Observations, (to be submitted).*
  - *Silvestro, F., Campo, L., Rudari, R., De Angeli, S., D'Andrea, M.,* ***Rodila, D.,*** *and Gabellani, S. (2016). Impacts of EC-Earth Global Climate Model RCP4.5 climate change scenario on maximum daily streamflow quantiles at global scale, Journal of Climate (submitted article).*
  - ***Rodila, D.,*** *Bacu, V., and Gorgan, D. (2012). Comparative Parallel Execution of SWAT Hydrological Model on Multicore and Grid Architectures, in International Journal of Web and Grid Services (IJWGS), Vol. 8/3, September 2012, pp.304 – 320, http://dx.doi.org/10.1504/IJWGS.2012.049172.*
  - *Bacu, V., Mihon, D., Stefanut, T.,* ***Rodila, D.,*** *Abbaspour, K., Rouholahnejad, E., and Gorgan, D. (2013). Calibration of SWAT Hydrological Models in a Distributed Environment Using the gSWAT Application, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 66–74, ISSN 2158-107X.*
  - *Gorgan, D., Bacu, V., Mihon, D.,* ***Rodila, D.,*** *Abbaspour, K., and Rouholahnejad, E. (2012). Grid based calibration of SWAT hydrological models, in Journal of Nat. Hazards Earth Syst. Sci., Vol. 12/7, pp. 2411-2423, http://dx.doi.org/10.5194/nhess-12-2411-2012.*
  - *Mihon, D., Bacu, V.,* ***Rodila, D.,*** *Stefanut, T., Abbaspour, K., Rouholahnejad, E., and Gorgan, D. (2012). Grid Based Hydrologic Model Calibration and Execution, Chapter in the book: Advanced in Intelligent Control Systems and Computer Science, Dumitrache I. (Ed.), Springer-Verlag, Vol. 187, pp. 279-293, ISBN 978-3-642-32548-9, http://dx.doi.org/10.1007/978-3-642-32548-9_20.*
  - ***Rodila, D.,*** *Bacu, V., and Gorgan, D. (2012). Geospatial Applications on Different Parallel and Distributed Systems in enviroGRIDS Project, in European Geosciences Union - General Assembly EGU 2012, Vienna, Austria, April 22-27, 2012, (abstract).*

- ***Rodila, D.,*** *Bacu, V., and Gorgan, D. (2011). Comparative Analysis of Distributed and Grid Based Execution of SWAT Model, in 3PGCIC 2011 - Sixth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Barcelona, Spain, October 26-28, 2011, pp. 273-278, http://dx.doi.org/10.1109/3PGCIC.2011.49.*
- *Bacu, V., Mihon, D., **Rodila, D.,** Stefanut, T., and Gorgan, D. (2011). Grid Based Architectural Components for SWAT Model Calibration, in HPCS 2011 - International Conference on High Performance Computing and Simulation, Istanbul, Turkey, July 4-8, pp. 193-198, ISBN 978-1-61284-381-0.*

Chapter 7: Porting Environmental Applications to Parallel and Distributed Infrastructures

7.1. The Need of Parallel and Distributed Infrastructures in Environmental Sciences

The urgent need of Environmental Sciences for large storing and computational resources to cope with the immense challenges of managing voluminous environmental data and complex, large-scale environmental applications is emphasized by the numerous environmental initiatives to make use of parallel and distributed infrastructures. In the following we will present some of these initiatives.

The British Geological Survey has initiated the Environmental Virtual Laboratory (http://www.evo-uk.org/) (Vitolo et al., 2015 [230]) project and it is exploring the provisioning of data services, Web-enabled environmental models, and a suite of on-line local community tools in the spirit of the Cloud paradigm of software as a service.

The US Department of Agriculture Natural Resources Conservation Service is developing the Cloud Services Innovation Platform (Lloyd et al., 2012 [137]) to offer data and modeling services for use in the field.

The US Environmental Protection Agency has developed the WATERS (Watershed Assessment, Tracking and Environmental Results System) (WATERS [237]) program that provides services that perform various data services and related analysis like watershed delineation.

These applications suggest a certain momentum in the community toward moving more of the tasks needed to support environmental sustainability, such as running environmental simulation models or large databases, to remote computer servers on the Cloud rather than on PCs (Laniak et al., 2013 [130]).

Studies such as (Di et al., 2003 [53]; Muresan et al., 2006 [157], Gorgan et al., 2012 [91], Colceriu et al., 2013 [43], Sun et al., 2013 [219]) applied a successful approach to extend Grid computing to the remote sensing community and to make OGC Web services Grid-enabled. These studies considered that the Grid has a great potential for geospatial disciplines.

Gridification of Earth Science applications has been a concern in the FP7 project enviroGRIDS (http://www.envirogrids.net/). A main goal of this project was the gridification of the Black Sea catchment to support its sustainable development through the development of a Grid enabled Spatial Data Infrastructure (gSDI). This goal was achieved through the integration into the Grid infrastructure of different services and tools among which we can mention the SWAT hydrological model and the Geospatial Web services – OGC.

Yang et al. (2011 [248], 2013 [249]) reviewed research results in Cloud Computing addressing particular needs in geospatial and Digital Earth applications (Goodchild et al., 2012 [89]). The conclusion was that the combination of service oriented architecture with Cloud computing faces challenges of storage and scalability in a global context, reflecting the community's need to manage and process large scale amount of observations and processes data from environmental models and other services.

Wang et al. (2013) [236] examined remote sensing processing methods and strategies for large-scale meteorological monitoring in real-time and natural disaster warning scenarios. The conclusion was that Cloud computing is effective especially when the management of huge amounts of data and distributed parallel processing are essential requirements.

Fustes et al. (2014) [74] described the usage of Cloud computing resources for data and processing intensive marine applications, such as the detection and localization of marine spills using remote sensing methods and advanced segmentation algorithms.

Karmas et al. (2015) [118] proposed a scalable geospatial platform for the online and real-time harvesting of valuable information from big Earth Observation (EO) data. Their datasets are stored and pre-processed automatically on their hardware. They have been tested and validated the efficiency and automatically processing of high-resolution satellite data for different geospatial, environmental, agriculture, and water engineering applications. The authors also propose an automatic data acquisition and pre-processing component in which Landsat 8 raw data are downloaded, stored, and pre-processed automatically. These steps are done through a series of python scripts, which control, facilitate and automate the entire operation. They check the Landsat 8 archive for any newly acquired dataset and download the ones corresponding to the interest area. Within this work, the authors emphasize the importance of automatizing both data acquisition and processing of Landsat data in real time harvesting of valuable information of the Earth system.

Lee and Kang (2015) [133] describe the challenges and the opportunities of Geospatial Big Data. They also present a system architecture, developed with the support of the Ministry of Land, Infrastructure, and Transport of the Korean Government, on interactive analytics for real-time or dynamic geospatial Big Data. The presented system consists of three layers: geospatial Big Data integration and management, geospatial Big Data analytics, and geospatial Big Data service platform. Distributed computing is used for retrieving and filtering the geospatial Big Data coming from various sources (satellites, drones, and vehicles, geospatial networking service, mobile devices and cameras) as well as for further analysis.

Similar recent initiatives to analyze and process geospatial Big Data using distributed computing exist in literature.

Kramer and Senner (2015) [125] propose a software architecture to process large geospatial data sets in the Cloud, using multiple algorithm design paradigms such as MapReduce, in-memory computing or agent base programming. The system is based on workflow executions, which are described in a domain specific language (DSL), parsed, interpreted and executed through a processing chain on a given Cloud infrastructure, using a scalable and fault tolerant distributed file system and respecting constrains defined by the user. In this case, the Cloud infrastructure is also used for storing and distributing large data sets but also for processing large volumes of geospatial data.

Geospatial Big Data challenges are also described by Nativi et al. (2015) [160] within the context of GEOSS (Global Earth Observation System of Systems) and particularly its common digital infrastructure. The authors introduce a fully brokering approach implementation (GEO DAB, already described in 3.6.3) building on Cloud computing technologies. Their solution is based on a hybrid Cloud deployment model, composed from a few distinct private and public Cloud infrastructures that remain unique entities but are bound together to enable data and application portability.

All these initiatives, research works, and projects emphasize the urgent need of parallel and distributed commuting for environmental services and application, to boost productivity and achieve higher performances towards a sustainable environment. Granell et al. (2016) [95] also emphasizes a wide use of standard-based web service implementations to support the development of flexible and dynamic services for environmental modelling applications. The

service abstraction can successfully enable ready to use, on demand services that can easily use Cloud computing.

7.2. Challenges

Based on the literature review and our practical experience, we have identified a list of challenges in porting environmental application on different parallel and distributed infrastructures and solving aspects of the interoperability between Environmental Sciences and Computer Science areas:

- Environmental application particularities and specificity;
- Technical understanding of an environmental application: structure, execution and data flow, data management, etc.;
- Application parallelism: task, data, mixed, thread level;
- Application type: legacy code, stand-alone, distributed, etc.
- Application complexity:
- Processes complexity;
- Data volume, heterogeneity and complexity;
- Application security.

7.3. Experiments

The enviroGRIDS project (http://envirogrids.net/) is a FP7 research project that aimed to develop a Spatial Data Infrastructure (SDI), targeting the Black Sea Catchment region, able to store, analyze, process and visualize obtained data while performing distributed simulations of environmental changes. These simulations had as purpose the assessment of the sustainability and vulnerability in the interest region and consist in executing different scenarios. To achieve all this, modeling the Black Sea catchment was necessary using high resolution data and a performant hydrological model – SWAT (Soil and Water Assessment Tool), described in section 6.3.

The calibration of large SWAT models involves a large set of input and output data and a high number of simulations, which requires a set of basic requirements. Decision makers also need to obtain near real time output from SWAT models to be able to make reliable and

meaningful predictions and due to this time constrain, most of the hydrological models are executed with fewer function calls, which is reflected in the not so precise obtained results.

In the following, we will focus on the calibration of different instances of SWAT model, which have been performed in the framework of the enviroGRIDS project.

7.3.1. SWAT Calibration Execution on Grid – gLite Middleware vs. Multicore (UTCN)

The following experiments were performed on the Grid infrastructure (gLite middleware) installed at the Technical University of Cluj-Napoca (UTCN). We performed a large number of SWAT simulations (within the calibration process), with different parameter sets, on the same large-scale model used by Rouholahnejad et al. (2011) [200]. Based on this, we have made a comparative analysis of the speedup obtained by a Grid infrastructure compared to a Multicore environment, on the same application model. The calibration was submitted on the Grid directly from command line, without using any interfaces, by using only some well-defined scripts. The goal of these experiments was not to present the calibration of the SWAT models on the Grid as a user-friendly application but to underline the advantages and the capabilities offered by the Grid infrastructure in the process of SWAT calibration, compared with those obtained on a Multicore architecture. The experiments we have performed have as purpose the comparisons of the obtained results with those obtained under Multicore execution of the same model, to proof a better performance and a better scalability when executing on the Grid.

In the parallel processing experiment reported in (Rouholahnejad et al., 2011 [200]), a program or multiple computational threads are executed using more than one CPU or processor core. The Multicore executions benefit indeed of all the advantages of a PC and the most important ones are the full control of the job being processed and the easy access to the available resources. Comparing to this, Grid computing imposes some restrictions when it comes to accessing and using the available resources (obtaining a valid Grid certificate, registering and obtaining permissions to use the resources of a certain Virtual Organization, use some specialize tools to gain access to the Grid resources, to submit tasks for execution and to collect the tasks results, etc.). The control of the jobs submitted to execution is not as direct as in the other case but with the help of many available Grid tools and applications such as GANGA, DIANE (described in section 5.4.4) this is no longer a problem.

DIANE is based on a master-worker computing model. The master node is responsible for mapping and coordinating the received tasks to the assigned workers while the worker nodes are responsible for the actual execution of the tasks. DIANE does not submit the jobs directly to the Grid, but it uses GANGA to start a number of agents on different worker nodes. The master-worker paradigm is widely used in parallel applications. It has proved to be efficient when using different degrees of granularity as well as when the partitioning of a task is easy to compute and there are low dependencies (Xhafa et al., 2010 [245]). All these additional requirements introduced by the Grid infrastructure are by far compensated by the huge gain in the execution time for large and complex processes.

We have performed a comparative analysis based on the parallel execution of SWAT calibration, both on the Grid and Multicore architectures. We have used a large hydrological model, covering the Danube River Basin, build using the SWAT2009 program. The parallelization of SWAT is accomplished in this research at the simulation level by simply executing several SWAT runs with different parameters. Each simulation runs the same SWAT model but with different input parameters value. The execution of an iteration consists in performing three important phases: the pre-processing phase, the actual execution phase and the post-processing phase. In the pre-processing phase, the input parameters are generated randomly but within a specific range for each simulation. In the actual execution phase, each simulation is run on different worker nodes inside the Grid infrastructure and in the post-processing phase, the output of each simulation is retrieved and processed. The whole calibration process is presented in details in section 6.3.2.

For the purpose of this case study, the SWAT calibration was executed over the Grid using some well-defined scripts. In the enviroGRIDS project, the SWAT calibration is performed over the Grid using the gSWAT (http://cgis.utcluj.ro/applications/gswat) application but for these experiments we have chosen to focus on the actual parallel execution instead on the graphical user interface provided by the gSWAT. The execution has been invoked through the command line, using dedicated scripts. These scripts can be run only within a certain directory structure, which must contain the calibration inputs, the calibration outputs, backup and the executable files. The calibration inputs contain the observed values, the parameters intervals and the output reaches. The calibration outputs consists of the best values of the simulation parameters and different statistics, the backup directory contains a copy of the SWAT model, and the executable

files contain all the files that are executed during the calibration process - scripts and executable programs.

The steps for calibrating the SWAT model over the Grid are presented in Figure 43:

1. Create the necessary directory structure, needed for running the execution scripts. This directory structure is similar with the one used in SWAT-CUP and in gSWAT for performing a model calibration. This step is performed on the server.
2. Copy the default files needed for calibration in their appropriate places inside the created directory structure. The default files (default input files and executable) are stored in a repository on the server and they are the same for each calibration project.
3. Copy the calibration input files in the created directory structure. The input files are stored in a repository on the server. Using the gSWAT interface, the input files can be uploaded on the server from any location.
4. Run the pre-processing phase in which the input parameters are generated randomly but within a specific range for each simulation. This phase is performed in the SUFI2 pre.bat script, which executes the SUFI2 LH sample.exe program. At this step, Latin hypercube samples are drawn from parameter spaces prior to hydrologic simulation (Rouholahnejad et al., 2011 [200]). The Latin hypercube sampling leads to n parameter combinations, where n is the number of desired simulations and should be relatively large (approximately 500 - 1000) for obtaining meaningful results. The n independent samples are stored in a file called par val.sf2. After this phase, the obtained parameter sets are independent and can be input for parallel simulations.
5. Create the model calibration archive. For each calibration, we create such an archive, containing the directory structure with all the necessary files. The archive is uploaded more easily on the Storage Element, inside the Grid, reducing also the communication overhead between the Storage Element and the workers. Each worker performing a simulation of the calibration process has to connect to the Storage Element and copy the archive locally to have access to the created directory structure and the necessary files needed to run the given task. The archiving step is very important especially for high-resolution projects, which can result in thousands of input files.

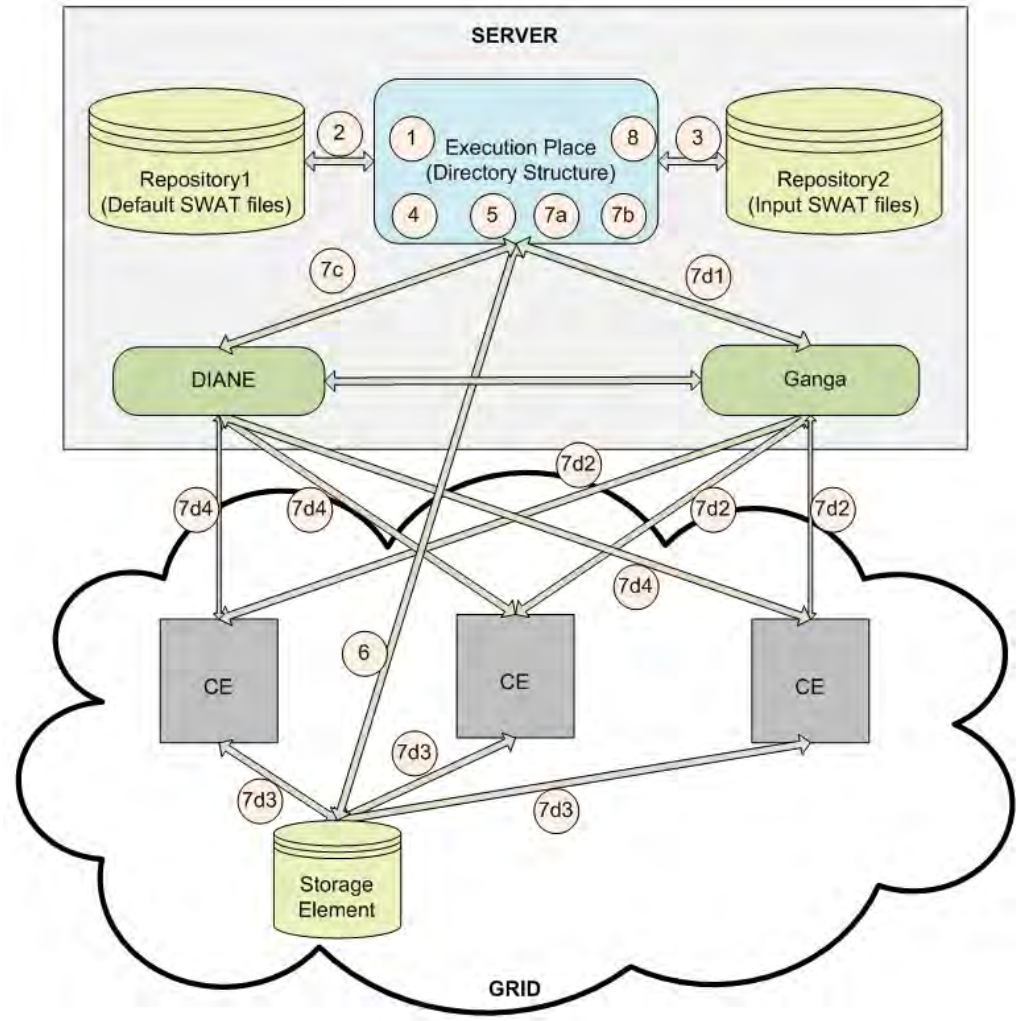

Figure 43: SWAT Calibration Steps on Grid

6. Upload the model calibration archive to the Storage Element to become accessible to all the workers. This step is executed from a gLite machine. It requires the existence of a proxy-certificate and the necessary permissions for connecting and storing files on the Storage Element.
7. Start the parallel calibration process in which each simulation of an iteration is run on a different worker inside the Grid infrastructure. This execution part consists in running the simulation program n times and extract from the SWAT output files the simulated output variables of interest, corresponding to the observations. The actual execution is performed using the SUFI2 run.bat, which runs the SWAT Edit.exe program. At this phase, the set of sampled parameters obtained in step 4 are copied in the input SWAT

files, in their appropriate place, process that can become quite complex especially if the model has a large number of input files as well as parameters. After the parameters are in the proper places, the SWAT model is executed and the outputs are extracted from the SWAT output files (Rouholahnejad et al., 2011 [200]) using the SUFI2 extract rch.exe program. Each such simulation represents one job executed by one worker in the Grid. The parallelization is achieved at this phase. We use DIANE and GANGA tools to run the jobs on the Grid, performing the following steps:

a. Create the script to be executed on each worker node.
b. Create the DIANE script in which we specify the input data, the output data and the execution task for each worker node.
c. Start the DIANE master, which will coordinate the worker nodes, started to execute the simulations tasks and will collect the results from each job.
d. Launch the jobs to execution:
    i. Start the Grid worker nodes using GANGA.
    ii. The GANGA tool will start at most as many worker nodes as required, depending also of the availability of the Grid resources.
    iii. Each started worker node will connect to the DIANE master node to get information on the task it has to execute. Before starting the execution, a worker node has to connect first to the Storage Element and retrieve the input files specified by the DIANE master. In this experiment, the input file is the model archive we have uploaded in step 6. After the worker node obtained the model archive, it has to extract the contained files and to reconstruct the necessary directory structure for a proper execution. Depending on the configured number of started worker nodes and on the availability of the resources on the Grid, a worker node will execute one or several tasks. If the number of worker nodes is smaller than the number of simulations we want to perform, some tasks will have to wait for the execution of others, increasing the total execution time.
    iv. After a worker node finishes the execution of a job, it connects to the DIANE master and sends the archived execution results.

8. Run the Post-processing phase in which the output of each simulation is retrieved and processed. This phase is the last one and it is performed using the SUFI2 post.bat, which executes a series of programs (SUFI2 goal fn.exe, SUFI2 95ppu.exe and SUFI2 new pars.exe). Those programs have the scope of computing the objective function, computing the 95% percent prediction uncertainty and computing the updated parameters, suggesting them to the next iteration (Rouholahnejad et al., 2011 [200]). The execution of such a process is repeated for as much iterations as needed to obtain a satisfactory outcome. More details of this process are presented also by Rouholahnejad et al. (2011) [200] and by Bacu et al. (2011a) [16].

The technical specifications of the Grid infrastructure (gLite middleware) used for testing are the following:

- gLite middleware;
- Resources:
    - one Computing Element (CE) and one Storage Element (SE);
    - Worker Nodes (WNs) – computational resources – 128 physical CPUs with 1024 logical CPUs;
    - Storage Element – storing resources ~ 13 TB;
- Ganga tool – used as frontend for job definition and management;
- Diane tool – employed for efficient usage of the distributed computing infrastructures.

We have performed experiments on two instances of the Danube river model. The model has been developed by our enviroGRIDS partners from EAWAG (http://www.eawag.ch/) as described by Rouholahnejad et al. (2011) [200] and the two instances of the model differ from the size point of view (i.e. the amount of data and the number of parameters taken into consideration for the calibration process). The first instance (Danube1) is a large project and is the one used for tests on Multicore architecture. The second instance (Danube2) is even larger from the size point of view and due to its dimensions it could not be tested on the available Multicore machines (Rouholahnejad et al., 2011 [200]).

For Danube1 project, the calibration of the SWAT was done by running the SUFI2 program and setting 48 simulations. The execution of this process took around 2 days to run on a

server without any parallelization option. In (Rouholahnejad et al., 2011 [200]) the number of nodes that perform simulations could be the same or fewer than the number of available CPUs. The maximum number of jobs that could be submitted is computed based on the project size and on the available memory too. Similar, in Grid we can approximate an optimum number of workers that are needed to run the jobs as fast as possible but theoretically the Grid brings unlimited resources so that the number of executed jobs is not dependent on the available memory or on the number of processors.

To be able to make a meaningful comparative analysis based on the results obtained by Rouholahnejad et al. (2011) [200] and those obtained in our research, we use the same performance measures which are commonly used to evaluate the performance of parallel computation (Rouholahnejad et al., 2011 [200]): speedup and efficiency. The formulas for the two measures are the followings:

S(n) = T1 / Tn is the speedup for n parallel sessions;

Where:

- T1 is the computed time of the task when only one processor is used;
- Tn is the computed time when n processors are used.

E(n) = S(n) / n is the efficiency of a parallel system of n processors.

The ideal speedup is defined by the number of processors/worker nodes. For example, in a machine with eight processors, the ideal speedup is eight. In a Multicore machine, the gap between the ideal speedup and the obtained speedup grows as the number of processors exceeds and as the number of jobs increases. This is due to the increase in the communication of each CPU with the hard disk, resulting in a loss of speed due to the disk limitation (Rouholahnejad et al., 2011 [200]). Based on the number of parallel jobs set to run on a Multicore machine, the files in the project (which are in a large number, especially in large projects) are simultaneously read and written to the hard disk. This is the reason why the speedup decreases, for large projects, as the number of parallel processes increases (Rouholahnejad et al., 2011 [200]).

7.3.1.1. Multicore execution results

Figure 44 presents the execution results, in seconds, obtained for executing one iteration (48 simulations in this case) in the process of calibrating the Danube1 project on a Multicore server, using different number of cores. As seen from the figure, there is a clear improvement between running on a single core, which took approximately 1 day, 11 hours and 30 min (127,920 sec) and running on 24 cores, which took 4 hours and 15 min (15,267 sec).

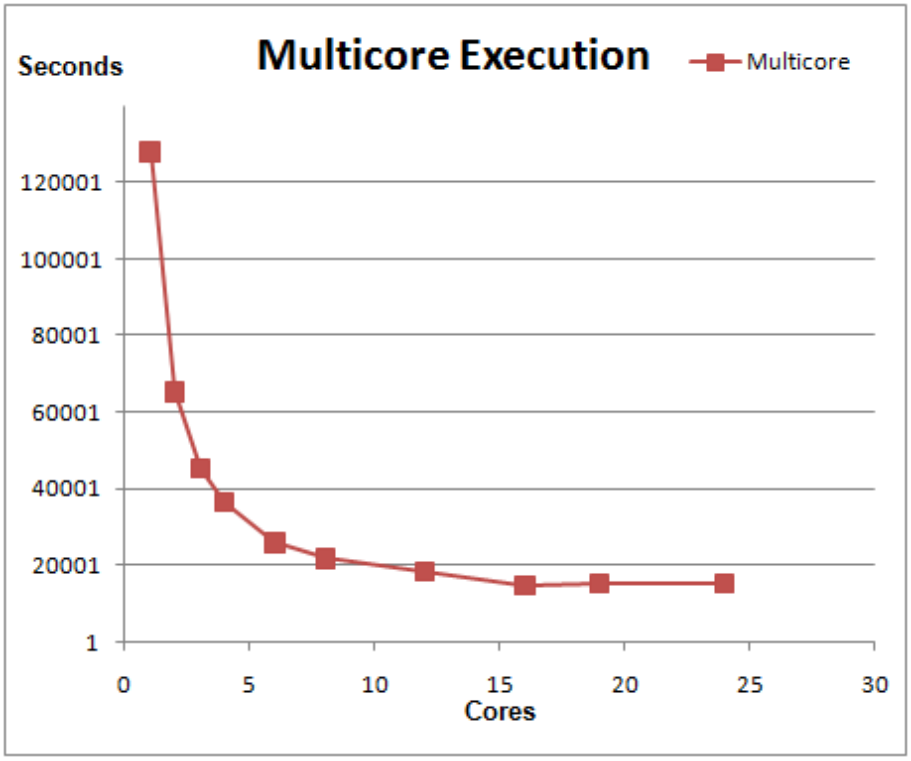

Figure 44: Danube1 SWAT Calibration - Multicore Results

7.3.1.2. Grid execution results

Similarly, Figure 45 presents the execution results, in seconds, obtained for executing one iteration (48 simulations in this case) in the process of calibrating the Danube1 model on a Grid architecture, using different number of worker nodes.

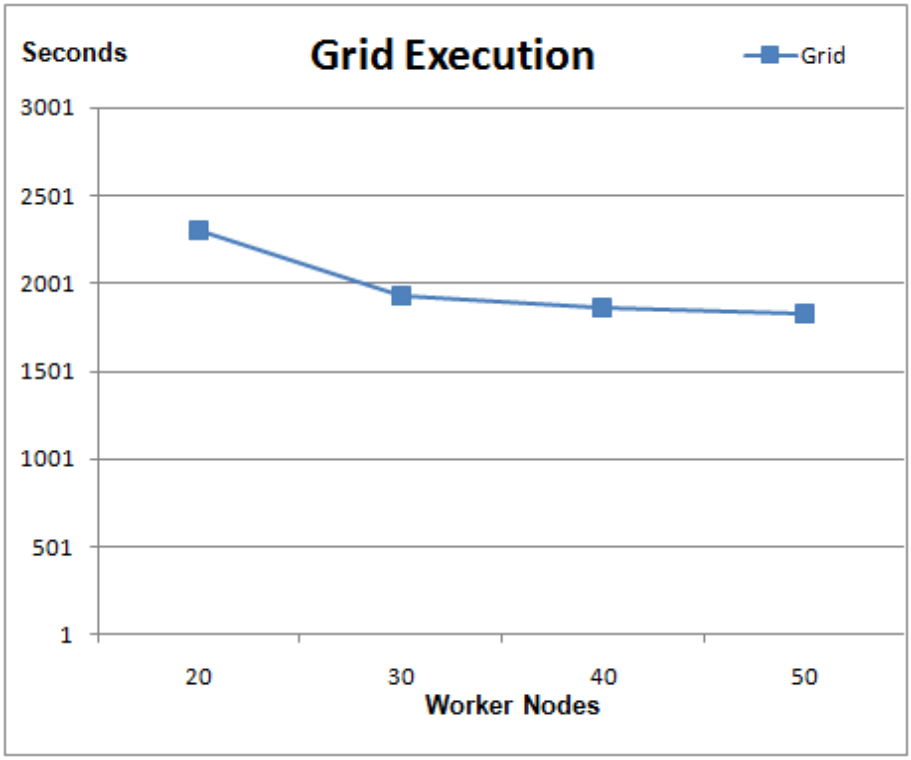

Figure 45: Danube1 SWAT Calibration - Grid Results

The experiments were carried out using the gLite middleware, inside the VO supported by CERN vo.gear.cern.ch and using DIANE and GANGA tools. The variable in these experiments was the number of worker nodes started for executing the 48 simulations of one iteration: 20, 30, 40 and 50 worker nodes. The improvements are even significant when executing on the Grid. The same execution, which took 1 day, 11 hours and 30 min (127,920 sec) when executing sequentially on one machine, it took around 39 min (2,305 sec) on 20 worker nodes and around 31 min (1,825 sec) on 50 worker nodes. The results obtained both on Multicore and Grid architectures can be compared in this case from the total execution time point of view when performed on around 20 processing units (cores and worker nodes respectively). At this level, the Grid results are more than 6 times better then Multicore results.

Both Figure 44 and Figure 45 show the improvements brought by parallel architectures especially in the execution of real-world computation-intensive applications such as calibration of a large-scale hydrological model, which was the case of our study. Our goal though was to highlight the fact that the Grid architecture can bring even better improvements than the Multicore architecture especially due to the large number of resources it can offer but also due to

the specific management of these resources. Although the number of worker nodes that we used on the Grid is contained within another range comparing with the number of cores used by Rouholahnejad et al. (2011) [200] we can still observe the improvements brought by the Grid architecture considering the speedup and the efficiency presented in Figure 46 and Figure 47.

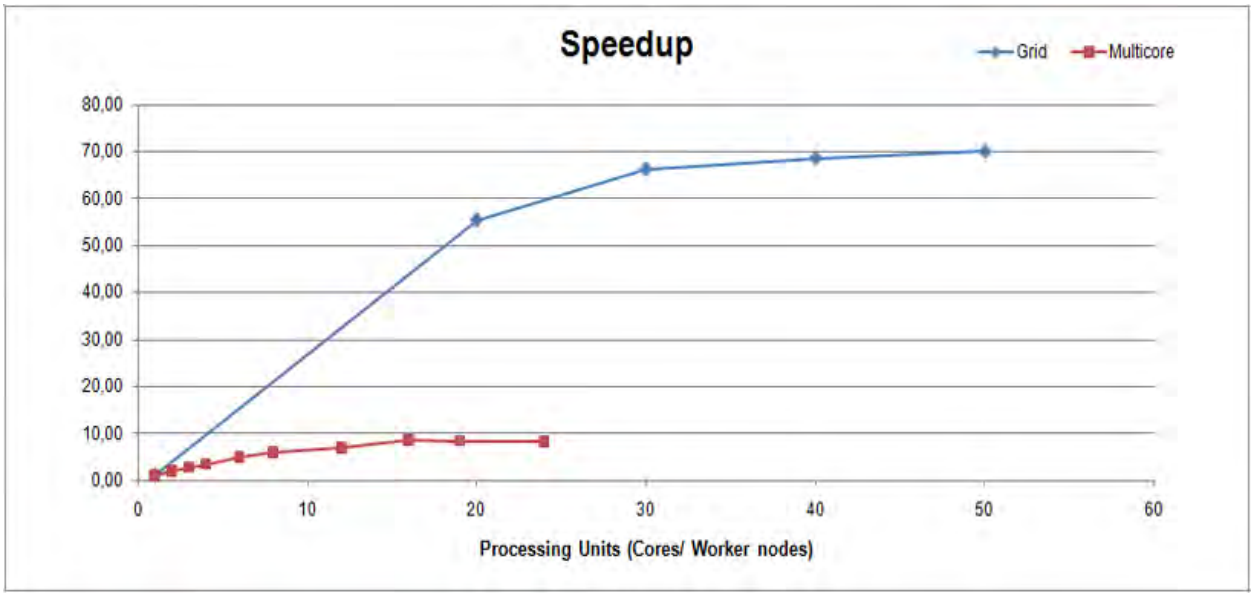

Figure 46: Danube1 Model - Comparative Speedup Multicore vs. Grid

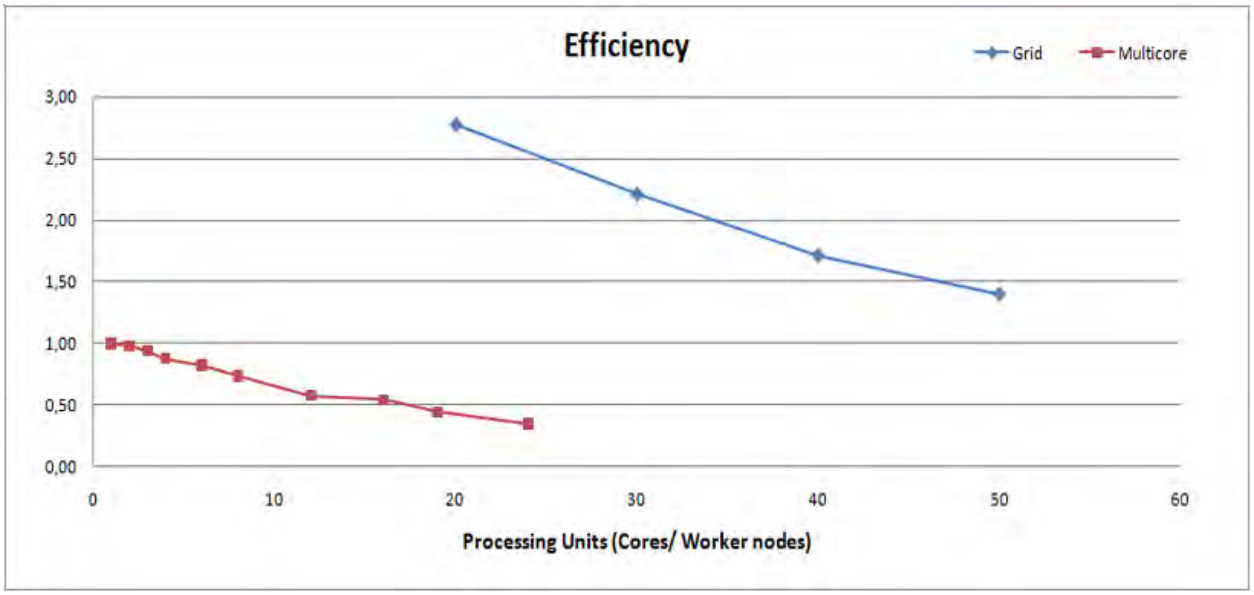

Figure 47: Danube1 Model - Comparative Efficiency Multicore vs. Grid

Figure 46 presents the speedup obtained for both architectures, considering different number of processing units and having as reference point the sequential execution on a single core station. The processing units refer to cores in the Multicore architecture, and to worker nodes in the Grid architecture. The figure clearly highlights the difference between the two architectures and the great improvements brought by the Grid infrastructure.

Similarly, Figure 47 presents the efficiency obtained for both architectures. Once again, this figure confirms, if needed, that using the Grid architecture we obtain a better efficiency then using the Multicore architecture. Although the trend lines of the two graphics are somehow similar, the obtained values for Grid are clearly much higher.

We have performed measurements over the Grid on Danube2 to highlight the scalability offered by this infrastructure, although we have no comparison with results obtained on the Multicore architecture, as the project is too large to run on the available Multicore machines, especially for a large number of simulations. For the execution of the Danube2 on the Grid, we have set the number of simulations to 24, 100 and 500.

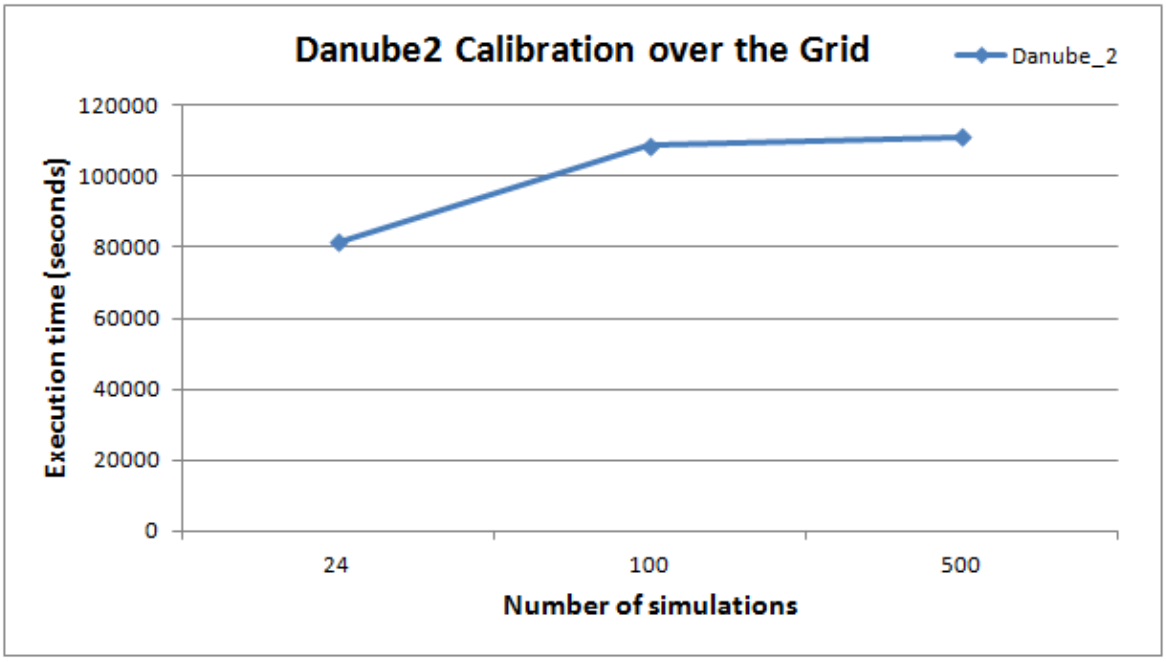

Figure 48: Danube2 SWAT Calibration on Grid – Execution Time

Figure 48 presents the execution of Danube2 and proves the Grid scalability, especially for large-scale applications such as the execution of a large-scale SWAT hydrological model.

The input files used for calibrating the Danube2 model have a size of 1,3 GB and consists in 327,000 files which have to be backup for execution. The archive we have to copy on the Storage Element (contains the input files, the default files and the executables needed for calibration) has 1,6 GB. This archive is copied only once on the Storage Element, at the beginning of the calibration process, but each worker node has to copy it locally for performing its tasks.

Due to the large number of resources in the Grid infrastructure, the execution of the Danube2 model is scalable as the number of simulations performed for the calibration increases. When performing 100 simulations, the actual execution on the Grid of the calibration process took approximately 30,18 hours (108,635 sec.). When increasing the number of simulations to 500 (normally a good simulation number for obtaining meaningful results), the execution on the Grid took approximately 30,85 hours (111,032 sec.).

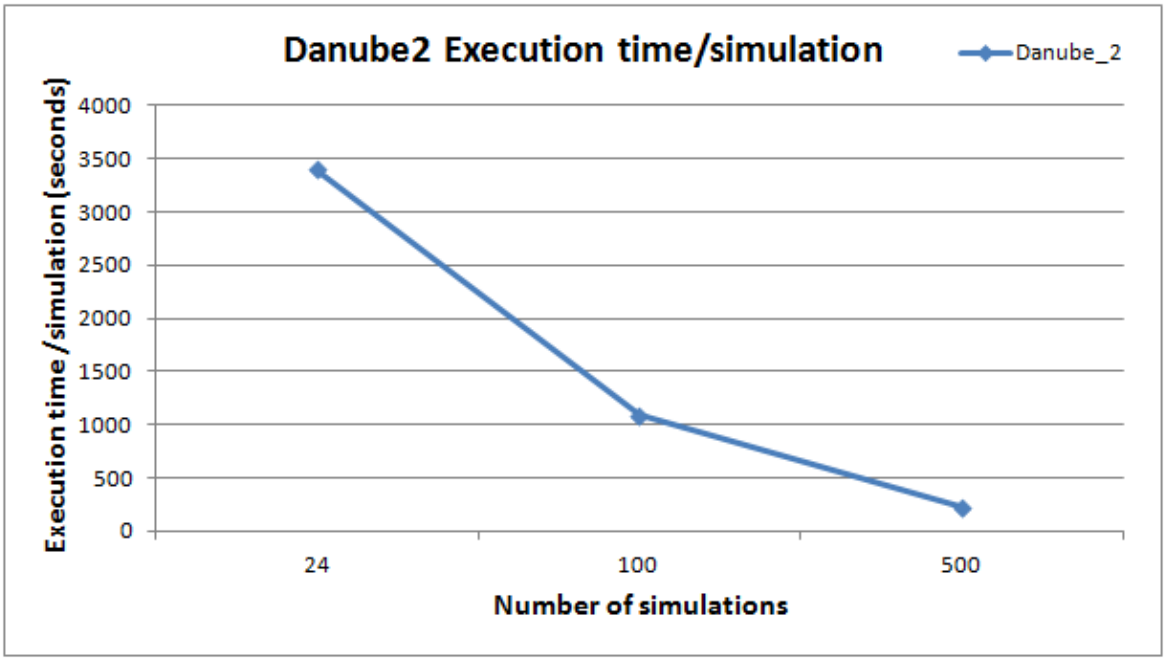

Figure 49: Danube2 SWAT Calibration on Grid - Execution / Simulation

The scalability of the Grid can be deduced also from Figure 49, which presents the execution time/simulation in the calibration of the Danube2 model, as the number of simulation increases. When performing 24 simulations, the average time of one simulation is approximately

57 minutes (3,398 sec.). When the calibration process is set to 100 simulations, the average time of one simulation is approximately 18 minutes (1086 sec.) while for 500 simulations, the average time of one simulation is 3,7 minutes (222 sec.). As the number of simulations in the calibration process increases, the average time/simulation decreases.

For obtaining even better execution times and an easier model management, we are considering mapping the calibration model to Grid databases as future work. This will improve the time needed to update all the calibration parameters in the appropriate files and will allow the definition of execution scenarios based on calibrated models in a more flexible way, considering the large volumes of data related to SWAT models. For achieving this goal, we have to consider the parallel and Grid databases architectures as well as important issues and constrains which can appear in this area.

To conclude this result analysis, there are indeed some additional requirements that have to be fulfilled for accessing a Grid infrastructure (obtaining a Grid certificate, registering and obtaining permissions for running in a certain Virtual Organization, using some dedicated tools to access the available resources, to submit jobs for execution, to collect the job results, etc.), but the improvements Grid brings in the execution of real world, large-scale application are significant comparing to other parallel architectures, especially to Multicore.

7.3.2. SWAT Calibration Execution on Baobab Cluster (UNIGE)

The execution of SWAT calibration was done on Baobab cluster infrastructure using the SLURM (http://slurm.schedmd.com/) workload manager. The steps are quite similar with the ones performed in the Grid except that the input files were placed in the common file system and instead of sending jobs to CEs, we have launched jobs on individual nodes in the cluster.

7.3.2.1. Platform Specifications

- HPC Cluster – BAOBAB – provided by UNIGE (http://baobab.unige.ch/);
- bought by end of 2012 and operational since early 2013;
- master node in charge of administration and backup;
- 57 compute nodes, each with
    - 2 Sandy Bridge Intel(R) Xeon(R) CPU E5-2660 @ 2.20GHz cpu with 8 cores each;

    - 64 GB of RAM;
        - for a total of 912 cores.
- 1 compute node with;
    - 4 Sandy Bridge Intel(R) Xexon(R) CPU E5-4640 @ 2.40GHz cpu with 8 cores each.
- a server node providing 40 TB (extended in December 2013) shared file system **FraunhoferFS (FhGFS)**;
- **InfiniBand** 4xQDR (40Gbit/s) connectivity between master, nodes and storage.

7.3.2.2. Execution Flow

1. Archive the input files;
2. Copy the archive to the local file system of each requested node (/scratch);
3. De-compress the archive on / scratch in a unique directory (on each node you can have 16 parallel computations)
4. Launch the jobs;
5. Compress the output in an output archive and copy it on the parallel file system (in the user ‘s /home directory);
6. Once the jobs are finished, you don't have access to your /scratch data anymore.

7.3.2.3. Results

Figure 50: SWAT Calibration on Baobab Cluster - Execution Time

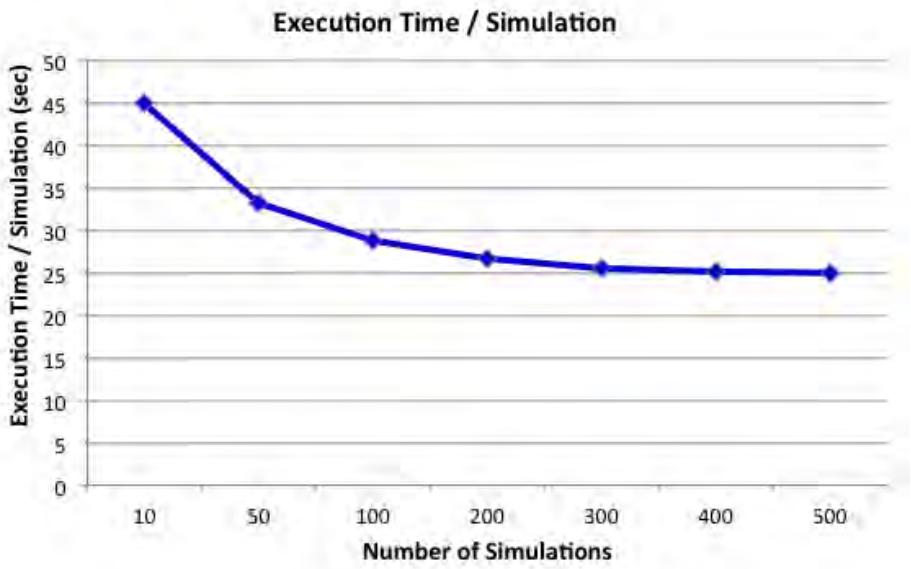

Figure 51: SWAT Calibration on Baobab Cluster - Execution Time/Simulation

7.3.3. SWAT Calibration Execution on Open Stack infrastructure (UNIGE – HEPIA)

The particularity on the Hepia cloud, in executing the above mentioned SWAT calibration steps, is that the input data was stored on a proxy machine and we have copied the input data, to each launched VM, using multicast (Udpcast software). This approach reduced significantly the download time.

7.3.3.1. Platform Specifications

- hepiaCloud - academic Cloud platform based on OpenStack, connected to SwissACC (Swiss Academic Compute Cloud) platform/project;
- **Available ressources -** 41 compute nodes available;
- vcpus free: 287 / 304;
- ram free: 1953 GB / 1992 GB;
- disk free: 9060 GB / 9148 GB;
- All virtual machines in a private network which require the usage of an ssh gateway (gw.lsds-rg.org).

7.3.3.2. Execution Flow

1. Archive the input files;
2. Copy the archive to a Web server – accessible by each virtual machine.
3. Launch instances with a predefined image and flavor;
4. Copy the input data on each virtual machine, using multicast (Udpcast software).
5. De-compress the archive on each virtual machine and start the execution;
6. Compress the output in an output archive and copy it on a proxy machine.
7. Wait for all the computations and delete the created instances.

7.3.3.3. Discussions

Issues:

- Downloading the same data several times will rise a network bottleneck issue;
- Disk I/O performance issue;
- No available resources to match the full requirements;
- Small range of IPs (~200 IP) for this use case (required 500 VM).

Solutions:

- Using multicast (over UDP instead of Unicast) - reduces significantly the download time independently of the number of instances.
- Increasing the amount of RAM in the VM flavor to 5GB and create a tmpfs (a file system in memory) to minimize the I/O overhead.
- Computation and data storage is made on RAM (including the downloading of the input data).

7.3.4. SWAT Calibration Execution on Open Stack infrastructure (UNIGE – SwissACC)

The execution of SWAT calibration on the SwissACC Cloud – Hobbes was performed using the boto library to access the data from and to S3 compatible Object Store

7.3.4.1. Platform Specifications

- SwissACC – Swiss Academic Compute Cloud;
- Swiss wide computational science platform offering;

- Resources;
- Services;
- High quality know how for user and application support;
- Provides access to Cloud platforms - UZH Cloud (Hobbes) http://cloud.gc3.uzh.ch/;
- Compute nodes available: 80;
- vCPUs available: 80;
- RAM available: 160 GB.

7.3.4.2. Execution Flow

- Personalize a VM image (called SWAT image) based on an Ubuntu standard OS and we incorporate into it the required data of the SWAT model and the binary code.
- Spawn new VMs instances from the SWAT image. Practically, this requires optimizing the data transfer of the SWAT image through the local network in order to avoid an instant network bottleneck. To this end, we have tested two approaches. The first one consists in using a smart distributed object store system "Cepth" for the SWITCH Cloud case. The second one consists in using multicast data delivery to transfer the SWAT image from its storage folder to the destined machines of the Hepia-Cloud platform. Both approaches showed a good performance. The last step consists in starting the computation where each VM performs a parameter sweeping of the model, runs the simulation, uploads the result files on a given location and finishes its execution.

7.3.4.3. Discussions

Copy on write (COW) technique is an optimization strategy based on the idea that when multiple tasks use identical copies of the same information, each task can be given pointers to the same resource instead of creating copies of that information for each process. The problem with this strategy is that when a local copy has been modified, the other processes are not aware of that change. When a task attempts to make a change to the shared information, a private copy of that information is created and the tasks is redirected to make the changes on that private copy, preventing thus the changes to become visible for the other processes. All these operations take place in the operating system kernel, making everything transparent.

The test performed using a smart distributed object store system "Cepth". For this approach, the execution steps are the following:

1) Prepare a base volume, which will include the basic SWAT files and the input files specific for each particular use case. This means that we will personalize a VM image (called SWAT image) based on an Ubuntu standard OS and we will incorporate into it all the required data.
2) Spawn the necessary number of VMs instances from the SWAT image. Using the smart block storage (e.g. Ceph) we will optimize the data transfer of the SWAT image through the local network in order to avoid an instant network bottleneck.
3) Each VM will execute the SWAT model with different parameter sets and upload the results in storage. The execution and the data changes are performed by each VM in its volume.

7.3.5. SWAT Calibration Execution on Windows Azure (UNIGE – SwissACC)

To perform the execution of SWAT model on Microsoft Azure Cloud, we have developed a program that starts automatically a given number of Linux VMs on Azure. Upon starting, each VM runs a script, which starts the execution as described above. Within the Azure platform, we only had access to 4 virtual machines so the tests perform here were just a proof of concept for the procedure used to port applications on this infrastructure.

For these tests, we used a Java program that starts automatically a given number of Linux VMs (maximum 4) on Azure. We have created an Azure storage container and uploaded the SWAT archive there. After this, we have started automatically a given number of Linux VMs on Azure. Upon starting, each VMs runs a script which:

- Copies the input archive locally, from the Cloud storage;
- Executes the SWAT model on this input;
- Retrieves and copies the results back into the storage.

7.3.6. Global Flood Model Execution on Baobab Cluster (UNIGE)

The execution of the Global flood model consists of two main procedures:

- Generation of hydrological model input, using the Global Climate Model - EC-Earth model;
- Execution of the Continuum hydrological model.

The execution of the first procedure /1 year/1 domain has an output of approx. 1.4 GB. For the total number of 30 domains and for the total execution period (past and future) ~150/185 years the output data after the execution of the first procedure is ~ 6,5/7,8 TB while the final output (after the execution of the second procedure) consists of another ~ 1-2 TB.

We have successfully executed 13 out of 30 domains on Baobab cluster. These domains are already calibrated while the rest of 17 domains are still in the process of being calibrated.

The tests performed within this experiment were useful in gaining experience in working with large amount of data within a single execution. We have identified challenges and drew conclusions on possible solutions to transfer, store and process large amounts of data.

7.3.7. Gridification of OGC Web Services (UTCN)

The OGC (Open Geospatial Consortium [167]) Web services (OWS) are Geospatial services used to exchange information in an interoperable and efficient way over a distributed environment. The implementation of OGC specifications is a step forward into the process of sharing and making the geospatial information accessible to different communities but also a step forward in achieving an interoperable environment. Using the OGC standards, different GIS applications can work together, exchange information over a network and interoperate. The OGC Web services process data on demand, based on users' requirements and return the data under different formats (content and structure), as specified by the users (Di, 2004 [54], Werder and Kruger, 2009 [239]).

In (Rodila and Gorgan, 2012 [198]) we have analyzed the interoperability between the Geospatial and the Grid infrastructures through the gridification of the OGC Web services, using a mediation approach. We have made several test on these services executed over the Grid infrastructure (gLite Middleware), with a varying number of features in the database (amount of data) and a varying request complexity (number of performed service requests). These tests were done during the enviroGRIDS project and they proved the existence of a complexity boundary for the execution on each computing background. Based on this boundary, the execution is more efficient on a computational platform or another.

The experiments performed as a proof of concept were applied on an instance of WFS service, developed as a standard implementation inside the *deegree* project (http://www.deegree.org/), and had as goals to analyze the behavior of the execution time obtained when executing on the Grid environment, under different created conditions. The main goal was to emphasize the behavior of a Complexity Computation Component in correctly filtering the received requests as Grid or Web requests.

Figure 52: OGC Gridification - Grid vs. Web Execution - 10 Features

Figure 53: OGC Gridification - Grid vs. Web Execution - 100 Features

The variable parameters modified during the experiments are the amount of data (number of features in this case), the number of queries enclosed in a service request and the number of working nodes inside the Grid (Rodila and Gorgan, 2011 [197]). The performed experiments were meant to emphasize not only the ideal case to execute a gridified OGC Web service but also the cases that raised several problems. The execution environment was the gLite platform and the execution VO was the one created for the enviroGRIDS project envirogrids.vo.eu-egee.org. We have used the GANGA and DIANE Grid tools (described in section 5.4.4), developed by CERN, for the job management: division of jobs, submission on the Grid, monitoring, result merger, etc.

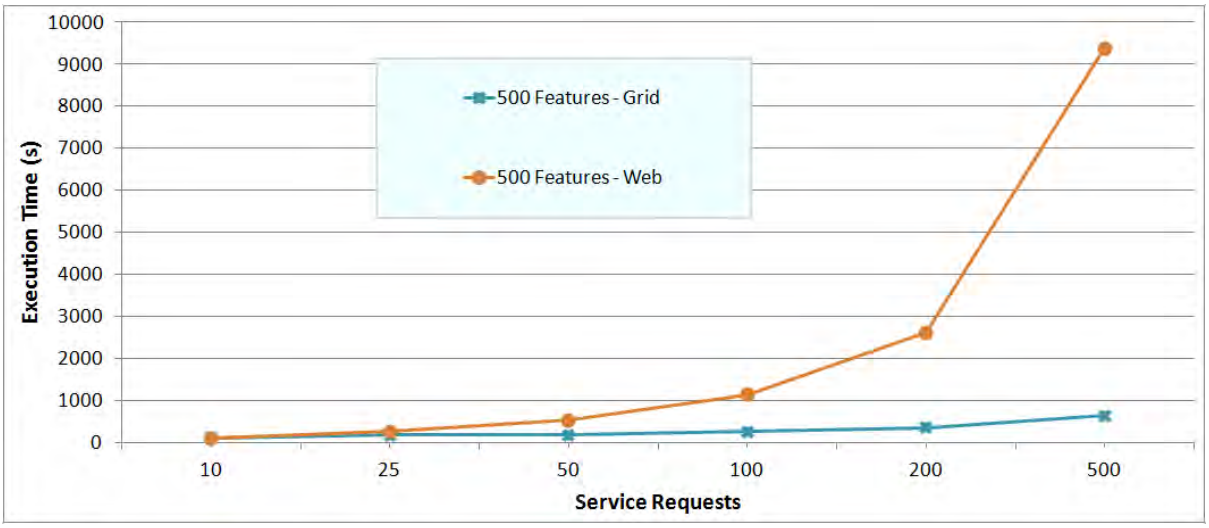

Figure 54: OGC Gridification - Grid vs. Web Execution - 500 Features

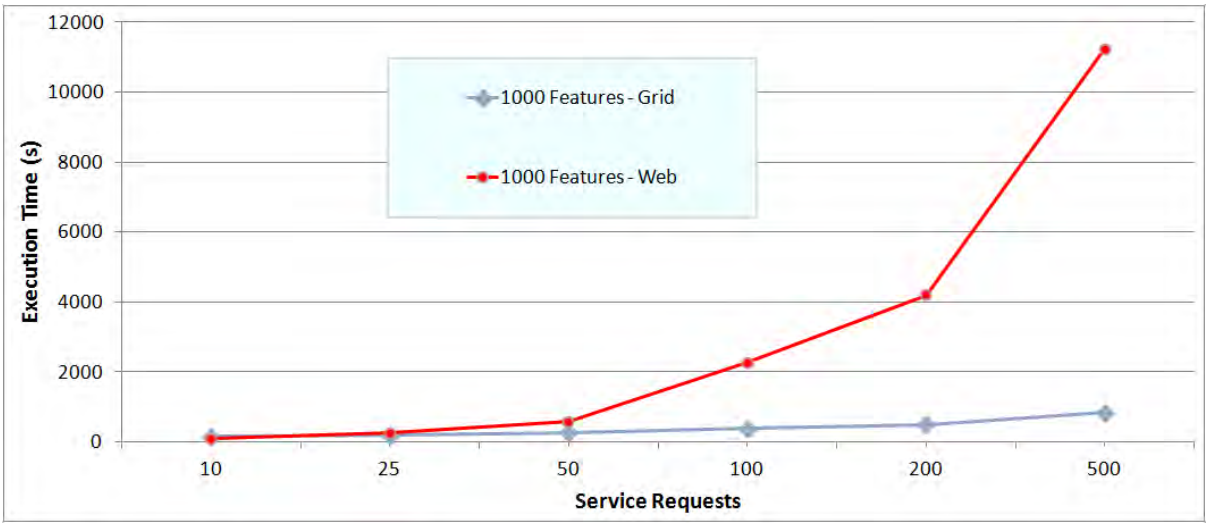

Figure 55: OGC Gridification - Grid vs. Web Execution - 1000 Features

The platform used for the enclosed tests was formed of 128 quadcore processors, equivalent to 512 cores. Each such core was considered a worker node inside the gLite platform. These tests proved the existence of a complexity boundary below which the execution of an OGC Web service request is more efficient when it is performed on a Web server (a single computer) than over a Grid infrastructure.

Figure 52, Figure 53, Figure 54, and Figure 55 bring forth the comparative results of the service execution both over the Web and over the Grid, with 10, 100, 500 and 1000 features in the database and a varying number of performed service requests (10, 25, 50, 100, 200 and 500 requests). The figures highlight the complexity boundary (at the intersection of the two graphics in each figure) and the point at which it is reached. The results analysis concluded that the complexity boundary is estimated at a lower number of service requests as the number of features in the database (amount of requested data) increases.

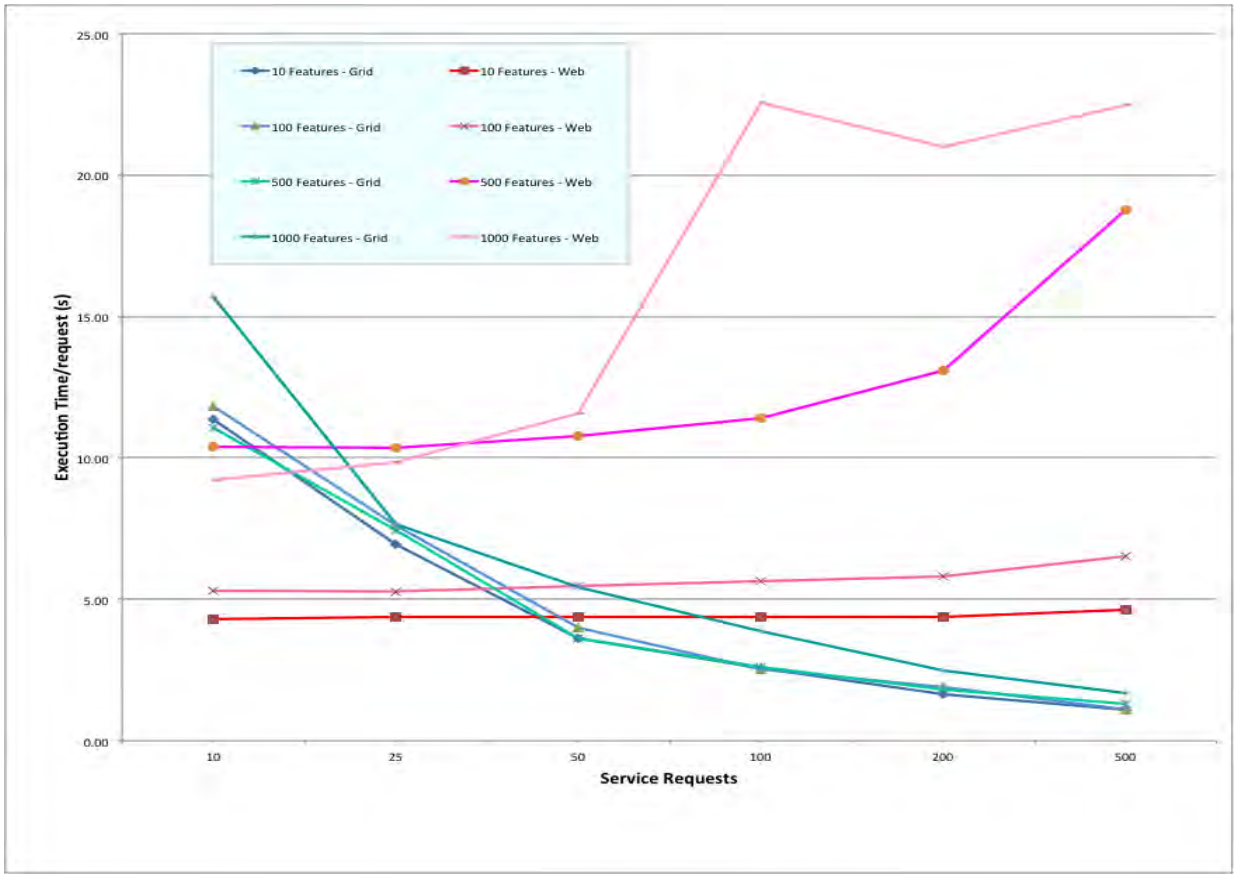

Figure 56: OGC Gridification - Execution Time / Request

Figure 56 illustrates the execution time/request results. From this graphic we can observe that the Grid is slightly dependent on the number of features in the database (amount of data), as it disposes of a large number of computational resources while the Web execution time increases as the number of features increases, although not proportionally. Another important observation pointed out by this graphic is that the Grid execution time/request decreases as the number of service requests increases while the Web execution time/request increases under the same condition.

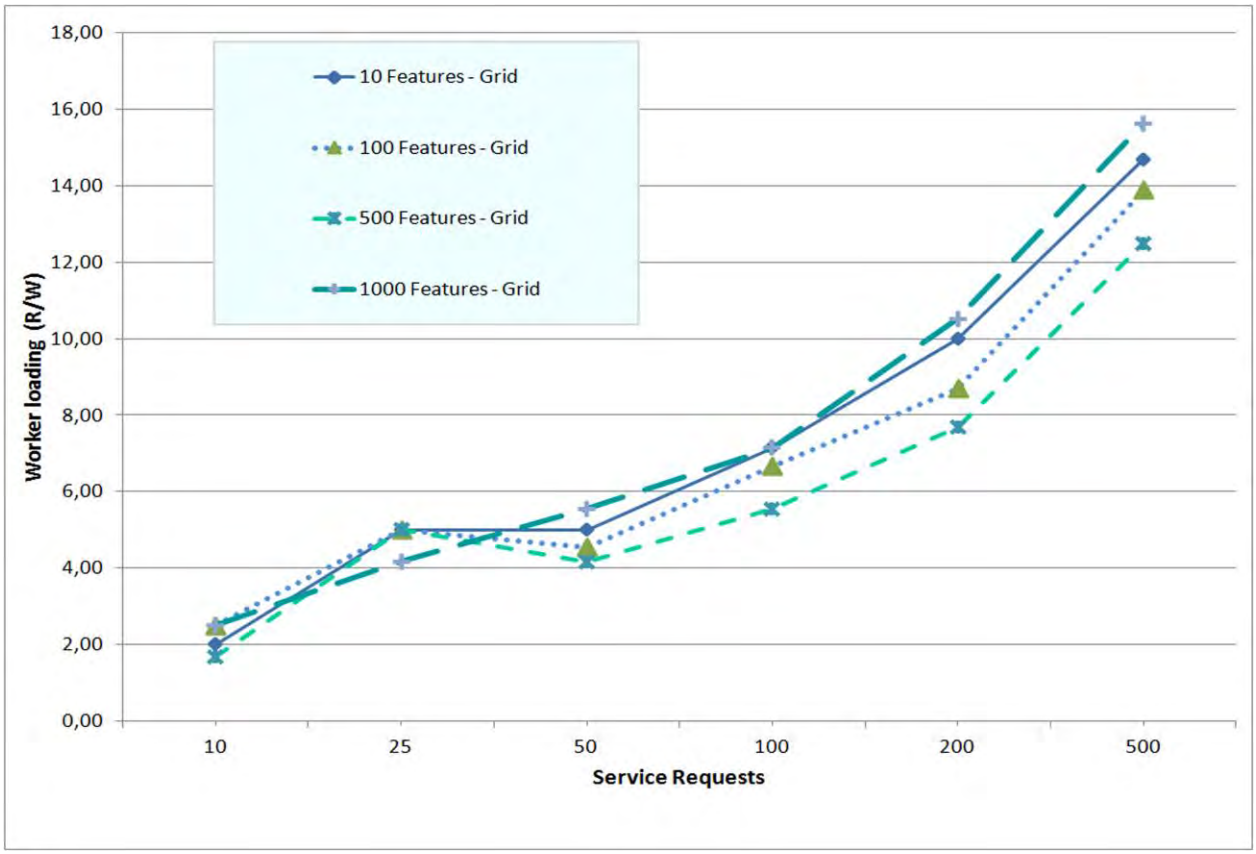

Figure 57: OGC Gridification - Worker Loading (Request / Workers)

Figure 57, highlights the worker loading in the Grid i.e. how many requests does a worker node in the Grid perform, while Figure 58 underlines the worker loading in the Grid considering also the number of features in the database. Both graphics emphasize a better resource utilization as the number of requests increases. They also highlight once again that the

Grid is slightly influenced by the number of features in the database, i.e. the amount of data. The results show that the Grid infrastructure is useful especially for large amounts of data and-or for a high number of service requests.

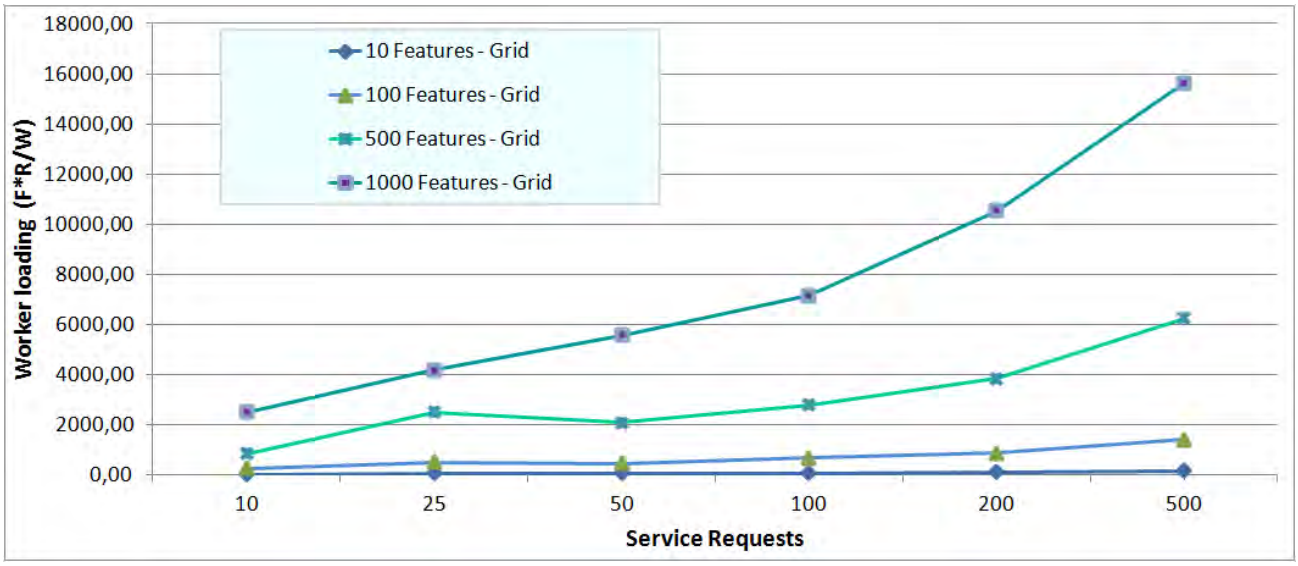

Figure 58: OGC Gridification - Worker Loading (Features * Requests / Workers)

The conclusion formulated by these results is that there are cases in which a local server is more appropriate to run simple OGC Web services requests than choosing a parallel and distributed platform and dealing with all the necessary resource management and application deployment issues. For complex requests, the usage of such a platform is indeed beneficial. The Grid solution is scalable because the performances increase as the number of features and requests increases (number of applications, users and data). The Web solution is less scalable as the performances decrease under the same conditions.

Through these experiments, we have dwelt on the major problems that make the interoperability between the two platforms hard to achieve but also on a mediation approach for gridifying the OGC Web services. This approach was introduced as a possible solution to the interoperability problem.

7.3.8. Landsat 8 Data Acquisition (UNIGE – UNPE/GRID-Geneva)

Landsat 8 imagery is freely available for registered users through USGS network using Web interfaces such as EarthExplorer (http://earthexplorer.usgs.gov/), Glovis (http://glovis.usgs.gov/), or USGS Application Programming Interface (API). These images are

also archived and freely available (without registration) through Google Earth Engine (GEE) and Amazon Web Services (AWS). These last two data providers are accessible either directly via HTTP protocol, or via gsutil (https://cloud.google.com/storage/docs/gsutil), which is a Python application that gives access to Google Cloud Storage from command lines. In (Rodila et al., 2016), we have analyzed and we have discussed the performances and the reliability of these three Landsat 8 data providers and their associated access methods (Table 1).

Table 1: Landsat 8 Data Providers and Accessing Methods

Provider / Access method	http	gsutil
USGS	usgs_http	-
GEE	gee_http	gee_gsutil
AWS	aws_ pds_http	aws_pds_gsutil

To be able to perform the automatic data acquisition flow, we have developed a testing platform, which is described in Figure 59.

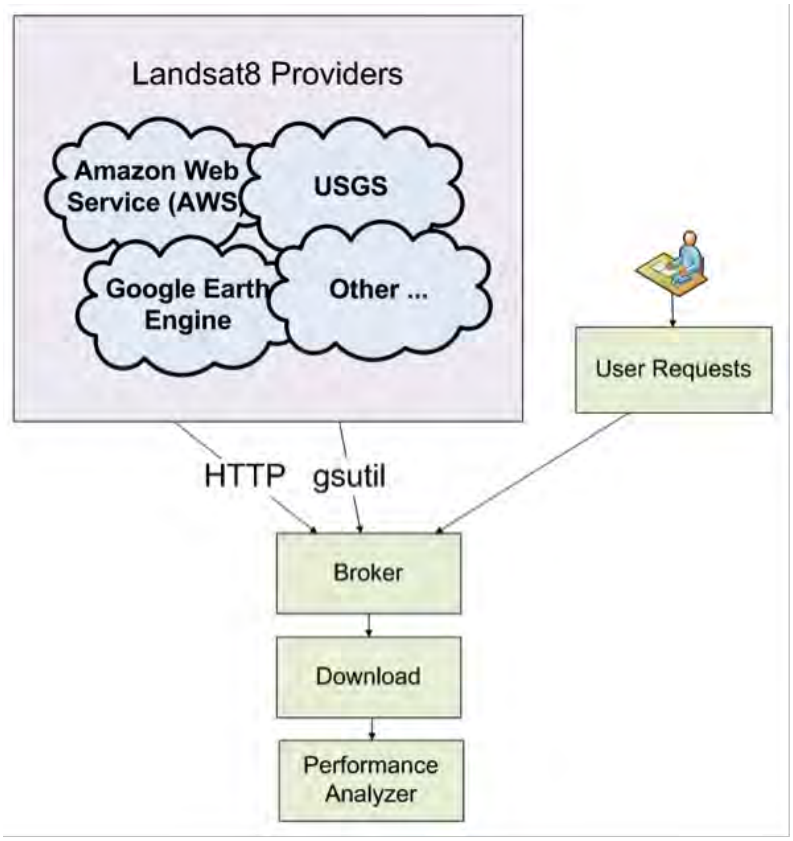

Figure 59: Landsat 8 - Data Acquisition Platform

The components of this platform are developed in Python. The Broker Component is used to automatically connect to different available providers using available access methods based on user requirements. The Download Component is responsible for automatically downloading the previously selected data while the Performance Analyzer Component is used to monitor and analyze different performance parameters of the data download process from each data provider on each transfer access method (Table 1) referred as protocol in the following lines.

The main objective of the following tests was to define and analyze which combination of (provider and access method) is faster and what are the possible network instabilities or the potential speed limitation after the download of a large number of scenes.

In order to avoid possible interaction between providers, each session of test was performed randomly (i.e., protocols were run in random order).

Three profiles of bands combinations were randomly used:

- 3 bands (e.g., 2,3,4) to simulate the case of a simple composite process,
- 3 bands (e.g., 2,3,4) and panchromatic (e.g., 8) to simulate the case of a simple composite process with pan-sharpening,
- All bands (e.g. 1 to 11) to simulate the case of a more complex process.

To avoid possible download speed limitation in case of recurrent download of the same scene (noticed during the development phase of the test), sites (path and row) were selected randomly. The number of acquired scenes on a site randomly varied from 1 to 3.

The way providers distribute scenes varies. USGS and GEE are providing a single zipped file containing all bands, but using different compression format (respectively tar.gz and tar.bz). AWS allows a direct access to each band in a geotiff format with the compress deflate option (meaning they can be used directly). Consequently, unzipping of data was added in the test process, as well as a process of cleaning (removing zipped file and unnecessary bands).

A comparison between the data access phases of two different providers (GEE and AWS), using gsutil, is shown in Figure 60. This figure shows the distribution of these phases (download only for AWS, shown in red; and download, unzip and clean for GEE, shown in blue) using as an example the access of all bands of scene IDs LC81980212015103LGN00,

LC81980212015279LGN00, LC81980212014260LGN00. This was tested on the 23rd of February 2016 and gives just an example to emphasize the differences in the data acquisition process, based on providers.

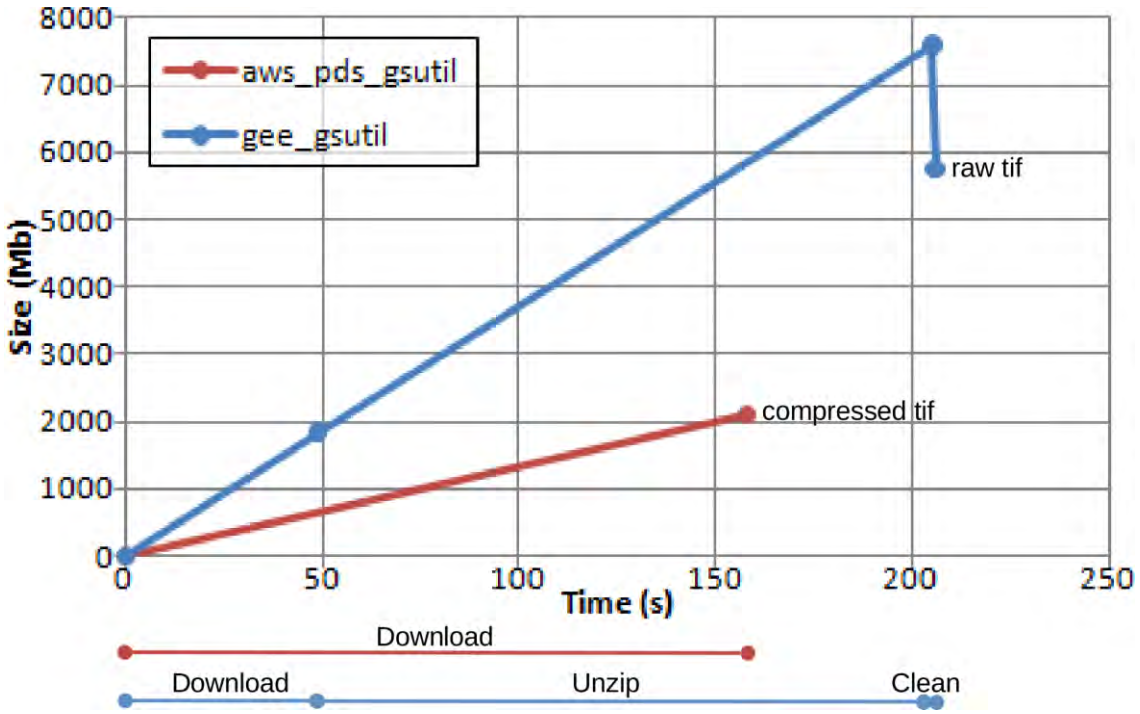

Figure 60: GEE vs. AWS Acquisition Phases

The size of acquired bands differs between providers (Figure 60). This means that for a given scene the size of downloaded data (zipped or not) and of the acquired band will not be comparable, as well as any download speed issued from these values. Consequently, the total time needed to get "ready to work" bands for a given set of scenes was used as a performance parameter (158 and 206 seconds in Figure 60). In order to compare protocols independently of the size of the dataset, total time of each protocol in each round of test was normalized by the minimum total time of the session (Total Time Ratio). In the Figure 60, the minimum time was 158s, then aws_pds_gsutil ratio was 1, and gee_gsutil was 1.3 (206/158). This way, it is possible to answer to the trivial question "what is the faster (provider, access method) combination to access a dataset?"

7.3.8.1. Local Machines

The described benchmarking tests were performed on a server located in the University of Geneva Network (1 Gb connection) on a Xen Virtual Machine (2.9 GHz, 8 Gb RAM, 4cpus). All protocols were tested in random order every 3 hours between the 18.2.1016 and the 25.2.2016. Each protocol was then tested 56 times, 140 scenes were acquired for a total of 409 Gb of downloads. The total execution time of the performed tests (per provider and session) is shown in Figure 61, Figure 62, and Figure 63.

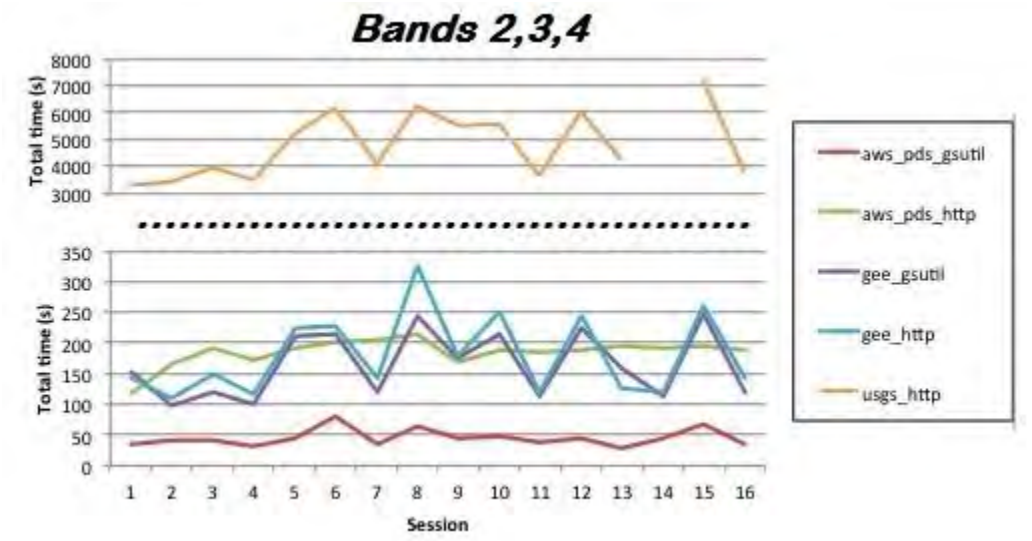

Figure 61: Landsat 8 Data Acquisition - Bands 2,3,4 - Execution Time

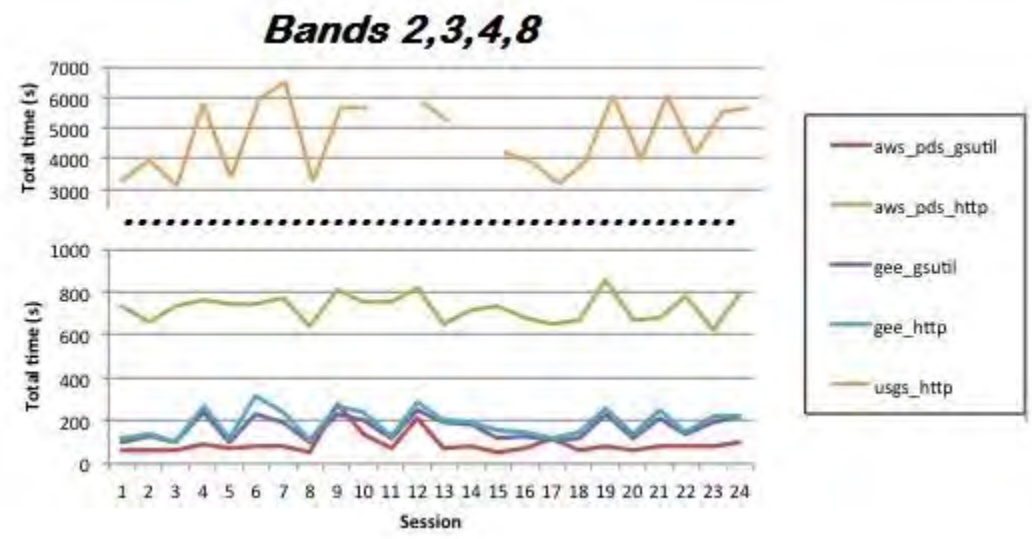

Figure 62: Landsat 8 Data Acquisition - Bands 2,3,4,8 - Execution Time

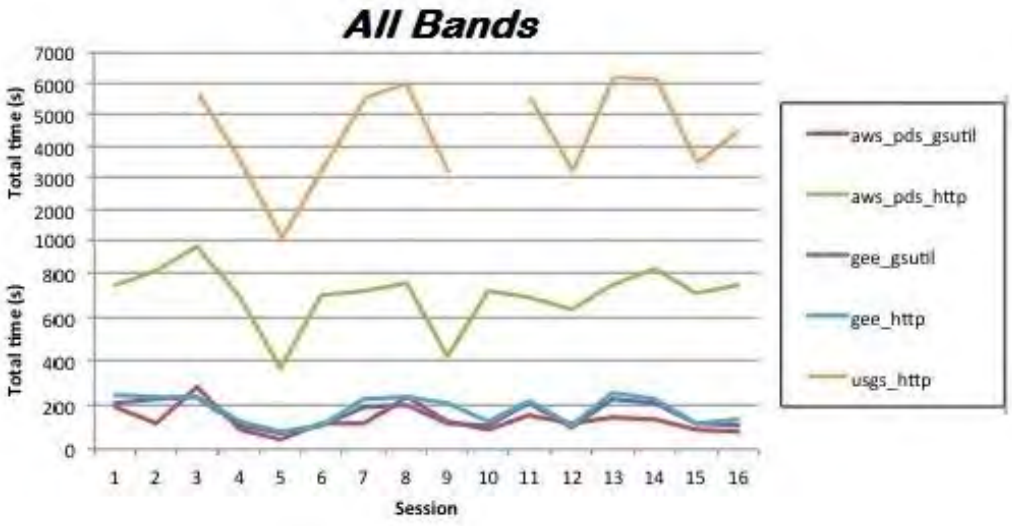

Figure 63: Landsat 8 Data Acquisition - All Bands - Execution Time

Considering that the USGS performance is really poor, having all the providers with the corresponding access methods together in one graph is not showing clearly the difference between all the combinations (provider, access method). For that reason we have created interrupted graphics (marked with black dots). This interruption can be seen on the vertical axis, were we have different time intervals.

As seen in the graphics, gsutil has a better connection as http is always less effective and unstable (notice the break in the usgs_http line (orange), in all graphics, indicating an interruption). The USGS protocol is by far the slowest option and data acquisition process was interrupted at different times.

GEE bandwidth is by far the fastest but the gains from the increased download speed are lost when unzipping and extracting the selected bands. Moreover, as the provided geotiff images are not compressed, storage is not optimized compared to data provided by AWS (Figure 60).

The benefits of using the presented sets of (provider, access methods) for retrieving Landsat 8 data are related first of all with the fast and automatic way of accessing the data but also to the cost reduction of not storing large amounts of data in personal storages. The problem that we might see in the case of an automatic retrieval of data is associated with the changes that might appear in the API access or in the structure of the Landsat 8 archive.

After analyzing and comparing the benchmark results, our conclusion is that USGS protocol should be used as a last option when imagery is not available through other protocol

(typically within the 24 hours following the data acquisition). Depending on the specific needs, or the type of application as well as on other factors, the choice of the data provider and associated data access method may vary. Another interesting possibility would be to also implement a combinations of more access methods from different providers and have a Mediator Component able to choose the best one, depending on availability, use case or preferences.

The presented benchmark tests were performed as part of a bigger initiative for automatic monitoring of land cover changes using satellite imagery called Live Earth Monitoring System (LiMES) that will be presented in coming publications. The results will be further used to implement the automatic data acquisition component, which is vital for obtaining effective and efficient processing results.

7.4. Lessons Learned

The applications from Environmental Sciences usually target large geographical areas, modeled at high resolutions and require a large amount of geospatial data so that the parallelization methods would spread not only on computational parallelism but also on data parallelism. The geospatial information is very important and yet a big issue in many fields due to the large amounts of data that has to be analyzed and processed for obtaining meaningful information, which can be used further on in geospatial applications. These applications provide the necessary functionality for handling geospatial data but they lack the computational and the storing resources when running on desktop stations or on the Web. An important goal in Environmental Sciences is to handle geospatial information using geospatial services and applications and to exchange this information on a distributed environment and in an interoperable and efficient way. For this reason, it needs fast and reliable tools and components to remotely access geo-referenced data. Desktop computers are no longer sufficient for large-scale applications. Parallel and distributed infrastructures such as Multicore, cluster, Grid, Cloud, etc. seem to provide the necessary functionalities for solving most of the problems related to the geospatial data: handle complex computations through parallelism, both at data level and processing level, support data management and data security. Depending on application features, data model, processing requirements, environment and resource availability, user requirements, etc., one of such infrastructures could be more appropriate than other ones. To be able to benefit from all these capabilities, a good and easy to use methodology is needed to be able to port the

necessary geospatial applications and services to parallel and distributed infrastructures and to obtain high performances. We need to analyze the conditions in which a specific parallel and distributed infrastructure is efficient and optimum for a certain application. An optimum environment can be chosen from different perspectives: from the user point of view, an optimum environment might be the one that gives the best execution times and offers the best guarantee that the execution will complete successfully while from a developers point of view, an optimum environment might be the one that requires minimum application changes and minimum/balanced usage of resources.

The experiments performed in this chapter were not meant necessarily to improve the applications performances, although in most of the cases we have achieved this, but rather to observe the behaviour of each infrastructure and to understand what are their strengths and their weaknesses.

To summarize, the following are the main lessons learned from our set of performed experiments. These lessons are the foundation to our next chapters:

- Environmental applications are characterized by longevity (typical long life time) and in many cases large scale.
- Environmental applications experience a high degree of heterogeneity at different levels: interfaces, metadata and data models, data providers, formats, coordinate reference systems, resolutions, ontologies, etc.
- The heterogeneity and interoperability issues in environmental applications can be addressed using two general approaches:
    - Standardization – defining common specification for interfaces, metadata and data models. This is a long, slow process, which requires commitment and adoption from the participating systems, high ICT expertise, and most of the times complex specifications.
    - Mediation – adapting and harmonizing heterogeneous interfaces, metadata and data models. This is possible only if the harmonization and adoption is theoretically and practically feasible. This implies that distinct data models must be mapped to a higher-level conceptual model, able to reconcile the heterogeneous implementations.

- The rapid growth of the volume of environmental data requires an improvement in the efficiency of acquisition and processing. Real-time, rapid and efficient access and processing of massive amount of data has become a necessity. There are several ways of improving the efficiency, among which we mention the optimizations of the processing algorithms and techniques and the execution of these algorithms and techniques on parallel and distributed processing environments (Sun et al., 2013 [219]).
- Long term benchmarking of the Landsat 8 satellite data accessibility showed efficient protocols exists to access the data easily, effectively, and consistently. As with any automated workflow, any change in the way data are provided (standards or API) or a server interruption would immediately lead to a failure. Anyhow the large panel of data providers as well as the numerous access methods, guarantee data availability at any time with the incorporation of failover heuristics into the harvesting script giving it the ability to switch between protocols in case of non-response. Data acquisition can also take advantage of parallel and distributed infrastructures, especially for storing large amounts of downloaded data (before being processed) and retrieving the data in parallel in case several data providers are involved.
- The use of computational resources from different distributed infrastructures can be achieved through pluggable components, which can access each composing infrastructure connected to a central manager able to coordinate the scheduling of jobs on these resources, provisioning on resources and adapted execution depending on the type of chosen resource.
- Porting applications and services to parallel and distributed infrastructures is a very difficult task to accomplish since each platform requires particular details to take into account when integrating applications that have not been designed to run on that platform in the first place. These applications have to be modified to have a particular structure or to use particular programming API for accessing the resources from each individual architecture, without knowing too much details about the running platform.
- Fusion tools and services to merge different heterogeneous data sources should include functionality such as: processing, feature extraction, situation assessment, modelling and prediction services, preparing and aggregating environmental data into formats suitable for human use and automated services.

- Big Data should be kept close to the producer and the moving of voluminous data representing geospatially distributed fields over networks should be minimized.
- A data mediation layer is necessary to convert data from sophisticated formats of Environmental Sciences to more familiar formats of the software development community.

7.5. Personal Contributions

- Execute different environmental applications on different distributed infrastructures:
    - SWAT calibration on gLite (Grid) vs. Multicore (UTCN);
    - SWAT calibration on Baobab cluster (UNIGE);
    - SWAT calibration on OpenStack Cloud (UNIGE - HEPIA);
    - SWAT calibration on OpenStack Cloud (UNIGE - SwissACC);
    - SWAT calibration on Windows Azure Cloud (UNIGE - SwissACC);
    - Global Flood Model on Baobab cluster (UNIGE);
    - OGC services Gridification on gLite (Grid) (UTCN);
    - Landsat 8 Data Acquisition on local servers (UNIGE - UNEP/Grid-Geneva).
- Observe the behavior of each infrastructure and understand what are their strengths and their weaknesses in porting environmental applications.
- Explore alternative solutions to connect different computing backend - GC3Pie tool, provided by the GC3 team (ETH – Zurich).
- Published Papers:
    - ***Rodila, D.,*** *Bacu, V., and Gorgan, D. (2012). Comparative Parallel Execution of SWAT Hydrological Model on Multicore and Grid Architectures, in International Journal of Web and Grid Services (IJWGS), Vol. 8/3, September 2012, pp.304 – 320, http://dx.doi.org/10.1504/IJWGS.2012.049172.*
    - ***Rodila, D.,*** *and Gorgan, D. (2012). Geospatial and Grid Interoperability through OGC Services Gridification, in International Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS), Vol. 5/6, December 2012, pp. 1650 – 1658, ISSN: 1939-1404, http://dx.doi.org/10.1109/JSTARS.2012.2217115.*

- *Silvestro, F., Campo, L., Rudari, R., De Angeli, S., D'Andrea, M.,* ***Rodila*****, D.,** *and Gabellani, S. (2016). Impacts of EC-Earth Global Climate Model RCP4.5 climate change scenario on maximum daily streamflow quantiles at global scale, Journal of Climate (submitted article).*
- *Bacu, V., Mihon, D., Stefanut, T.,* ***Rodila*****, D.,** *Abbaspour, K., Rouholahnejad, E., and Gorgan, D. (2013). Calibration of SWAT Hydrological Models in a Distributed Environment Using the gSWAT Application, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 66–74, ISSN 2158-107X.*
- *Gorgan, D., Bacu, V., Mihon, D.,* ***Rodila*****, D.,** *Abbaspour, K., and Rouholahnejad, E. (2012). Grid based calibration of SWAT hydrological models, in Journal of Nat. Hazards Earth Syst. Sci., Vol. 12/7, pp. 2411-2423, http://dx.doi.org/10.5194/nhess-12-2411-2012.*
- *Mihon, D., Bacu, V.,* ***Rodila*****, D.,** *Stefanut, T., Abbaspour, K., Rouholahnejad, E., and Gorgan, D. (2012). Grid Based Hydrologic Model Calibration and Execution, Chapter in the book: Advanced in Intelligent Control Systems and Computer Science, Dumitrache I. (Ed.), Springer-Verlag, Vol. 187, pp. 279-293, ISBN 978-3-642-32548-9, http://dx.doi.org/10.1007/978-3-642-32548-9_20.*
- ***Rodila*****, D.,** *and Gorgan, D. (2012). Mapping Geospatial Applications onto Parallel and Distributed Environments, in ePaMuS 2012 – 5th International Workshop on Engineering Parallel and Multi-Core Systems – The Multi-Core Workshop, Palermo, Italy, 4-6 July, 2012, pp. 443 – 448, http://dx.doi.org/10.1109/CISIS.2012.152.*
- ***Rodila*****, D.,** *Bacu, V., and Gorgan, D. (2011). Comparative Analysis of Distributed and Grid Based Execution of SWAT Model, in 3PGCIC 2011 - Sixth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Barcelona, Spain, October 26-28, 2011, pp. 273-278, http://dx.doi.org/10.1109/3PGCIC.2011.49.*
- ***Rodila, D.,** and Gorgan, D. (2011). A Mediation Approach in Geospatial Web Services Gridification, in ICCP2011 – IEEE International Conference on*

Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 25-27, 2011, pp. 541-548, http://dx.doi.org/10.1109/ICCP.2011.6047928.

- ***Rodila, D.,** Bacu, V., Ardelean, V., Borlea, C., and Gorgan, D. (2011). Geospatial Web Services Gridification in enviroGRIDS, in European Geosciences Union - General Assembly EGU 2011, Vienna, Austria, April 03-08, 2011, (abstract and presentation), http://meetingorganizer.copernicus.org/EGU2011/EGU2011-11469.pdf.*
- *Bacu, V., Mihon, D., **Rodila, D.,** Stefanut, T., and Gorgan, D. (2011). gSWAT Platform for Grid based Hydrological Model Calibration and Execution, in ISPDC 2011 - 10th International Symposium on Parallel and Distributed Computing, Cluj-Napoca, Romania, July 6-8, 2011, pp.288-291.*
- *Gorgan, D., Bacu, V., Mihon, D., Stefanut, T., **Rodila, D.,** Kokoszkiewicz, L., Rouholahnejad, E., Abbaspour, K., and van Griensven, K. (2011). Grid Based Hydrological Model Calibration and Execution by gSWAT Application, 2011 International SWAT Conference, 15-17 June, 2011, Toledo, Spain, 2011.*
- *Bacu, V., **Rodila**, **D.,** Kokoszkiewicz, L., Rouholahnejad, E., Mihon, D., van Griensven, A., Abbaspour, K., and Gorgan, D. (2011). Calibration and Execution of SWAT Models over GRID Architecture, Presentation to ESSI-12, EGU 2011, Vienna, 3-8 April, 2011.*
- ***Rodila, D.,** and Gorgan, D. (2010). Integration of Spatial Data Infrastructures with Grid Environment, in SYNASC 2010 – 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, September 23-26, 2010, pp. 269-277, http://dx.doi.org/10.1109/SYNASC.2010.63.*
- ***Rodila, D.,** Gorgan, D., and Bacu, V. (2010). The Interoperability between OGC Services and Grid Environment in EnviroGRIDS Project, in 3PGCIC 2010 – International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Fukuoka, Japan, November 4-6, 2010, IEEE Computer Press, pp. 387-392, ISBN: 978-0-7695-4237-9, http://dx.doi.org/10.1109/3PGCIC.2010.65.*
- ***Rodila, D.,** Chatenoux, B., and Giuliani, G. (2016). Landsat 8 resources - a short comparison of different data access providers and methods (submitted paper).*

New Methodology and Framework Proposal

Chapter 8: Conceptual Description of Environmental Applications

8.1. Introduction

The computing industry has experienced a sustained exponential growth in processor performance until recently by scaling clock frequency and enhancing uni-processor micro-architecture (Raman, 2011 [182]). Since 2004, however, uni-processor performance has grown at a much slower pace. Increasing design complexity, power and thermal constraints, and diminishing returns from uni-processor micro-architectural enhancements are the primary reasons for the slowdown. Meanwhile, Moore's law continues to be followed with a doubling of the number of transistors per unit area approximately every two years. Processor designers leverage these additional transistors by placing multiple cores on the same die (Raman, 2011 [181]). Parallel execution resources available for the execution of an application may vary from different number of cores or computing nodes to different sizes of memory or storage nodes. The performance goal of an application may also vary, depending on different functions such as cost, throughput, etc. Another variable item is the program's workload. All these elements of variability compose the execution environment of a program (Raman, 2011 [181]). To be able to efficiently execute an application, we have to adapt it's structure to this variable execution environments in a flexible manner which implies that the program should not be encoded with a single static parallelism configuration (Suleman et al., 2010 [218], Raman et al., 2011 [182]).

To our knowledge there is no convenient tool/framework to allow a user to easily express and control the execution of an environmental application in heterogeneous computing environments, without having expertise in sophisticated workflow systems or control of the backend functionality. The main goal of this section is to fill that gap and to propose a conceptual model of environmental applications, which will be a key component in a general methodology for porting these applications on different parallel and distributed infrastructures. The conceptual model facilitates and simplifies not only the understanding of the application structure but also the general execution on different computational platforms. It provides a platform-independent, robust, convenient and easy way to use a mechanism that allows a user to execute an application on a heterogeneous computing environment, and as such provides a first

step towards the automation of this process. The execution of the applications and the selection of the computing environment(s) can be done automatically by an intelligent component, based on a complete conceptual model as well as on application related information provided by the user and other useful information such as availability of computing environments, previous execution history of the application, etc.

8.2. Conceptual Models

Conceptual modelling is an activity of formally describing properties as well as actions of the physical and social world, having as purpose a better understanding, communication and visualization.

The descriptions that arise from conceptual modeling are meant to be used both by humans and machines. The concept of conceptual modeling was first associated to semantic data modeling but soon it has found applications in many other fields such as modeling organizational environments, modeling software development processes or even modeling different parts of the world for better human communication and understanding (Mylopoulos, 1992 [158]).

Conceptual models, mostly graphic, are used to represent both static and dynamic phenomena and they usually play an important role in communication between developer and user, in understanding of a new domain, they provide a good documentation and provide input in the design process. High quality conceptual models enable also early detection and correction of errors (Wand and Weber, 2002 [234]).

A model is in fact an approximation, with different levels of details, of the real world systems. It is a physical, mathematical or logical representation of a system, phenomenon or process and serves as a representation of an event/thing that is real or deliberately created. A model is thus produced by abstracting from reality a description of the system, with the observation that not all the aspects of the system are represented, as that would be too timely, complex and expensive. A model can then represent the system at some point of abstraction or at different levels of the abstraction, depending on the requirements but also on the modeler decisions (which can be challenging) and having the purpose to represent the system in a reliable way (Sokolowski and Banks, 2010 [217]). Modeling means making a simplified logical representation of a real world environment or system. The model is not an identical replication of a real-world system considering that not all the information detailing the environment is

considered, rather it is a plausible representation of the parts of the system that matter the most for a specific purpose.

Considering that Environmental Sciences is a complex and interdisciplinary domain, conceptual models are useful methods for facing these challenges of deep understanding of the studied phenomena. Conceptual models are useful in improving the coherence and analyzing the environmental issues and integrating knowledge. They help the user to understand the complexity of environmental systems but also the variety of scientific approaches there exists to formulate and solve the environmental problems (Fortuin et al., 2011 [72]).

8.3. Environmental Applications Conceptualization

8.3.1. Definition

Scientific applications typically use two types of parallelism to get more benefits from large computing systems (Bochenina, 2014 [29]):

- Task parallelism – different tasks of an application workflow can be executed concurrently;
- Data parallelism – each task can be executed on more than one processor at a time.

Task parallel application workflows are usually represented as DAG (Direct Acyclic Graph), which represents the dependency among tasks, based on their execution time and communication time. In a DAG, the vertex or the node weight represent task processing time and the edge weight represent data dependencies and communication time (also known as communication cost) between tasks. A DAG is usually represented as (Amalarethinam and Josphin, 2015 [8]):

$$G = (V, E)$$

Where:

- V is a set of nodes / vertices;
- E is a set of directed edges.

The source node of an edge is called parent node and the sink node is called child node. A node with no parent is called entry node while a node with no child is called an exit node. The

critical path is a series of tasks that must be completed for an application execution to finish on schedule. Each task on the critical path is a critical task.

A DAG example is illustrated in Figure 64. It contains 10 Tasks, T1 is the entry node, T 10 is an exit node with no child node and T3 is a parent node (for T6 and T7).

DAG representations are frequent for complex applications and the composing tasks (with the associated dependencies) require advanced scheduling procedures that must consider QoS requirements.

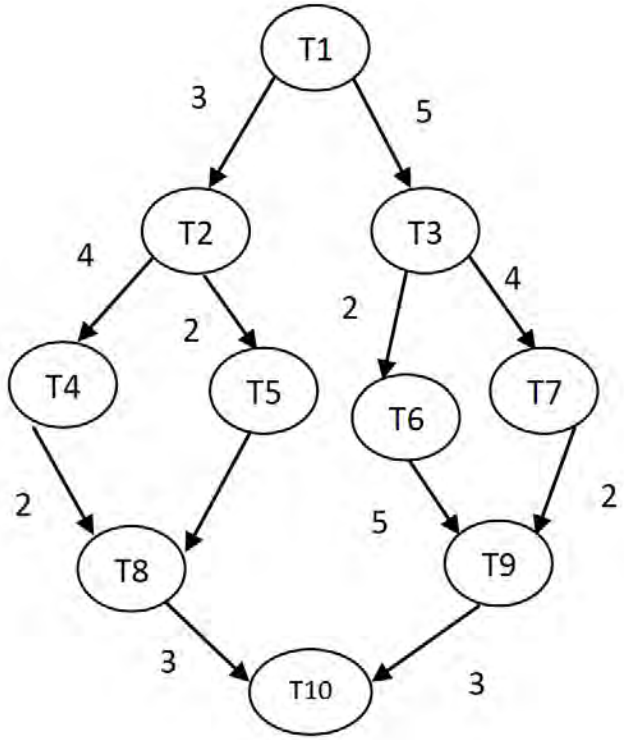

Figure 64: DAG sample

8.3.2. Formal Description of an Environmental Application

Based on our conceptual model, an environmental application can be described using the following notation:

EnvA = (I, E, S, SEnvA, LE)

where:

- $I = \{I_1, I_2, \ldots, I_n\}$ - Input data set, n >= 0;
- $E = \{E_1, E_2, \ldots, E_m\}$ - Executions (Processes) set, m >= 0;

- $S = \{S_1, S_2, \ldots, S_p\}$ - Services set, p >= 0;
- $SEnvA = \{SEnvA_1, SEnvA_2, \ldots, SEnvA_r\}$ – Sub-applications set, r >= 0;
- $LE = \{LE_1, LE_2, \ldots, LE_s\}$ – Loop Executions set, s >= 0.

The input elements are represented of environmental data that can come from different providers, in different formats and different sizes. These elements can represent the inputs of executions, of loop executions, of services, or of sup-applications.

Each execution and loop execution has associated a set of pre-conditions and a set of post-conditions. The set of pre-conditions is composed from the set of inputs necessary for that execution and the set of processing requirements necessary for running that execution. A valid execution must have valid pre-conditions. The set of post-conditions is usually composed from the set of outputs of that execution and possibly the specifications on where to store the output.

- $Pre(E_i) = \{I_1, I_2, \ldots | I_i$ is input for $E_i\}$ U $\{PR_1, PR_2, \ldots | PR_i$ is processing requirement for $E_i\}$
- $Post(E_i) = \{O1, O2, \ldots | O_i$ is output for $E_i\}$

The schematic representation of a conceptual model, using the above-descried components is the following:

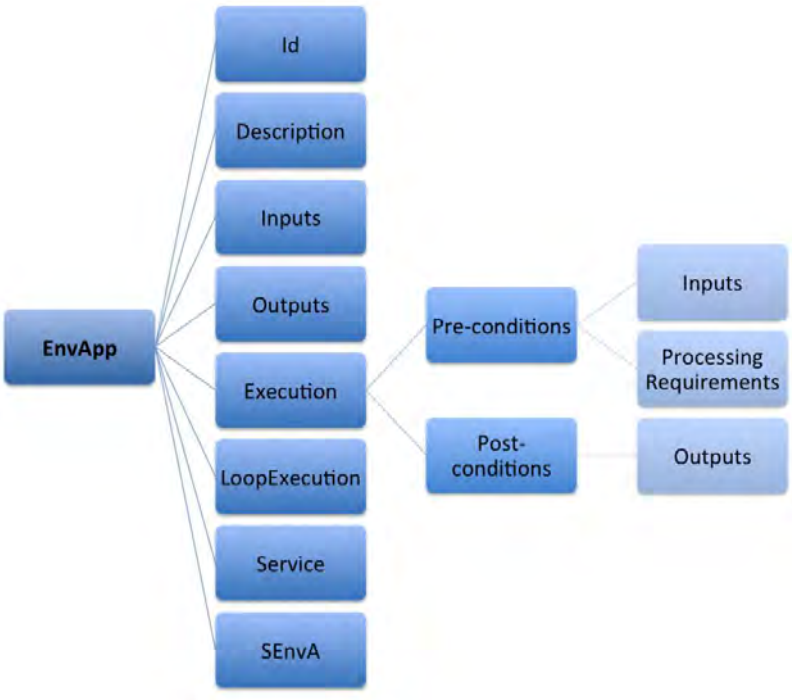

Figure 65: Conceptual Model Structure

Each component of the conceptual model is stored in the database with a list of characteristics:

- Each **input** has associated the following information:
  - *Id* – unique identified;
  - D*escription* – a short description of the input;
  - *Type* – the *type* of the input (which can vary from a satellite image to an environmental statistical indicator);
  - *Location* – specifies where is the input stored (which infrastructure and what element). In the case of a Grid infrastructure, this can represent the address of a Storage Element (SE) and the path inside that SE. In case of a Cloud, the address of the Cloud Storage can be given, etc. In the case the input represent the output from another execution in the application, the location is the id in the database of the output element.
  - *Access mode* – the list of possible credentials to be able to access the input from the given location.
- An **Execution** is defined in the database using the following information:
  - *Id* – unique identifier of the execution;
  - *Description* – short description of the performed execution;
  - *Inputs* – the set of inputs;
  - *Outputs* – the set of outputs;
  - *NrOfLoops* – attribute to specify the number of loops in case of a loop execution. For normal executions this is normally 1;
  - *CPU* – processing requirement needed to run the execution;
  - *Memory* – processing requirement needed to run the execution;
  - *Observations* – field used to describe additional observations regarding the execution.
- The Sub-Application element (**SEnvA**) allows the insertion of an already executed application (flow) within another workflow. This element is described using:
  - *Id* – unique identifier of the sub-application;
  - *Description* – short description of the sub-application;

- *Inputs* – the initial inputs of the inserted sub-application;
- *Outputs* – the list of outputs;
- *CPU* – processing requirement needed to run the sub-application;
- *Memory* – processing requirement needed to run the sub-application;
- *Observations* – field used to describe additional observations regarding the execution of the sub-application.

- The **Service** element is used to describe an execution performed by a Web or Geospatial service. For this, we use the following attributes:
  - *Id* – unique identifier in our database;
  - *Description* – short description of the service;
  - *Address* – the access address of the service;
  - *Request* – the execution request at the specified address;
  - *Inputs* – the initial inputs of the service request;
  - *Outputs* – the list of outputs;
  - *CPU* – processing requirement needed to run the service;
  - *Memory* – processing requirement needed to run the service;
  - *Observations* – field used to describe additional observations regarding the execution of the service.

The language used to describe the presented conceptual model for an environmental application is an extension of the PDGL (Process Description Graph Language) used in the gProcess platform (http://cgis.utcluj.ro/applications/gprocess, Bacu et al., 2009 [15], Rodila et al., 2009 [194]). This language allows the definition of additional control flows, compared to a simple DAG. It allows the definition of the environmental application conceptual model, which is an abstract description, but it also allows the description of an instantiated conceptual model, in which we can map all the information of the application and of the execution of the application on different computing resources.

8.3.3. Execution Patterns

For the definition of the conceptual model, we have defined a series of execution patterns that can be used:

8.3.3.1. Simple Execution

The execution flow in this case is a simple one in which the user defines the input(s) that are entries for a single execution point producing some output(s). The inputs and the outputs can be of different types and can be specified in different formats.

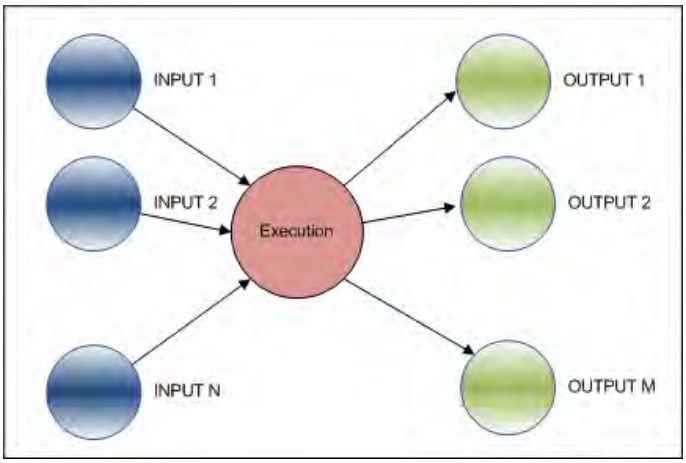

Figure 66: Conceptual Model - Simple Execution

8.3.3.2. Sequential Execution

In this case the execution flow is modeled as a sequence of several executions. The execution of a step normally depends on the results of a previous execution. That is why synchronization has to be taken into consideration. Simple Execution is a particular case of Sequential Execution in which we only have one execution node.

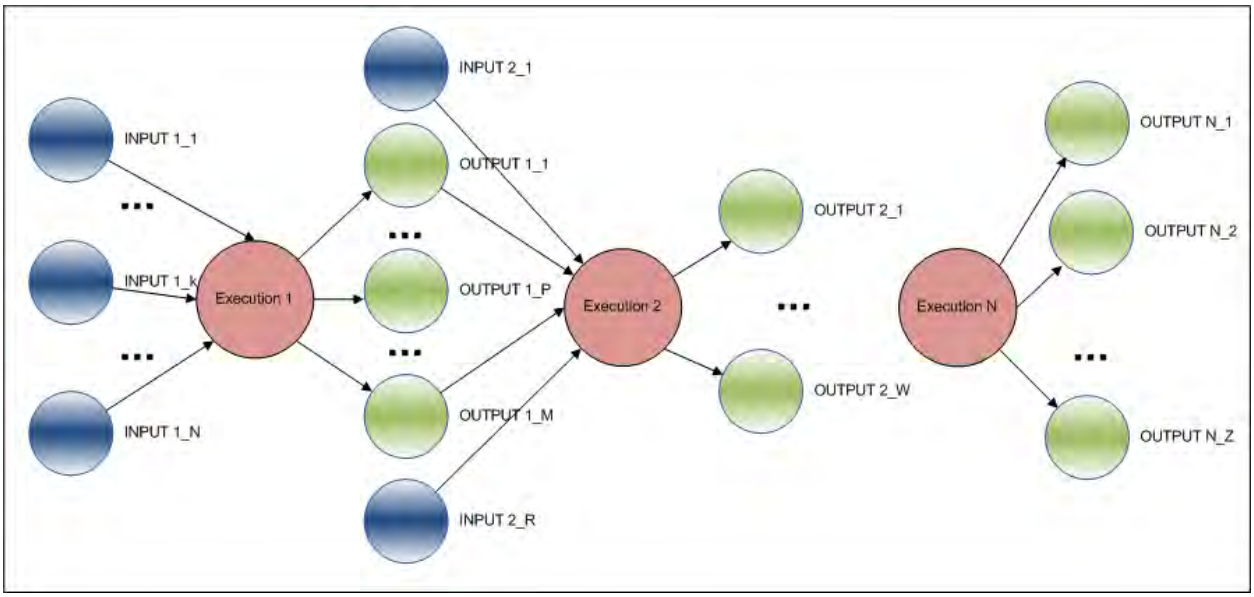

Figure 67: Conceptual Model - Sequential Execution

8.3.3.3. Parallel Execution

The execution flow in this case is composed of several executions that are run in parallel. Each step is independent and can be executed concurrently with the others. A "Simple Execution" is also a particular case in which we only have one execution node.

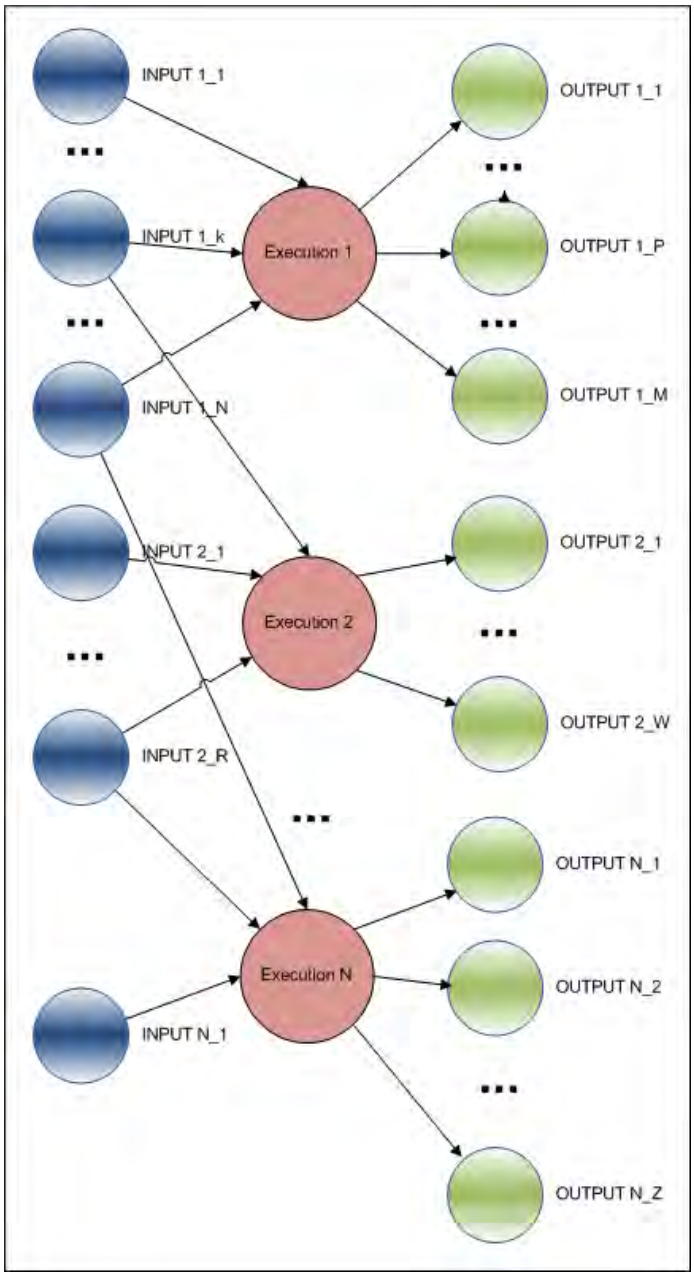

Figure 68: Conceptual Model - Parallel Execution

8.3.3.4. *Composed Execution*

The composed execution has a complex structure in which one can include different types of executions: simple, sequential or parallel. This is useful if a user wants to save/use previously computed executions without having to define them again. All other mentioned cases: Simple Execution, Sequential Execution and Parallel Execution can be considered particular use cases of this type.

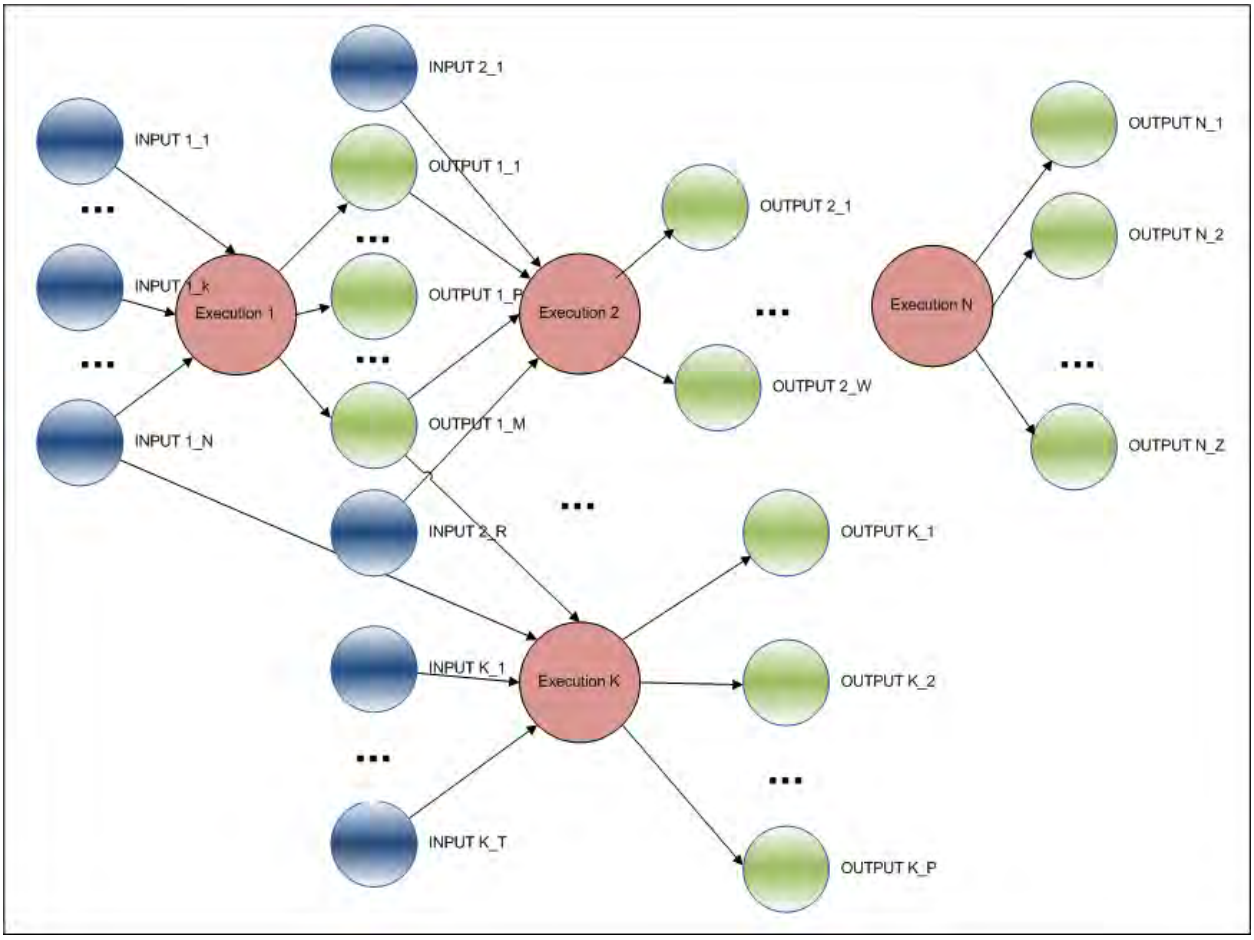

Figure 69: Conceptual Model - Composed Execution

8.3.3.5. *Loop Execution*

The execution flow in this case consists in executing the same module several times. The module can be composed of several types of executions or it can be one of the already presented types. This is useful when the same set of executions has to be repeated several times. An example of this case can be the calibration of a climatic model. The inputs can be the same set of parameters or a slightly different one, but the outputs are usually different.

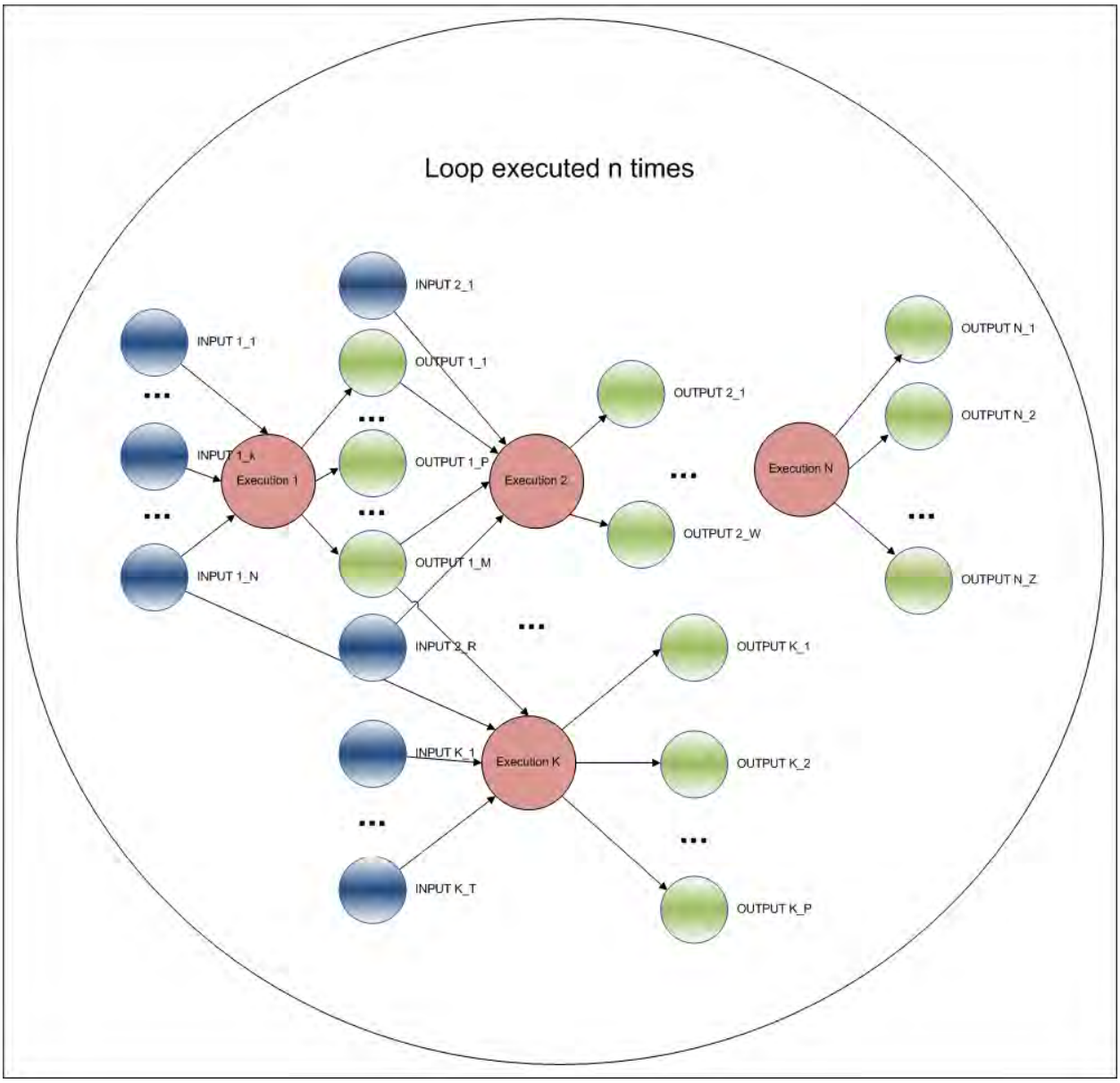

Figure 70: Conceptual Model - Loop Execution

8.3.4. Methodology

The proposed conceptual model covers all the above described use cases. Using this model, a user can easily describe the structure and the execution flow (workflow) of his/her application. The actual execution of this model can be done through instantiation, i.e. binding the workflow tasks to specific resources (different for each application and for each execution use case). The conceptual model provides a flexible way of specifying an environmental application

without being concerned with the low-level implementation details. The tasks in the conceptual model can be mapped on any executable platform at run time using mapping mechanism. In a concrete model, the specific resources of the applications are bind to the tasks. At this level, new tasks may also appear, related to data movement between tasks and/or repositories. The concrete model can be generated (either full or partial) either before or during the execution.

The steps to complete the conceptual model for a specific application are the following ones:

1. Define all the inputs of the application. Here the user has to specify for each input what is its type, how it can be accessed and to which execution task it belongs. The inputs can either be initial inputs or they can also be outputs from other executions.
2. Define all the outputs of the application by specifying as well their type, where should they be stored and from what execution they come from.
3. Define the executions tasks within the applications. Depending on what inputs are associated with a specific task, we can decide if the task will be executed in parallel or sequentially with other tasks. The loop executions are specifically described within a loop tag in the file in which the user has to specify what executions are parts of the loop and how many times the loop is repeated.
4. Define the Composed Executions if any. At this point the user can specify the path to an already defined conceptual model (sub-application) of a previous application.

The parser intended to process the defined conceptual model will explore all these pieces of information. The conceptual description can be used in general for any type of application so far but the specificity of the environmental science field will be modeled in the parser component, as this is the level where the differences appear concerning especially the input and output data, as well as the algorithms used to handle environmental data.

8.3.5. Examples

A general approach to port an application to a computing environment would include the following steps:

- **Preprocessing phase** (in which the framework gathers the information about the available computational system, the executed application, the user preferences, etc. In this step we define the application conceptual model as well);
- **Analysis phase** (in which we model both the application structure and the execution flows, based on the gathered information in the previous phase);
- **Decision phase** (in which we make decisions on which computing environments in more suitable, based on the application developed model, availability of the platforms, user preferences, history, predictions …);
- **Execution phase** (in which we actually execute the application on the selected computing backend, based on the defined policies);
- **Monitoring phase** (in which the execution is monitored and information is collected for future executions).

The performed practical experiments as well as the knowledge gathered after reviewing the scientific literature in this area formed together the starting point and the foundation on which the conceptual model, for describing a general environmental application, was built. The conceptual description contains specific details of the mapped application, such as: name, description, initial and intermediary inputs, outputs, executable processes, cost associated with each execution, etc. All these details are stored in a file and are used not only in determining the structure of the application but also the execution flow (i.e. control flow, specifying the order of the activities to be executed) and the data flow (specifying the input and the output data for each activity/task to be executed). Having this information in a common standard way is a step forward to automatize the mapping of applications on different computing infrastructures.

The conceptual model structure is presented in Figure 71:

```
<EnvironmentalApplication>
          <Input idDB="" name="" description="">
               <PreConditions>
               ...
               </PreConditions>
               <LocalPath path="" />
               <PostConditions>
                    <Output idDB="" />
               </PostConditions>
          </Input>
          ...
          <Service idDB="" name="" description="">
               <PreConditions>
                    <Input idDB=""/>
                    <Input idDB=""/>
                    ...
               </PreConditions>
               <URL path=""/>
               <PostConditions>
                    <Output idDB="" />
               </PostConditions>
          </Service>
          ...
          <Execution idDB="" name="" description="">
               <PreConditions>
                    <Input idDB=""/>
                    <Input idDB=""/>
                    ...
               </PreConditions>
               <PostConditions>
                    <Output idDB="" />
               </PostConditions>
          </Execution>
          ...
          <SubApp idDB="" name="" description="">
               <PreConditions>
                    <Input idDB=""/>
                    <Input idDB=""/>
```

```
                    ...
               </PreConditions>

               <PostConditions>
                    <Output idDB="" />
               </PostConditions>
          </SubApp>
          ...
          <Loop idDB="" name="" description="" idDB="">
               <Execution idDB=""\>
               <SubApp idDB=""\>
               <Iterations nr=""/>
          </Loop>
          ...
</EnvironmentalApplication>
```

Figure 71: Environmental Applications Conceptual Model Structure

An example of conceptual model for the SWAT hydrological model:

```
<EnvironmentalApplication>
          <Input idDB="1" name="BSInst11" description="initial
IO directory of the SWAT model, instance 11">
               <PreConditions>
               </PreConditions>
               <LocalPath
path="/home/denisa/SWAT/BSInst11/InOut/" />
          </Input>
          <Execution idDB="1" name="pre-processingBSInst11"
description="preprocessing phase of the SWAT model, instance
11">
               <PreConditions>
                    <Input idDB="1"/>
               </PreConditions>
               <URL
path="/home/denisa/SWAT/BSInst11/Executable/Pre-processing/"/>
               <PostConditions>
```

```
                    <Output idDB="1", name="BSInst11.tgz"
description="generated calibration archive of the SWAT instance
11" />
               </PostConditions>
          </Execution>
          <Execution idDB="2" name="iterationBSInst11"
description="one iteration of the SWAT model, instance 11">
               <PreConditions>
                    <Input idDB="1"/>
               </PreConditions>
               <URL path="/home/denisa/SWAT/
BSInst11/Executable/Iteration/"/>
               <PostConditions>
                    <Output idDB="X", name="BSInst11-ItX.tgz"
description="Iteration X result archive" />
               </PostConditions>
          </Execution>
          <Loop idDB="1" name="BCInst11Calibration"
description="calibration execution phase of the SWAT model,
instance 11">
               <Execution idDB="2"\>
               <Iterations nr="100"/>
               <PostConditions>
                    <Output idDB="Y", name="BSInst11-OUT.tgz"
description="Calibration result archive" />
               </PostConditions>
          </Loop>
          <Execution idDB="4" name="post-processingBSInst11"
description="post-processing phase of the SWAT model, instance
11">
               <PreConditions>
                    <Input idDB="Y"/>
               </PreConditions>
               <PostConditions>
                    <Output idDB="", name="BSInst11-final.tgz"
description="final results of the SWAT instance 11" />
               </PostConditions>
          </Execution>
</EnvironmentalApplication>
```

Figure 72: SWAT Conceptual Model - Use Case

8.3.6. Conclusions

Our planet is a complex and interconnected system that can be analyzed at many levels of details. To be able to understand this system as well as the relationships behind it and to be able to answer the ever-growing list of questions regarding all the changes that we are experiencing, research has to be done on data at all these levels. Analyzing Big Data has become possible recently both due to the increasing capabilities of computational resources (and hardware advances) and to the availability of tools, algorithms and techniques used to take advantages of these resources. As the computational power increases, more accurate and efficient methods will be developed.

The environmental datasets, beside containing data of different formats and types, are distributed and stored by different organizations so these large amounts of data are neither easy to find nor to collect and interpret. The complexity and heterogeneity of data formats leads also to data interoperability and data usability problems. The description of a conceptual model, able to describe a general Environmental Sciences application, had to take into consideration all these aspects. The data is heterogeneous and the interpretation of data differs from each resource and scientist to another. Considering this, ontologies and semantic web technologies have also a great potential to provide the required semantic interoperability (Parekh, 2005 [175]).

Allowing easy integration of environmental applications with high performance computing resources can greatly help tackling the many challenges and questions that Environmental Sciences are facing. This can lead environmental scientists to better take advantage of the powerful information that geospatial data can offer in order to scientifically influence environmental management decisions. To that end we have introduced in this paper a solution to easily model environmental applications and to facilitate their integration with different parallel and distributed environments.

Taking into account the growing need for computational speed, storage and scalability, that environmental applications demand, the users usually tend to use or to switch more than one execution platform for obtaining the necessary resources. To be able to easily switch between these platforms we have analyzed and we came up with an application conceptual model, which hides the complexity of different types of environmental applications and that provides an easy and flexible way to map an environmental application to an execution platform. Using this model, a user can easily describe the structure, the data flow as well as the execution flow

(workflow) of the application. The aim of the proposed application conceptual model was to provide a platform-independent, robust, convenient and easy to use methodology, which would allow a user to execute an application on a heterogeneous computing environment.

8.4. Personal Contributions

- Propose a conceptual model of environmental applications, based on theoretical knowledge and practical experience gained on Chapter 7. The proposed model is a key component in a general methodology for porting these applications on different parallel and distributed infrastructure.
- Published Papers:
    - ***Rodila, D.,*** *Ray, N., Gorgan, D. (2015), Conceptual Model for Environmental Science Applications on Parallel and Distributed Infrastructures, Environmental System Research, Vol. 4/23, 2015, http://dx.doi.org/10.1186/s40068-015-0050-1.*
    - *Gorgan, D., Mihon, D., Bacu, V.,* ***Rodila, D.,*** *Stefanut, T., Colceriu, V., Allenbach, K., Balcik, F., Giuliani, G., Ray, N., and Lehmann, A. (2013). Flexible Description of Earth Data Processing over HPC Architectures, Big Data From Space Symposium, Frascati, Italy, 5-7 June, Abstract Book, pp. 39.*
    - *Colceriu, V., Mihon, D., Minculescu, A., Bacu, V.,* ***Rodila, D.,*** *and Gorgan, D. (2013). Workflow Based Description and Distributed Processing of Satellite Images, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 50–57, ISSN 2158-107X.*
    - *Bacu, V., Mihon, D.,* ***Rodila, D.,*** *Stefanut, T., and Gorgan, D. (2011). gSWAT Platform for Grid based Hydrological Model Calibration and Execution, in ISPDC 2011 - 10th International Symposium on Parallel and Distributed Computing, Cluj-Napoca, Romania, July 6-8, 2011, pp.288-291.*
    - *Gorgan, D., Stefanut, T., Bacu, V., Mihon, D., and* ***Rodila, D.*** *(2010). Grid based Environment Application Development Methodology, in Large-Scale Scientific Computing, Springer Journal LNCS 5910, pp.499-506, ISBN 978-3-642-12534-8, http://dx.doi.org/10.1007/978-3-642-12535-5_59.*

Chapter 9: Hybrid Computing Environment (HCE)

9.1. Introduction

A large number of scientific and research communities but also business organizations have access nowadays to several parallel and distributed computing infrastructures such as clusters, Grids, Clouds, etc. but they don't necessarily have the required technical expertise to use all of them. Providing an interface between these communities and the available computing infrastructures, which will allow transparent and uniform access to all these sets of heterogeneous resources, seems to become extremely important. To efficiently manage and build such an interface, new technologies, services and methodologies have to be developed to coordinate, manage, schedule and execute scientific applications over the available distributed computing infrastructures, which form a Hybrid Computing Environment (HCE).

9.2. Similar Initiatives

Some of the most important initiatives identified in the literature, focusing on the interoperability between two or more Distributed Computing Infrastructures (DCIs) are presented in the following. Most of them consider only two DCIs, either Grid and Cloud, or Grid and cluster, and rarely they focus on integrating all the available infrastructures.

The European FP7 SHIWA project (SHIWA, 2012 [211]) (Sharing Interoperable Workflows for large scale scientific simulation on available DCIs) aimed to develop new workflow interoperability technologies to allow publicly available workflows to be used by different research communities on different workflows systems and executed on multiple distributed computing infrastructure.

The European FP7 ER-flow project (ER-flow, 2013 [62]) (Building a European Research Community through Interoperable Workflows and Data) collaborates with SHIWA project (disseminates the achievements of SHIWA project) in trying to build a workflow user community across Europe by providing different tools, services and trainings to develop, share and run workflows with the SHIWA Simulation Platform. Both projects support the execution of workflows among a growing number of Distributed Computing Infrastructures (DCIs).

Other initiatives for combining the usage of different combinations of distributed computing infrastructures are described in (Kim et al., 2009 [123], Kloh et al., 2010 [124], Calatrava et al., 2011 [35], Karoczkai et al., 2015 [119])

Kim et al., 2009 [123] focus on the integration of High Performance Grids with Cloud environments while determining different usage models for applications (acceleration, conservation and resilience). They use the Comet Cloud [46] software (and a pull based scheduling approach) to automatically execute applications on dynamically federated hybrid infrastructures based on Grids and Clouds.

Kloh et al., 2010 [124] analyze the usage of hybrid infrastructures for the execution of workflows. In their approach, the scheduling criteria are chosen through a Service Level Agreement (SLA) specified by the user (runtime, execution cost, reliability and network bandwidth).

Calatrava et al., 2011 [35] describe different approaches to integrate the usage of Grid and Cloud-based resources for the execution of High Throughput Computing scientific applications. The authors present the advantages of using a hybrid infrastructure in reducing the execution time of HTC applications when compared to single infrastructures.

The meta-brokering approach defined in (Karoczkai et al., 2015 [119]) to map the parameter study jobs of a workflow to different DCIs is to distribute the load of the workflow evenly among the connected DCIs, based on the number of available resources. This implies a load balancing that puts more job instances to DCIs having a higher throughput (having more resources to execute a certain number of jobs). This approach does not take into account the current background load of the managed DCIs (the running and the waiting jobs at that moment). They apply a weighted job instance distribution among the available infrastructures and rely on resource priority services to determine the high uncertainty and unpredictable load of each available infrastructure while assigning it a certain weight. Their solution for parameter study workflows obtained a more efficient distribution of job instances among the available computing resources with a shorter makespan.

Improving scheduling algorithm’s efficiency improves application performances. Most of the current schedulers base their scheduling decisions on the computing power (and utilization) of the available resources but other aspects of improving the efficiency of scheduling (make the resource selection and schedule generation more efficient) can be considered such as network

related information, idea presented in (Yamini et al., 2011 [247]). Paying attention only to the load of the computing resources can lead to the selection of powerful unloaded computing resources with an overloaded network, which will decrease the performance, especially when the jobs require a high network I/O. Yamini et al., 2011 [247] propose a modified resource selection strategy by making the scheduler consider the available bandwidth before generating a schedule and submitting a job to the resource. Available bandwidth between the node submitting the job and the node to which the job is submitted is checked and if this link is already constrained because of other possible computations in that node, the next node with less constrained links is chosen by the scheduler. Their proposal is applied on a Grid environment, extended with Cloud resources on demand in case of peak demands or heavy requirements.

Other Grid scheduling strategies extended to use on demand Cloud resources in case of peak demands or heavy requirements are presented literature by (Blanco et al., 2009 [28], Ostermann et al., 2009 [170], Rings and Grabowski, 2012 [192]).

The CometCloud (CometCloud [46], Diaz-Montes et al., 2015 [55]) project at the Rutgers Discovery Informatics Institute (RDI2) is an autonomic framework, which enables real-world applications on dynamically federated, hybrid infrastructure integrating (public & private) Clouds, data-centers and Grids. It offers programming support for Cloud bridging (on the fly integration of Grids, commercial and community Clouds and local computational environments), Cloudbursts (dynamic scale-out for dynamic workloads, spikes in demands or other unusual requirements) and various programming paradigms and application requirements. CometCloud can create a nimble and programmable environment, which can automatically evolve over time, adapting to changes in the application requirements and infrastructure (Diaz-Montes et al., 2015 [55]). The 3 layer architecture of CometCloud framework is illustrated in Figure 73 and includes 3 key layers: infrastructure/federation, autonomic management, and programming/interface.

The infrastructure/federation layer manages the dynamic resource federation and provides important services such as (Diaz-Montes et al., 2015 [55]):

- Information lookup (used for resource discovery);
- CometSpace (a shared, decentralized shared coordination space which provides a single management space and multiple shared execution spaces).

The autonomic management layer allows users and applications to define objectives and policies for resource provisioning and application execution while satisfying different user

defined constraints and application requirements. This layer also monitors the execution progress and prevents agreement violations.

The programming/interface layer provides functionalities to describe application workflows (XML documents) and resources (availability, policies and constrains).

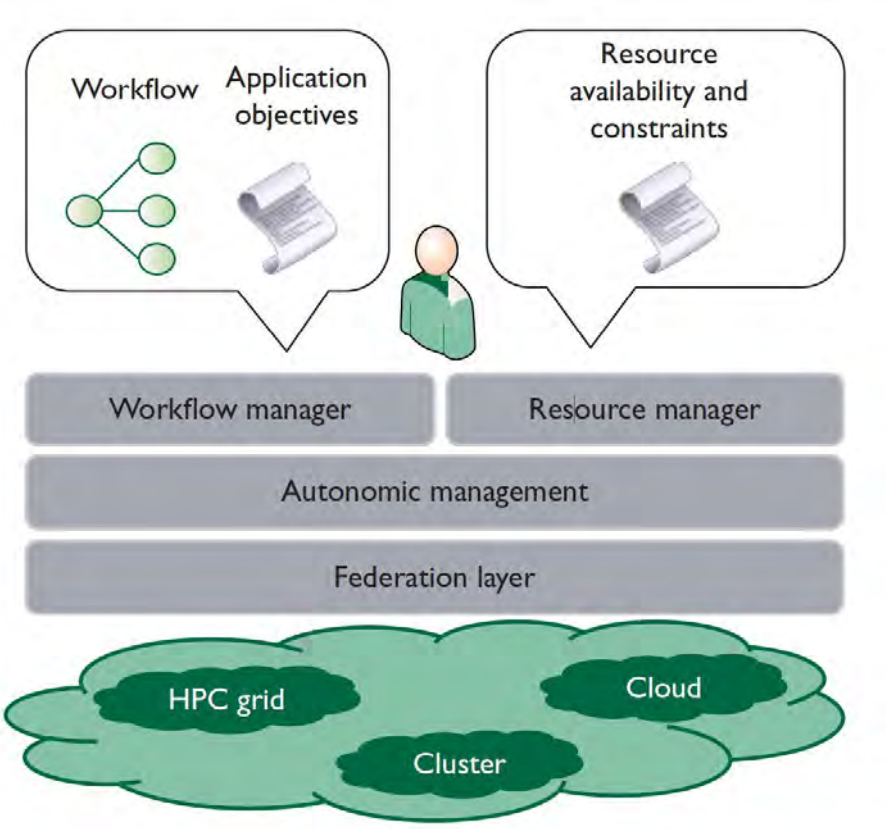

Figure 73: CometCloud Architecture (Diaz-Montes et al., 2015)

9.3. Distributed Systems Interoperability

Interoperability is defined as the ability of making systems and organizations work together (inter-operate). People and organizations working in an interoperable environment should be able to exchange knowledge (or information) as well as to use the extracted knowledge (information) to generate new one on top of them (Reynoso et al., 2014 [188]).

Interoperability between two or more systems foster innovation, allows the creation of new and innovative systems through composition of interoperable systems, increases system availability and reliability and provides a great mean to success (Rings et al., 2011 [191]).

The interoperability and interoperation of different distributed systems have to be done both at the technical level and at the logical level. Different ways of achieving interoperability between two systems exists: standards, adaptors, mediators, gateways.

The long-term approach to achieve interoperability is the implementation of agreed specifications (standards), which capture both requirements and functionality and usually define interfaces and specify protocols to be used for communication via these interfaces. Ideally, these specifications are independent from implementations and are flexible enough for later improvements (Rings et al., 2011 [191]). The problem with standards is that usually it takes a long time for them to be accepted and adopted within a community and it requires long testing periods to validate that an implementation follows the standards, which are rarely unambiguous, and even longer tests are needed to verify if two implementations are interoperable. However, standardization can enable interoperability in a multi-vendor, multi-network and multi-service environment.

In literature we can find several interoperability standard initiatives for Grid and Cloud computing. The Open Grid Forum (OGF) Open Cloud Computing Group (OCCI) Working Group (OGF [166]), the IEEE Standards Association (IEEE Standards Association [109]), the Distributed Management Task Force (DMTF) Cloud Management Standards (DMTF [57]) and the Open Cloud Consortium (OCC) are the major initiative for Cloud systems. OGF Grid Interoperability Now (GIN) is an interoperability initiative in Grid computing described by Rings et al., 2009 [190].

Interoperable distributed systems should allow applications to use them simultaneously. For this, commonly agreed protocols are required for information exchange and overall management. This is still hard to achieve and it will take a long time till all the distributed systems will adopt and implement these agreed protocols. Our proposed solution is the introduction of a broker component (Mediator), which will allow users to access and use the functionalities of different distributed systems in a transparent manner.

9.4. Challenges

There are a large number of applications with interesting workload characteristics and resource requirements, which can take advantage of a hybrid computing environment to reduce execution time, reduce cost (currency or resource allocation) or handle unexpected runtime

situations (unexpected delays or unexpected failures) (Kim et al., 2009 [123]). But developing and executing application in such a complex and dynamic computing environment presents new and significant challenges. Among these challenges we have identified the followings:

- Coordination and management of available computing infrastructures;
- Application decomposition into workflow, jobs and tasks;
- Determining and provisioning the necessary mix of hybrid resources for executing the applications workflows;
- Dynamically scheduling the application tasks across the hybrid execution environment;
- Satisfying multiple and possibly changing application objectives: performance, cost (budget), error recovery (resilience), etc.

9.5. Scheduling

One of the main problems in all distributed infrastructures remains the application scheduling. Scheduling is defined as the assignment of resources to consumers in time, according with a task policy and ordering communication between tasks, and it's considered most of the time a computationally hard problem (NP-Complete) (Pinedo, 2008 [177], Pop, 2008 [178], Bochenina, 2014 [29], Amalarethinam and Josphin, 2015 [8], Lopes and Menasce, 2015 [139]). The complexity of the scheduling process within a HCE if given by several factors such as: resource heterogeneity, size and number of tasks, variety of policies, high number of constrains, platforms interoperability.

A scheduler is supposed to use some best resource selection strategy in terms of user requirements while also adapting with dynamic resources. A scheduler receives a scheduling problem and returns a schedule for it while interacting with different services such as a local resource manager, information services, forecasting, submission, security, and execution services (Yamini et al., 2011 [247]). The scheduling process is described by a scheduling algorithm, which is defined as a procedure used by a scheduler to determine when a task can be run on which resource (Pop, 2008 [178]).

9.5.1. Conceptual Components of Scheduling

In literature there is a large numbers of papers defining scheduling problems and scheduling solutions in different ways but the definition of a scheduling problem and of the

corresponding solutions involves clearly defining important properties such as (Lopes and Menasce, 2015 [139]):

- Definition of **workload** and its components in a complete fashion. A workload defines the consumers of the resources and it's normally composed of **jobs**, defined as a collection of computational **tasks**. There are many properties that can be considered when defining a job considering the number of tasks it contains as well as the dependency relations and the communication needs among the tasks, the required resources, the quality of service, etc. A task could be defined, for a user, as anything that needs a resource, from a bandwidth request, to an application. Tasks can be homogeneous (requiring similar resource demands) or heterogeneous (having different resource needs). They can also be independent or dependent. Dependent tasks are the tasks that have precedence constrains and communications needs to be satisfied while the independency appears when there are neither precedence relations among the tasks not communication needs. Regarding the quantity of resources a job needs, they can be rigid (require a fix number of resources), moldable (the scheduler decides on the quantity of resources), malleable (moldable jobs whose computing requirements can change during execution) and evolving jobs (the user decides on the fly the quantity of resources). Jobs may also be associated or not to a Service Level Agreement (SLA) and penalties can be imposed when these SLAs are violated. Jobs not associated with SLA are considered best-effort jobs. Real-time jobs can also have hard deadlines or soft deadlines.
- The **resources** required to execute a workload consist of a set of ***distributed nodes*** or computers, with one or more processing cores, ***connected*** usually through a high-speed network. A resource could be anything that can be scheduled, a machine, processor, disk space and memory, a QoS network (Pop, 2008 [178]). Resources consist of whole computing units, with main memory, storage devices and network access. They can be organized in a local environment as computer clusters or in widely distributed and scalable centers. The resource heterogeneity in a hybrid-computing environment implies nodes with different computing power, in terms of processing, storage and communication speeds. In such an environment, we are also talking about dynamic scalable infrastructures with dynamic online capacity, which has to be shared not only among the tasks within a specific job but also among different jobs and different users.

- **Scheduling requirements** such as ***goals*** and requirements that must be targeted by the proposed solution. The goal in a scheduling problem is usually to ***optimize*** one or more ***performance metrics*** (multi-criteria optimization) affected by scheduling decisions. One of the most popular scheduling goals is the minimization of the makespan but other goals can be considered as well (maximize resource utilization, maximize throughput, meet deadlines, minimize energy costs, etc.). The makespan is defined as the time difference between the start and the finish of a sequence of jobs or tasks. The granularity of scheduling is also important and it's considered as a scheduling requirement. Usually this granularity is set to level 2: a workload is divided into jobs, which are further on divided into tasks. The job level scheduler decides which job should be executed next, what jobs are position in a waiting queue and in which order, what jobs are resubmitted due to failures or other errors, etc. Task level scheduler decides which task of a given job will run on which resource while the combination of the two levels implies a coordination of both schedulers. A third level can also be considered, in which the tasks consists of one or more processes that must be scheduled at the computing node assigned to execute the task but this is typically managed by the operating system.

As described above, scheduling problems usually also involve other aspects such as resource scaling, resource sharing, data locality, failure model and quality of service required by the workload. Lopes and Menasce, 2015 [139] give a detailed description of all these properties.

As both the workload and the resources may vary over time, scheduling is a dynamic activity and we have to consider its state both with its dynamic and static properties at time instances of interest. We also have to consider that for any scheduling problem there may be one or more scheduling solutions.

The main three steps involved in a scheduling process are (Pop, 2008 [178]):

- Resource discovery – generate a list of potential resources;
- Information gathering about these resources and selection of a best set of resources based on user requirements;
- Job execution (system preparation and submission), including job staging and system cleanup.

9.5.2. Scheduler Architecture

Our proposed Scheduler Architecture is illustrated in Figure 74. Users or Application submit scheduling requests to the Scheduler, which will assign the pool of tasks to appropriate resources based on the computed schedule. This schedule is developed based on the scheduling requests, resource monitoring data (provided by the Resource Monitoring Service), execution history, and configuration parameters.

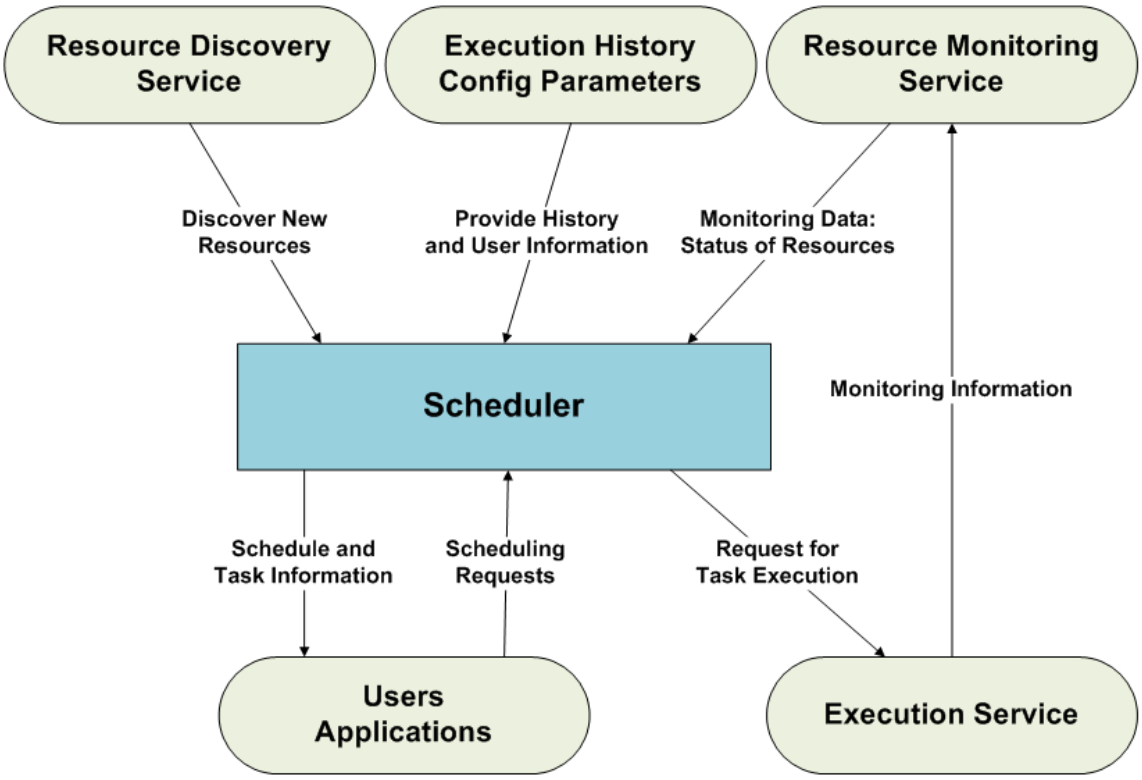

Figure 74: Scheduler Architecture

The Resource Monitoring Service gathers real-time information of the available shared resources in a heterogeneous and dynamic computing environment such as HCE and plays an important role in the scheduling process. The information provided by this service is used to generate automatic decisions for optimal assignation of tasks to resources. In a HCE, this service has to interact to the available Monitoring Services corresponding to each independent computing infrastructure, part of the HCE.

The schedule is sent for execution to the Execution Service, which also updates the Resource Monitoring Service with resource allocation changes.

The users/application receives feedback related with the chosen scheduling solution and the status of the tasks execution from the Scheduler. The Discovery Service is used to easily integrate new resources into the scheduling process.

9.5.3. Scheduling Properties

To achieve a general purpose scheduling approach, several properties have to be taken into consideration (Pop, 2008 [178]):

- *Efficiency* – both regarding the improvement of the performance of the scheduled jobs and the scheduler introduced overhead, which should be reasonably low to not counterattack the benefits;
- *Fairness* – sharing resources among users so that each user obtains his/her fair share;
- *Dynamics* – the allocation strategy should adapt to load changes and exploit the full extent of the available resources;
- *Transparency* – the behavior and results of a task execution should not be affected by the hosts on which it executes i.e. there should be no difference between local and remote execution. The ideal case is when the user is not even aware of the remote processing (except for the increase of performance), meaning that he is not involved in the selection of the resource, application changes, jobs submission, etc.

Considering the scheduling strategies, there are two well-known models (Lopes and Menasce, 2015 [139], Karoczkai et al., 2015 [119]): the push and the pull models. The pull models start a pilot job in a DCI and pull jobs to it, feeding the resource with jobs and avoiding relying on out-dated information on DCI load. The push approach is more traditional and submits all job instances simultaneously to the scheduling component of the workflow management system. This approach may cause significant overheads and bottleneck problems.

Based on the scheduler architecture, we can also split schedulers in three classes (Pop, 2008 [178]): centralized, hierarchical and decentralized. In a centralized scheduler, all the tasks are sent to a single place (entity) in the system, called server or master. The tasks are initially place in a queue, waiting for scheduling and resource allocation. The main problem of this scheduling model is the poor scalability with increasing number of resources while the best advantage is the efficiency of scheduling due to the big picture of the available resources and

pending applications (Pop, 2008 [178]). A hierarchical scheduler is organized on different levels, with the higher-level component managing directly larger sets of resources and indirectly a smaller set of resources, using lower-level components (which can be local schedulers). This model addresses the scalability and the problem of single-point-of-failure but it has to handle site autonomy. A decentralized scheduler has multiple components that work independently and collaborate to obtain a good schedule (Pop, 2008 [178]). The schedule requests in such a model could be processed by a local scheduler or transferred to other local schedulers where other scheduling policies could be applied. The efficiency in this model is lower due to the lack of a big picture regarding the resources and the running applications but it delivers a better fault tolerance and reliability.

Task scheduling processes can be categorized as static and dynamic. In static scheduling, the execution time of tasks and data dependencies between the tasks is known in advance and the scheduling is usually done during compile time. This type of scheduling is also called offline deterministic scheduling (Amalarethinam and Josphin, 2015 [8]). In dynamic scheduling, tasks are allocated to processors upon their arrival and the scheduling decisions are made at runtime. In this case, the tasks can be reallocated to other processors during the runtime. The dynamic scheduling is flexible and faster than the static one (Amalarethinam and Josphin, 2015 [8]). Dynamic scheduling is usually applies when it is difficult to estimate the cost of the applications or when jobs are coming online dynamically. This type of scheduling has two major components (Pop, 2008 [178]):

- System state estimation – collecting state information and constructing an estimate, based on which the tasks will be further assigned to selected resources using load balancing. Due to the NP-Complete nature of scheduling algorithms and the complexity of a HCE, reasonable assumptions are hard to make for obtaining optimal solutions.
- Decision-making – selection of an appropriate set of resources based on system state estimation and other factors.

9.5.4. Scheduling Algorithms

The most basic scheduler takes a job out of a set (pool) of available jobs and sends it to the CPU unit with the minimum amount of load. A job can be executed on a CPU if the sum of the memory needed by the job plus the current load of the CPU does not exceed the total amount

of memory of the selected CPU (Pop, 2008 [178]). If there is no CPU available, the job is added to a waiting queue, based on priorities, waiting for an available CPU to be found. When a job is scheduled on a CPU, it estimates the time needed for completion, based on the amount of power given by the CPU and remains in this state until a change appears (beginning or end of a job on the same CPU for example) or until the execution finishes.

9.5.4.1. Scheduling Algorithms for Independent Tasks

- ***Shortest Job First (SJF)*** – also known as *Shortest Job Next (SJN)* – non-preemptive algorithm that selects the waiting job with the smallest execution time to execute next. Although it requires accurate estimations of the runtime of all jobs that are waiting to be executed, this algorithm is simple and maximizes the job throughput (number of jobs run to completion in a given amount of time). The main disadvantage is the possible job starvations for jobs with a long execution time if short jobs arrive continuously.
- ***Earliest Deadline First (EDF)* –** dynamic scheduling algorithm, mostly in real-time systems, which uses a priority queue based on job deadlines. When an event occurs (a job is finished, a new job is released), the job with the closest deadline will be scheduled next. This algorithm guarantees that the jobs deadlines are met as long as the total CPU utilization is not more than 100% (Pop, 2008 [178]) but when the system is overloaded, the rate of missed deadlines cannot be predicted but the worst-case response times can be calculated.

9.5.4.2. Scheduling Algorithms for DAGs

Scheduling DAG tasks in distributed systems is a known NP-complete problem, which requires different mapping heuristics, based on tasks requirements, structure, and DAG complexity, and task assigned priorities as well as tasks dependencies (Pop, 2008 [178], Alexandrescu, 2012 [7]). The mapping heuristics, which are experienced-based methods, are used for solving problems when an exhaustive search is not possible. They use estimations of the time it takes for a task to run on a specific resource/machine (task execution time).

Some of the most used scheduling algorithms presented in literature are (Pop, 2008 [178], Alexandrescu, 2012 [7]):

- ***Highest Level First with Estimated Times (HLFET)*** – schedules a task to a processor that allows the earliest start time. In this algorithm, the static level (SL) of a node is calculated but the communication costs of the edges are ignored.
- ***Earliest Time First (ETF)*** – computes at each step the earliest start times of all ready nodes and selects the one with the smallest start time (compared to all processors). When two nodes have the same earliest start time, the one with the higher SL is chosen.
- ***Cluster ready Children First (CCF)*** – dynamic algorithm in which the graph is visited in topological order and the tasks are submitted as soon as the scheduling decisions are taken (Forti, 2006 [71]). The algorithm has two job queues: the RUNNING-QUEUE and the CHILDREN-QUEUE. When a task is submitted for execution, it is added in the RUNNING-QUEUE and when it is extracted from this queue, all its successors are inserted into the CHILDREN-QUEUE.
- ***Hybrid Remapper*** – dynamic list scheduling algorithm having as starting point an initial DAG labeled with the execution and the data transfer times. The first phase of the algorithm consists in partitioning the set of tasks into blocks so that the tasks in a block do not have any data dependencies among them. At runtime, the second phase is executed and implies remapping the tasks considering the changes made to the initial statically plan.
- ***Modified Critical Path (MCP)*** – uses the As Late As Possible (ALAP) time of a node as a scheduling priority. This ALAP is determined by first computing the length of the critical path and then subtracting the blevel of the node from it. The algorithm constructs a list of nodes in ascending order of ALAP times and then schedules them such that a node is scheduled to a processor that allows the earliest start time using the insertion approach (Pop, 2008 [178]).

9.5.4.3. Scheduling Heuristics

Some of the most common heuristics used to optimized scheduling processes in distributed systems, described in literature (Pop, 2008 [178], Alexandrescu, 2012 [7]) can be divided in two categories: heuristics that consider only one task at a time in the mapping process (such as: OLB, MET, MCT, SwA, KPB, etc.) – these heuristics usually have a low algorithm complexity and a short mapping time but they provide poorer results, and heuristics that consider

all the tasks (such as: MnMn, MxMn, Duplex, GA, SA, etc.) – provide better solutions but they usually take longer to run.

- ***Opportunistic Load Balancing (OLB)*** – one task is picked arbitrary from the group of tasks and assigned to the next machine expected to be available. This heuristic does not take into consideration the expected execution time on that machine (implying a possible poor makespan) but it's simple and offers high process utilization.
- ***Minimum Execution Time (MET)*** – each task is picked arbitrary and assigned to the machine with the least expected execution time for that task, regardless of the time the machine becomes available (implying a possible severe load imbalance).
- ***Minimum Completion Time (MCT)*** – each task is picked arbitrary and assigned to the machine with the minimum expected completion time for that task, avoiding the cases in which OLB and MET performs poorly.
- ***Switching Algorithm (SwA)*** – a hybrid of the MET and MCT heuristics, designed for use in dynamic environments but it can also be used in static ones. In this heuristic a tasks list is generated that includes all unmapped tasks in a given arbitrary order. The first task in the list is assigned using the MCT heuristic. Following, the heuristic used to map the task is determine based on the load balanced index calculated for the system: if the load balanced index is higher than a threshold, MET is selected, if the load balanced index is lower than the threshold, MCT is selected, otherwise the current heuristic remains selected. These steps are repeated until all the tasks are mapped.
- ***K-Percent Best (KPB)*** – in this heuristic, the "K-percent" of the machines with the smallest execution time for a given task are identified and the task is mapped to the machine in this subset with the minimum completion time. A value of K equal with 1/M% makes this heuristic to be the same as MET while a 100% value makes it the same as MCT.
- ***Min-Min (MnMn)*** – Out of the set T of all unmapped tasks, a set C of minimum possible completion times of all tasks on any of the machines is computed. The task with the minimum possible execution time is assigned on the corresponding machine and the process continues in the same way with the rest of the unmapped tasks.
- ***Max-Min (MxMn)*** – similar with Min-Min heuristic, but when the task form C with the overall maximum completion time is assigned to the corresponding machine, it is

removed from C and the process repeats until the set C is empty. This heuristic tries to perform tasks with longer execution times first, leading to a better balanced allocation of tasks.

- ***Duplex*** – combination of Min-Min and Max-Mix by performing both and choosing the better solution.
- ***Genetic Algorithms (GA)*** – used for searching large solution spaces. It computes multiple possible mappings, considered as chromosomes in the population. Each such chromosome has a fitness value, computed using the objective function of the scheduling algorithm (based on the desired performance criteria needed to be improved). In each iteration, all the chromosomes are evaluated based on their fitness value and only the best of them will survive in the next population, which contains also new allocations based on crossover and mutation. The stop condition can be either a certain number of iterations or a convergence of the chromosomes to the same mapping. Genetic algorithms usually take much longer to run and are used in situations where there is no strict time limitation in which the heuristic has to finish the task mapping (Alexandrescu, 2012 [7]).
- ***Simulating Annealing (SA)*** – uses the same representation of chromosomes as for the GA but considers only one possible solution (mapping) at a time. This heuristic allows poorer solutions to be accepted in the attempt of obtaining a better search of the solution space.
- ***A**** - search technique based on a μ-ary. It starts at the root node that is a null solution and as the tree grows, nodes (mapped tasks) represent partial mappings. Each parent node generates μ children, where μ is the number of possible mappings for task Ti. After this, the parent node becomes inactive. A pruning process is performed to limit the maximum number of active nodes in the tree at any time. Each node also has an associated cost function (of partial solution) plus a lower bound estimate of the time to execute the rest of the unmapped tasks. After the rood node generates μ nodes for the first task, T0 (representing the mapping of the task on the available machines), the node with the minimum cost function generates its μ children and so on until a redefined number of nodes are created. Any time a node is added, the node with the largest cost function is deactivated, pruning the tree, and so on until a complete mapping is found (Pop, 2008 [178]).

9.5.4.4. Near-optimal Scheduling Algorithms

The most representative near-optimal scheduling algorithms presented in literature (Pop, 2008 [178]) are:

- ***Random and Best of n Random*** – a random scheduling algorithm, which chooses a resource at random for each task in the application (just by checking if that task can be executed on the assigned resource). The optimization of this algorithm (Best of n Random) randomly generates n schedules and choses the one with the highest benefit value.
- ***Exhaustive Search*** – used both to discover the optimal solution to the scheduling problem and to define an upper bound on the computation time taken to select a schedule. It performs a depth first search on the tree of all possible scheduler, calculating the benefit of each scheduler and returning the optimal solution.
- ***Simulated Annealing*** – generalization of the Monte Carlo method, used for optimization of multi-variable problems. The algorithm generates possible solutions randomly and compares their benefit with a currently selected solution, discarding or accepting the new ones. The process is repeated and in each iteration either a maximum number of new solutions are accepted or a maximum total number of solutions are considered. Once an iteration is completed with no new solutions accepted, the current solution is returned as the best one.
- ***Game Theory*** – technique in which a number of players attempt to optimize their own payoff by selecting one of many strategies.
- ***Genetic Algorithms.***

9.5.5. Scheduling in a Hybrid Computing Environment (HCE)

The scheduling problem for jobs with precedence constrains has been shown to be NP-hard (Pinedo, 2008 [177], Bochenina, 2014 [29], Amalarethinam and Josphin, 2015 [8]), which propagates therefore in a hybrid computing environment (HCE) as well and requires solutions based on approximations and heuristic techniques. Methods based on runtime estimates, which can be inaccurate, are a well-known problems mentioned in the job scheduling literature (Karoczkai et al., 2015 [119]). HCE is a heterogeneous environment composed of different

distributed systems, which provide or not different type of public information regarding the current load of the infrastructure. An estimation of the current available capacity at a certain moment of time (runtime) is therefore difficult to make.

Within a HCE, an adaptive scheduling has to be used, which can change the scheduling decisions dynamically according to the previous, current and/or future resource status (Pop, 2008 [178]). Such an adaptive scheduling has to consider criteria such as: the heterogeneity of candidate resources, the dynamism of resource performance, and the diversity of applications.

When scheduling an application on a HCE, which involves a high degree of resource heterogeneity, the problem of selecting the most convenient infrastructure and the computational resources that are able to support the execution jobs, is a complex problem. We have to consider a series of common parameters from all distributed computing infrastructures:

- **Estimated execution time** of a job – either through user specification, use of a batch job or historical background of average execution time;
- **Data transfer time** – the time required for data staging from user to the computational resource and back. This is useful to be able to compute the overhead of a remote execution. This time would be different based on the chosen computational infrastructure. On Grid infrastructures, data is usually copied on a Storage Element (SE) from where every worker node can copy it locally for execution, using GridFTP protocol. The selection of SE is also important and it's usually done depending on the geographical distance (as close as possible to the Computing Elements (CEs)). Cluster architectures usually have a common file system so the transfer of data is really fast considering that the resources will see the data as local. Within a Cloud infrastructure, things tend to become more complex as there can be different solutions. We have experimented three of them and they are all presented in Chapter 7: . The most common solution is to copy the data to the closest Data Storage in the Cloud. Each VM can be configured to connect to the Data Storage and retrieve the data when needed. The results are copied back in the Data storage after execution, from where the user can easily retrieve them. Data intensive applications may have an overhead introduced by data transfer too big to take advantage of a remote execution. An estimation of this value can be done using the size of the input/output data, the bandwidth between the user and the storing place (which can be

dynamically recomputed after each transfer) and the type of chosen computing infrastructure.

- **Jobs dependences:**
  - **software** dependences – libraries, software packages, operating systems;
  - **hardware** dependences – CPU architecture, available RAM;
- **Job type –** influences the priorities assigned to each computing infrastructures. A HPC parallel job would be more appropriate to a cluster or a Grid infrastructure while independent serial jobs can equally run on the Cloud. Computational costs for each infrastructure still need to be taken into consideration.
- **User requirements/preferences/constrains –** the selection of the appropriate resources has to be done while considering also the user preferences, requirement and constrains. One of the important constrains could be the user authorization to one or more computing infrastructures. To be able to launch into execution jobs, each of the composing infrastructures requires a certain level of authentication and authorization. Clusters and Clouds are usually based on user-password credentials while Grid infrastructures require a user certificate emitted by a Certificate Authority (CA). Other common constrains or requirements could be the total execution time of the application, the financial cost of execution (considering the execution on public Clouds), application security, etc.

A series of particular parameter have to be considered based on specific computing infrastructure and we describe our vision on each of them in the following.

Within a **Cloud** infrastructure, we have to consider the specific parameters base on Cloud provider as well, but the most common ones are presented also in literature (Calatrava et al., 2011 [35]):

- **Access time to the Virtual Machine Image Repository and Catalog** (VMRC) – this involves the time to interact to VMRC, search for an image and transfer the image from the repository to the deployment site (dependent on the VM size and on the bandwidth) if the image was found.
- **Deployment time of the VMs** – time required to deploy a copied VMI on a physical infrastructure or, if such a VMI was not found, the time needed to deploy an empty VM and customize it with the necessary software and libraries. In some cases, this time is

increased also by the time required to boot up the physical node in which the VM is deployed.

- **User budget** – maximum budget allocated by the user in case a public Cloud on a pay per use basis has to be used to provision resources. This budget will restrict not only the number of deployed VM but also their active time.
- **Cloud security and QoS** – Cloud provider adherence to Service Level Agreement (SLA) and the obtained QoS based on failure history.

In a **Grid** infrastructure we have to consider the following important parameters:

- **Grid session creation and configuration time** – time needed to specify specific attributes to create a Grid session: user certificates, VO, WMS, WOMS, etc.
- **Proxy Certificate generation and delegation** (authentication and authorization) – time required to generate the proxy certificate from the user certificates. A job execution on the Grid infrastructure requires a valid proxy certificate. If a MyProxy server already exists, only a copy of the proxy certificate will be used by accessing the server using a server URL, username, password, credentials name and validity of the proxy certificate. Users are typically restricted to resources from one or several VOs.
- **Job description time** – using a dedicated description language (JDL for gLite).
- **Access time to get the list of compatible CEs** and the corresponding information about the computing resources: total number of processors, total number of available processors, total number of jobs, total number of executing jobs and waiting jobs, etc. All this information is needed to select the Grid resources satisfying the job requirements.
- **Access time to get the list of SEs** – to determine the closest SE to store the input and the output data and to minimize the data transfer time

Specific parameters that have to be considered in a **cluster** infrastructure:

- **Job description** (in SLURM for example).
- **Access the system information** – to estimate a job waiting time.
- **Queue waiting time of the job** – based on the user priority and the load of the cluster.
- **Maximum running time in the cluster –** Baobab cluster for example has a maximum running time for a job of 4 day; after this range, the jobs are cancelled even if they are

still running. Acquisition of personal nodes in the Baobab cluster allows a maximum execution time of 7 days on the specific nodes.

Workload and Resource Management System (WRMS)

A workload and resource management system is defined as a service that is provided by a distributed network computing system that manages a pool of named resources to optimize a performance metric for a given workload (Figure 75). The main characteristics of a WRMS are:

- Manages resources, including resource management conditions.
- Manages tasks, including creating, assigning the ranks, waiting and monitoring.
- Schedules and maps tasks to a set of resources and allocates resources for certain tasks at a time.

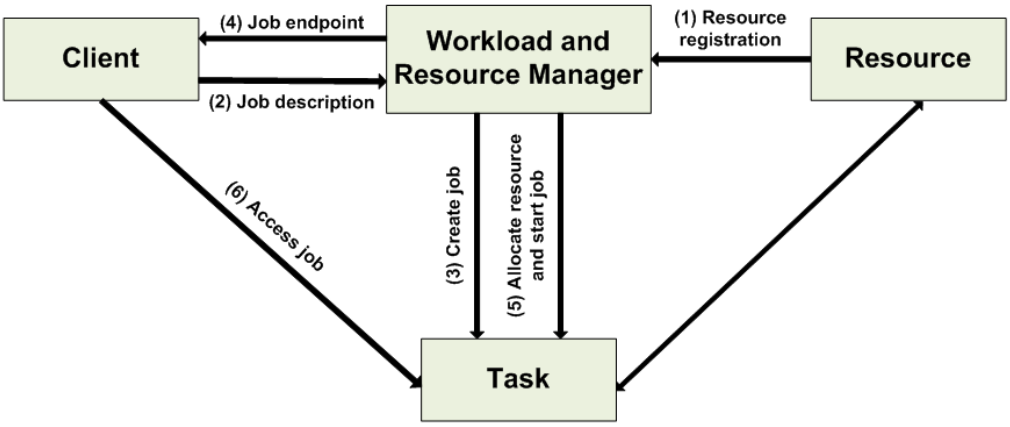

Figure 75: Workload and Resources Management System (WRMS)

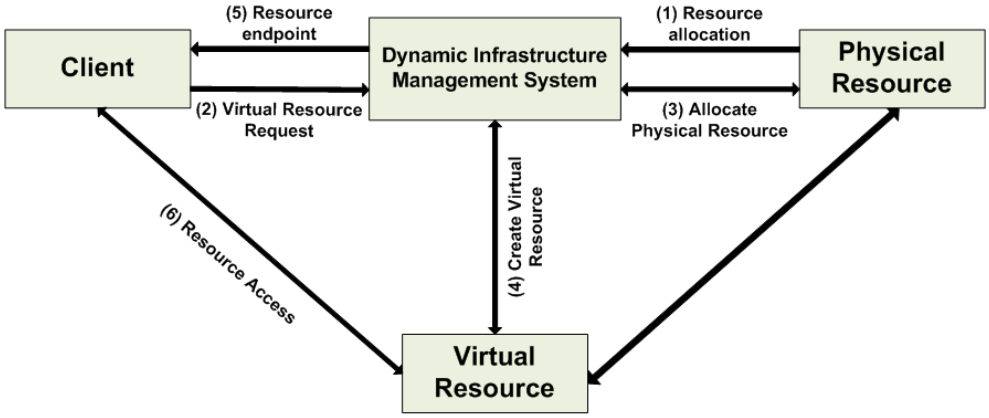

Figure 76: Dynamic Infrastructure Management System (DIMS)

Scheduling a workflow on one or more distributed infrastructures must take into account specific issues and some of them are also related to the application type and the associated workflow. The most common issues we have identified are: intermediary data transfer, identification of reliability level of computational resources, optimization criteria (minimize the response time, maximize the resource utilization, makespan reduction, cost minimization, best resource selection, time constrains, reliability, network bandwidth, etc.) and variation limits for multi-criteria algorithms, tasks grouping to reduce bandwidth consumption, history utilization for resource selection, QoS requirements.

9.6. Execution

The efficiency of parallel programs depends not only on how efficiently the program was parallelized but also on the execution environment (workload, platform, performance goal). The variability of any of these dimensions implies variability in the execution environment and requires the adaption of the parallelism in the application. In the absence of an efficient and intelligent tuning and packaging, a parallel program can perform even worse than the original sequential program (Suleman et al., 2010 [218]).

The parallelization process is an important topic as for the integration of the scientific applications within the parallel and distributed infrastructures and may occur at data level or/and at processing level. There are two concepts that have to be discussed when talking about task parallelization. First, a task can be divided into several independent sub-tasks operating on the same or different dataset, which we call task parallelism. This parallelism emphasizes the processing distributed nature. Secondly, a task can be divided into several subtasks, each processing a part of the whole dataset, which we call data parallelism, also known as partitioning or tiling in the geospatial community. As a whole, in this type of parallelism, the individual sub-tasks require no communication between each other; nevertheless the process of combining/merging the individual results often incorporates specific logic due to the partition borders dependencies (Rodila and Gorgan, 2011 [197]).

The execution of jobs on the Grid was already performed and presented in Chapter 7: . In our case this was done using CERN developed tools, GANGA and DIANE, for scheduling and launching into execution jobs on gLite Middleware based Grid infrastructures. Considering the

Grid platform, porting application on this platform raises different questions (Werder and Kruger, 2009 [239]):

- How to divide a given task into several subtasks, which then can be executed concurrently? In most cases, the solution to this problem depends only on the algorithms and the processes of the task itself, and is therefore not specific or limited to Grid-computing;
- How to make use of the Grid functionalities in order to accelerate the processing? In this case, the available Grid middleware and orchestration of Web services have to be considered.

The migration of existing applications to Cloud requires adapting them to a new computing environment. This migration process can be view from two perspectives (Andrikopoulos et al., 2013 [12]):

- Migrating the whole application on the Cloud, using virtualization, and delegating the adaptation effort to the resource management level;
- Migrating individual application layers or even individual architectural components on the Cloud, allowing a more flexible and a better control over the application migration.

From the migration point of view there are several types of applications (Andrikopoulos et al., 2013 [12]):

- Many applications are not ready to be moved on the Cloud because this computing platform is not mature enough yet for this type of applications such as safety critical applications (Badger et al., 2012 [19]).
- Embedded systems for example are the type of applications, which may not make sense to migrate them at all.
- Cloud-native applications are the applications specifically implemented for the Cloud.
- Cloud-enabled applications are the ones that need to be adapted for the Cloud environment.

What kind of migration can be performed depending on the application type? Adapting the Presentation Layer of an application to the Cloud environment is somehow similar to transforming applications in services (or exposing them as services). Concerns that affect the layers of an application when migrating to Cloud (Andrikopoulos et al., 2013 [12]):

1. What is the impact of the logical and physical distribution of the migrated application?
2. What elasticity mechanism can be used for different types of applications?
3. How does migration affect the multi-tenancy capabilities of applications?
4. How to calculate the cost of migrating the application and operating in the Cloud?
5. What is the impact of the Quality of Service levels of the application, and how is application security affected?

Rapid elasticity – the capability to quickly scale outward and inward depending on demand (essential characteristic of Cloud computing). Elasticity provides means for optimizing the resource usage for the cases of unknown loads (fluctuating). The enabling foundation of elasticity is the application scalability, which can be of two types: horizontal (adding more instances when required – depends on the application and on the application's components) and vertical (adding more computational resources to the application – depends on the service provider). The NIST report on Cloud computing (Badger et al., 2012 [19]) identifies limitations for the benefits of elasticity. Private Cloud scenarios exhibit basically the same limitations in maximum capacity similar to these of traditional data centers. Public Clouds are offering theoretically unlimited resources but at a certain cost. Hybrid architectures (combination of traditional and Cloud-enabled computing capabilities), combined with horizontal scalability are reported (Tak et al., 2011 [222]) to offer the best solution, at least in terms of cost effectiveness (only for certain type of applications however).

The migration of an application to Cloud entails a loss of control over the QoS characteristics due to the reliance on the QoS levels offered by the service provider. As a result, the QoS characteristics (especially availability and reliability) offered by a Cloud service provider appear to have a greater importance to application stakeholders than hosting the application traditionally.

Security is one of the major concerns and an obstacle to migrate to the Cloud. It entails both the communications and data aspects, but also the physical/digital one (the risk of losing or compromising data due to data center failures or other physical attacks).

The most important open issue affecting all application layers is probably the interoperability between Cloud service providers. The difficulties in interoperability between providers are also due to the lack of standardization and the different application models used by services.

The execution of jobs on a Cloud infrastructure involves actions such as: launching virtual machine, installing the necessary software and libraries required by the job execution (contextualization service), sending jobs to execution on the deployed VMs and shutting down VM based on the queue of pending jobs and other execution constrains. We consider the existence of a catalog and repository service of Virtual Machine Images (VMIs) such as the Virtual Machine Image Repository & Catalog (VMRC) (Carrion et al., 2010 [36], Calatrava et al., 2011 [35]) that enables searching for specific VMIs based on hardware and software requirements of jobs. When trying to execute a job in the Cloud, a VMI able to satisfy the job requirements and constrains is searched in this catalog. If such an image is found, it will be retrieved from the repository and deployed in the Cloud. If we cannot find an appropriate VMI, an empty VMI has to be deployed followed by the installation of all the necessaries software and libraries (contextualization service). Once the VMI is contextualized, it is stored in the VMRC and used for future executions.

9.7. Monitoring

The monitoring information is extremely important in dynamic scheduling processes because it offers a full view of the heterogeneous resources and the current execution and it enables high performance computing. Monitoring tools track not only resource usage but also network traffic, job distribution and others, and present this information for further effective decision making processes (Pop, 2008 [178]). The monitoring information can be used in different scenarios such as: scheduling, performance tuning, performance evaluation, resource utilization, fault recovery, debugging, user information, etc.

Specific infrastructures like Grids have monitoring components that provide information about the background load of the infrastructure through components called Information Systems. The monitored data they published is not always accurate (Karoczkai et al., 2015 [119]) and cannot be used for exact scheduling. Beside this, in Grid systems the execution time of running jobs is not known and estimations are hard to make (Lee et al., 2005 [131]). Local clusters on the other hand may provide more accurate load information but when dealing with commercial clouds, these types of information are usually hidden on the underlying infrastructure.

9.8. Fault Tolerance

A fault tolerant system is one that provides a set of services based on a pre-defined contract, in spite of possible system errors. This system has to detect, correct and eliminate errors while it continues to supply an acceptable set of services (Pop, 2008 [178]). The goal of such a system is to improve its characteristics so that the detection and elimination of errors is done as smooth as possible. The complexity and the distributed nature of HCE imply also a large set of possible errors, some of them common to all distributed systems and some of them particular:

- *Network errors* – caused by the communication channels and manifested through package losses or package corruptions;
- *Timing errors* – occur either due to the impossibility of establishing a connection or when the response time of one of the components exceeds a certain expected time;
- *Response errors* – caused by a service, which returns values outside of an expected range. A validation mechanism is necessary in this case;
- *Byzantine errors* – arbitrary errors that can appear during the execution and they refer most of the time to crashes or omissions. In these cases, the system enters in an undefined behavior state;
- *Physical errors* – caused by critical conditions of the physical resources (processor, memory, storage, network, etc.). In these cases, the corresponding physical resources should be removed or replaced;
- *Life cycle errors* – specific to components with services that can expire at a certain moment of time or need to be updated.
- *Interaction errors* – caused by incompatibilities at the communication protocol, security, workflows or timing. These errors are specifically larger in a HCE due to the diversity of distributed systems and the lack of interoperability standards between different computing infrastructures.

There are different approaches to provide fault tolerance in distributed systems. The main one is the rollback technique, which requires the creation of application states at different moments of time and restoring the execution to the last stable state in case of an error. Another technique is replication, which implies the execution in parallel of the application on multiple

resources and in the case one is interrupted, the execution continues on the active resources. Process migration is also a possible solution to provide fault tolerance and can be applied either at the application level or at task level.

9.9. Discussions

When considering a heterogeneous environment of computing resources, the most common problems that can appear are related to the efficient load balancing across the existing machines, scalability, fault tolerance and security. Another important problem in these environments is the efficient mapping of the tasks to appropriate available resources. Creating an intelligent component that uses artificial intelligence to solve most of the above-specified issues can be a good solution that can improve the system's reliability and efficiency. Although the research in this direction is limited, the outcomes are promising (Alexandrescu, 2012 [7]). The most important challenges that can appear in a computing heterogeneous environment are related to task mapping, transparency, communication and connection, fault detection and fault tolerance, synchronization, load balancing, security, resource scalability and resource discovery, etc.

There are different scenarios in which the usage of multiple heterogeneous resources, coming from different computing infrastructures, can be of significant importance.

1. Improve time performance – the usage of additional infrastructure resources is done to improve the application time-to-completion. In this scenario, the new resources are used to reduce the impact of queue wait time or to exploit an additional level of parallelism. When talking about the usage of public Cloud resources, appropriate budget constrain has to be taken into consideration.
2. Error prevention – the usage of additional resources is done to handle unexpected situations such as unanticipated downtime, inadequate allocations or unanticipated queue delays.

9.10. Conclusions

A single parallel architecture started to become inadequate to cover the needs of parallelism of some large-scale complex applications, therefore the increasing interest in scheduling for heterogeneous distributed systems. Optimized scheduling algorithms for multi-

criteria constraints are extremely important to achieve high resource utilization. Heterogeneous systems have proven to offer higher performance at lower cost than a single high performance-computing machine, in many practical cases (Pop, 2008 [178]).

Managing a Hybrid Computing Environment (HCE) formed of different distributed computing infrastructures (DCIs) and offering a transparent and flexible access of a large pool of highly heterogeneous resources are complex problems to solve and involves a large number of challenges that have to be considered and sophisticated approaches to be applied. Among these challenges we mention the followings:

- Managing and administration of large scale scientific applications;
- Conceptual descriptions of application through formalism and abstract concepts;
- Efficient scheduling of the jobs composing the application workflow over a large set of highly heterogeneous resources;
- Resource reservation mechanisms from a high uncertainty and unpredictable load of resources coming from different computing infrastructures;
- Adaptive executions of jobs on different distributed computing infrastructures;
- Error detection and error recovery mechanisms adapted to a HCE.

We have focused our attention on scheduling mechanism within a HCE because we consider this an essential component in obtaining high performance computing. Various strategies for scheduling in distributed systems have been developed and reported in literature and most of them are oriented towards static scheduling (Braun et al., 2001 [33], Pop, 2008 [178]). In this type of scheduling, both the assignment of tasks to processors and the execution start times are determined in advance. The tasks are assigned only once to a resource and the estimation of the execution cost can be made before the actual execution, informing and even given users the possibility to decide among several scheduling options. The main drawback of this type of scheduling is that it cannot be applied in a highly-dynamic environment, which undergoes a lot of changes such as: a node selected to perform a computation fails, or it becomes isolated from the system due to network failures, it is heavily loaded by other users/applications and the response time is not sufficient, etc. All these changes are possible in a HCE and therefore a dynamic scheduling mechanism is more appropriate. Similar, optimal schedulers are difficult to obtain in such a dynamic and complex environment and therefore sub-optimal ones seem a good

compromise. These algorithms use formal computational models and heuristics and instead of searching the entire solution space for an optimal solution, they stop when an acceptable solution is found.

Scheduling a workflow in a HCE must take into account specific issues and some of them are also related to the application type and the associated workflow. The most common issues we have identified so far are: intermediary data transfer, identification of reliability level of computational resources, optimization criteria (minimize the response time, maximize the resource utilization, makespan reduction, cost minimization, best resource selection, time constrains, reliability, network bandwidth, etc.) and variation limits for multi-criteria algorithms, tasks grouping to reduce bandwidth consumption, history utilization for resource selection, QoS requirements.

9.11. Personal Contributions

- Analyze and explore challenges in a Hybrid Computing Environment (HCE), composed from different distributed computing Infrastructures (DCIs): cluster, Grid, Cloud.
- Published Papers:
    - ***Rodila, D.,*** *Gorgan, D., Ray, N., and Lehmann, A. (2016). ENV2CE: Environmental Application Conceptualization and Execution on a Hybrid Computing Environment -Framework and Methodology Proposal, (to be submitted).*

Chapter 10: Environmental Applications Conceptualization and Execution on a Hybrid Computing Environment (ENV2CE)

10.1. Introduction

In this chapter we will introduce a new methodology **ENV2CE** (**Env**ironmental Applications **C**onceptualization and **E**xecution on a Hybrid **C**omputing **E**nvironment) for easily porting and executing environmental applications on a Hybrid Computing Environment (HCE) based on the application conceptual model (described in Chapter 8:) and a new framework architecture. The HCE is composed from several heterogeneous computing resources belonging to local machines, HPC cluster, Grid and Cloud infrastructures. We will describe in detail the HCE, the proposed framework and its components and the underlying methodology to efficiently execute environmental applications on such a system.

10.2. Similar Research Initiatives

Raman (2011) [181] presents a system for flexible parallel execution called Parcae. The author presents a separation of concerns of parallel application development, its optimization and its use to be able to execute parallel applications robustly across a variety of execution environments. Using the Parcae system, the authors guarantee that the specified performance goals of an application are met in a variety of application execution environments. According to this research, the applications developed in the sequential programming model (which includes also the legacy code applications) are enhanced automatically to execute flexibly on Multicore platforms. The Parcae system is composed of 3 important components: Nona compiler, Decima monitor and Morta executor. This system optimizes the execution of multiple flexible programs running on a shared parallel platform and reduces the execution time by -1.39% to 41.94% with concomitant energy savings of 23.9% to 83.9%. The authors have reached to the conclusion that there are several important factors to maximize parallel program performance:

- The ability to expose and optimize parallelism across multiple levels in a loop nest;
- The ability to express multiple types of parallelism simultaneously;

- The ability to expose application features to the runtime system.

Cooperative use of parallel resources, as orchestrated by a platform-wide resource manager in concert with each application's run-time system, is essential to maximize platform utilization. Tighter integration between the application-level run time system and the operating system's scheduler yields significant performance gains compared to when the two operate in isolation. The aim of the Parcae system is to support new parallelization optimizations and adaptive platform-level optimizations if possible suitable for the new executing infrastructures.

Several tools have been developed to extract parallelism from sequential code (Bridges et al., 2007 [34]) or to extract thread-level parallelism (OpenMP - http://www.openmp.org, Leiserson, 2009 [134]). Parallelism extraction is just one part of the problem of synthesizing well performing programs that execute efficiently in a variety of execution environments. The other, equally important, part is the tuning and packaging of the extracted parallelism. In the absence of intelligent tuning and packaging, a parallel program may perform worse than the original sequential program (Weissman, 2002 [238], Suleman et al., 2010 [218], Raman et al., 2011 [182]).

To summarize, Parcae enables the separation of concern of parallelism discovery and extraction from the concern of optimizing and redeployment to adapt the application to a new execution environment.

According to (Raman, 2011 [181]), the developments of parallel applications, which will be executed efficiently in a variable execution environment, have to take into consideration the following aspects:

- Correctly partition the application into parallel sections/tasks;
- Specify appropriate logic to the application to be able to adapt to the changes in the executing environment;
- Design and implement a component able to adapt the application parallelism to the executing environment.

Most of the existing solutions attempting to solve the parallelization of applications in a varying executing environment omit to treat separately these issues, resulting in limited portability, extensibility and flexibility.

Libraries and API such as OpenMP, Threading Building Blocks (TBB) (Reinders, 2007 [187]) and Pthreads (IEEE and the Open Group, 2004 [110]) impose the specification of a single fixed configuration of a parallel application (imperative specification). Such a single program configuration will become suboptimal as the execution environment changes (Suleman et al., 2010 [218]). A better approach would be the case in which the programmer should not worry about a specific parallelism configuration. This should be adopted automatically at the execution as it depends on the chosen execution environment, without manually code the adaptation logic. An ideal system should also enable the expression of multiple parallelism types (data parallelism, task parallelism, pipeline parallelism) and selection of one or more types according to the application's execution environment. In the ideal case, such a system should allow the specification of multiple performance goals without the need to rewrite the application code, assuring the portability of the application across different systems with varying goals and constrains (Raman, 2011 [181]).

Most of the proposed systems (Curtis-Maury et al., 2006 [50], Reinders, 2007 [187], Blagojevic et al., 2007 [27], Wang et al., 2009 [235], Suleman et al., 2010 [218]) able to adapt the execution of parallel programs to a variable executing environment are tied to a specific performance goal, a specific parallelism type or a specific mechanism of adaptation.

A high-level Grid and Cloud framework, which allows a smooth transaction from clusters and Grids to Clouds, is presented in (Amedro et al., 2010 [9]). They collect the information specific to the application infrastructure in a deployment file, separate from the application source code, allowing the application to run on different platforms without any modification. The authors have evaluated the performances obtained when running scientific applications on Cloud and they concluded that there are cases in which other traditional computing platforms perform better. These observations lead to the idea that the best approach in obtaining good results is a mixture between Cloud and other platforms. To facilitate the deployment of application in a hybrid Grid/Cloud environment, they used the ProActive Parallel Suite (http://proactive.activeeon.com/). ProActive middleware provides an abstract descriptor-based deployment model and gives the possibility to deploy an application on different platforms without changing the source code.

Giusti et al. (2010) [88] present the AMTHA (Automatic Mapping Task on Heterogeneous Architectures) algorithm for task-to-processors assignment and the MPAHA (Model of Parallel Algorithms on Heterogeneous Architectures) model.

The lack of common standards in Geospatial area is a great obstacle in the interoperability of this field with different existing execution platforms. Muller et al. (2013) [156] present an approach to share, reuse and even standardize geo-processing logic in the remote sensing area by moving code packages as self-describing software components. These components should contain algorithmic code and machine-readable descriptions of the functionality, platform and infrastructure. The research also presents a mechanism to distribute those packages on the Web and to integrate them in different varying execution environments.

Several other research studies have shown how Cloud providers are able to supply the needs of resources of different data intensive applications executing on different infrastructures.

Palankar et al. (2008) [174] have investigated the possibility of mixing the Grid and the Cloud infrastructures by allowing the Grid users to perform costly data operations on the Grid resources while utilizing the data availability provided by the Clouds. The study is focused on the Amazon's storage utility – Simple Storage Service (S3) – that aims to provide data storage as a highly available, low-cost service, with a pay-as-you-go billing model. The aim of the research is to integrate this service within the Grid infrastructure and to identify the needed requirements for such a storage service in this context. Although Grid applications can benefit from using Amazon services, such as improving data availability, Palankar et al. (2008) [174] highlighted that a balance between the benefits of Amazon services and the cost of using Amazon's infrastructure should be taken into account. "This balance involves performing expensive operations that generate large amounts of temporary data at the Grid infrastructure."

VioCluster (Ruth et al., 2005 [203]) is a virtualization based computational resource sharing platform, which allows to dynamically adapting the capacity of clusters by borrowing idle machines of peer domains. In this study, a broker is responsible for managing a virtual cluster/domain and it has borrowing and lending policies, which allow it to easily borrow and lend resources to another broker. Ruth et al. (2005) [203] conclude that dynamic machine trading between virtual domains decreases their job wait time and increases their resource utilization.

Chang and Karamcheti (2001) [40] introduce an application-independent adaptation framework which simplifies the design of resource-aware applications i.e. applications which

adapt their behavior to changes in the executing resource characteristics while ensuring a desired performance level. The need for adaptation decisions is no longer explicitly programmed in the application. An interface is responsible for exposing the adaptation choices as alternate application configuration while the execution of the application is performed in a virtual execution environment with diverse resource availability. This framework allows automatic run-time decisions for adapting an application and specifying when to adapt (by monitoring the application progress and the resource conditions) and how to adapt (by dynamically choosing an appropriate application configuration).

10.3. Methodology

Running applications across multiple computing resources has been done in previous works and some of them have been presented in section 10.2 but these approaches are not proposing a full methodology based on theoretical concepts and formalisms. They offer instead an API for a submission procedure and give examples on particular use case applications from different areas.

We propose a methodology that is developed based on lessons learned from deploying and porting different large-scale environmental applications on different parallel and distributed computing infrastructures, presented in Chapter 7: . This accumulated expertise from customized solutions helped us build a step-by-step, general, and possibly standard methodology to effectively and efficiently port environmental application on different computational backends. Giving the generality and the flexibility of the proposed solution, it can easily be extended to other scientific applications from other research fields.

The steps of our methodology are:

- **Preprocessing phase** - in which the framework gathers the information about the available computational systems, the executed application, the user preferences, etc. In this step we also define the application conceptual model.
- **Analysis phase** - in which we model both the application structure and the execution flows, based on the gathered information in the previous phase.
- **Decision/Scheduling phase** - in which we schedule the tasks to resources, we make decisions on which computing environments is more suitable, based on the

application developed model, availability of the platforms, user preferences, history, predictions, etc.

- **Execution phase** - in which we actually execute the application on the selected computing backend, based on the defined policies.
- **Monitoring phase** - in which the execution is monitored and information is collected for future executions.

The main objectives of ENV2CE methodology are to:

- Allow an easy and flexible execution of an environmental application;
- Obtain a faster execution time;
- Exploit to the maximum the parallelization within an application;
- Utilize resources more efficiently; and
- Provide the user the possibility to either choose what computational resources to use or to leave the system choose for her.

10.4. Proposed Framework

Most of the large scale scientific applications require large storage and computing facilities but unfortunately there is not a single standardized way to access these facilities, instead there are many protocols and tools used by different scientific, research, educational or private centers. In this regard, there is a clear need for a common standard to develop and execute large-scale scientific parallel and distributed applications. An innovative framework able to scale from single resources to multiple resources having different ownership, geographic locations, architectures and policies is a step forward in a new computational era.

In this section we propose such a framework, having the following properties:

- **Resource heterogeneity** – A hybrid-computing environment usually consists of multiple distinct types of resources. To overcome this heterogeneity, we proposed a software layer (a middleware) able to hide to the users the differences and the complexities of the incorporated computing infrastructures.
- **Scalability** – A system is described as scalable if the performance of the system remains the same even if the number of resources, requests and/or resources increases. The scalability in our case is assured by the scalability of the underlying computing

infrastructures but a general overview of the entire system is kept by a monitoring component.

- **Transparency** – Offering transparency to the users if very important. All the computing infrastructures, their interoperability and their underlying complexity are hidden from the user under a centralized system. The transparency can be handled at different levels:
    - **Access Transparency**: the users access the resources in a uniform manner, regardless of their type.
    - **Location Transparency**: the resources belonging to the same or to different infrastructures can be physically located in different parts of the world but the system provides uniform access to all resources.
    - **Failure Transparency**: failures should be detected and masked to the user till their recovery.
- **Fault handling** – The system provides a Fault Handling component able to detect and recover from possible failures.

Based on the above-described properties, we present the main framework functionalities:

- **Applications management**: the framework offers an application management component through a conceptual description of the structure of the applications, which gives a better view on the execution and data flows. It also offers functionalities to map the conceptual description of the application to physical resources and schedule its execution on a HCE.
- **Internal and external data management**: all distributed systems offers tools for data management but most of the time they can become complex and hard to understand and use. Our purpose is to offer a unified and transparent way to manage both the internal and the external data among the hybrid set of distributed infrastructures.
- **Access to a hybrid set of computing resources**, which implies also **authorization and authentication** to these resources. The framework offers access to a dynamic set of heterogeneous resources. Most of the time, the access to a set of resources only comes with user authentication and authorization processes. This ensures the security of the running applications on the accessed resources. The authentication offers a way to

establish the identity of the user while the authorization tells us what are the set of resources accessible by the user.

- **Conceptual description of environmental applications**: a simple and flexible conceptual model is available to the user to map an environmental application and to better understand its structure, its task decomposition, its execution, and its data flow.
- **Intelligent optimization of resource selection based on different criteria**: the framework provides an intelligent component (**Mediator**) able to optimize the selection of computing resources assigned to the execution of the application workflow tasks. This selection is done base on several criteria such as user preferences, tasks type, history, resource availability, and constrains.
- **Mapping, scheduling and execution of applications on a hybrid environment**: after conceptually defining the application, the user has functionalities to submit to execution of the workflow of tasks composing the application. The user has the option to specify different execution preferences and constrains. The framework offers a scheduling algorithm, which aims for a better resource utilization and better performances while following the imposed constrains.
- **Error prevention and recovery mechanisms**: the errors can appear at different levels in the implementation of the ENV2CE methodology, starting with the definition of the conceptual model of an application and ending with the result collection. The framework must ensure a robust way of detecting the errors, no matter the level in which they appear, as well as to provide mechanisms of error recovery if necessary.
- **Execution monitoring and results management**: monitoring the execution of the application on a HCE implies many functionalities: manage the individual distributed systems components and present a unified view of them, keep the user informed regarding the execution state, detect possible errors and adapt the context specific recovery mechanism in order to have a better success rate of execution.

The main idea of our hybrid environment is to provide access to a large set of highly heterogeneous resources, which differ in basic properties such as processor architecture, amount of memory, storage capacity, connecting network and performance. The goal is to provide a uniform perception of these resources by the end users. This means that all the technological

complexity should be hidden and all the resources should be seen similarly, whether they are located on a cluster in the same building or on a remote Cloud.

Hiding the technological complexity of such a system means dealing with numerous issues:

- The difference of resources not only in basic properties but also in small details. Important aspects of the resources include (Mateescu et al., 2011 [146]):
    - resource ownership: locally owned or externally owned;
    - resource accessibility (private or public);
    - resource sizing: quasi-static (resource grow when purchasing new hardware) or dynamic (by using external public resources);
    - resource allocation policies: exclusive (per organization, group, project or user) or shared (among organizations, groups, projects or users); and
    - application portability: tied to a specific platform or platform agnostic.
- The middleware and the tools used to access each resource, as each infrastructure has different access interfaces. This entails the particularities and the language of jobs submission for each platform;
- Data access of each resource;
- Security issues;
- Lack of connectivity between resources;
- Performance;
- The monitoring components, which may vary even within the same environment; and
- Procedures and tools for result collection.

10.4.1. Distributed Computing Infrastructures Technical Specifications

In this section we will present the technical specifications of all the parallel and distributed infrastructures used in our research. They are all members of our proposed HCE.

The need to access multiple computing platforms concurrently from a single framework/application comes often due to the impossibility to reserve a sufficient amount of computing resources at once within a single platform. Even if one platform would have the required number of resources needed for a large scale complex scientific application, in real

world that platform is not dedicated to a single user but to a distributed pool of end-users who also demand access to a range of resources.

The common goal of all distributed infrastructures is to provide efficient and transparent computing access over a distributed set of resources with as little effort as possible from the user point of view. Almost all these resources are nowadays equipped with multi-core processors and many core 'add-ons' which makes them more and more difficult to program, use and port applications on them, even for a computer scientist. Furthermore, combining heterogeneous resources together increases the difficulties and the issues to handle. The potential parallelism must now be explored at all levels of granularity.

10.4.1.1. Grid Infrastructure (gLite Middleware) (UTCN)

- gLite middleware;
- Resources:
  - one Computing Element (CE) and one Storage Element (SE);
  - Worker Nodes (WNs) – computational resources – 128 physical CPUs with 1024 logical CPUs;
  - Storage Element – storing resources ~ 13 TB;
- Ganga tool – used as frontend for job definition and management;
- Diane tool – employed for efficient usage of the distributed computing infrastructures.

10.4.1.2. Baobab Cluster (UNIGE)

- HPC Cluster – BAOBAB – provided by UNIGE (http://baobab.unige.ch/);
- bought by the end of 2012 and operational since early 2013;
- master node in charge of administration and backup;
- 57 compute nodes, each with:
  - 2 Sandy Bridge Intel(R) Xeon(R) CPU E5-2660 @ 2.20GHz cpu with 8 cores each;
  - 64 GB of RAM;
    - for a total of 912 cores.
- 1 compute node with:

    - 4 Sandy Bridge Intel(R) Xexon(R) CPU E5-4640 @ 2.40GHz cpu with 8 cores each.
- a server node providing 40 TB (extended in December 2013) shared file system **FraunhoferFS (FhGFS)**;
- **InfiniBand** 4xQDR (40Gbit/s) connectivity between master, nodes and storage.

10.4.1.3. OpenStack Cloud (UNIGE – HEPIA)

- HepiaCloud - academic cloud platform based on OpenStack, connected to SwissACC (Swiss Academic Compute Cloud) platform/project;
- **Available resources -** 41 compute nodes available;
- vcpus free: 287 / 304;
- ram free: 1953 GB / 1992 GB;
- disk free: 9060 GB / 9148 GB;
- All virtual machines in a private network which require the usage of an ssh gateway (gw.lsds-rg.org).

10.4.1.4. OpenStack Cloud (UNIGE - SwissACC)

SwissACC – Swiss Academic Compute Cloud (http://www.swing-grid.ch/SwissACC/) is a Swiss wide computational science platform offering:

- Resources;
- Services;
- High quality know how for user and application support;
- Provides access to Cloud platforms - UZH Cloud (Hobbes) http://cloud.gc3.uzh.ch/;
- Compute nodes available: 80;
    - vCPUs available: 80;
    - RAM available: 160 GB.

10.4.1.5. Windows Azure Cloud (UNIGE – SwissACC)

The available Windows Azure platform has only 4 virtual machines and it's a test platform. This is used just as a proof of concept to show the flexibility of the proposed framework to integrate at any time other Cloud providers or even other DCIs.

10.4.2. System Architecture

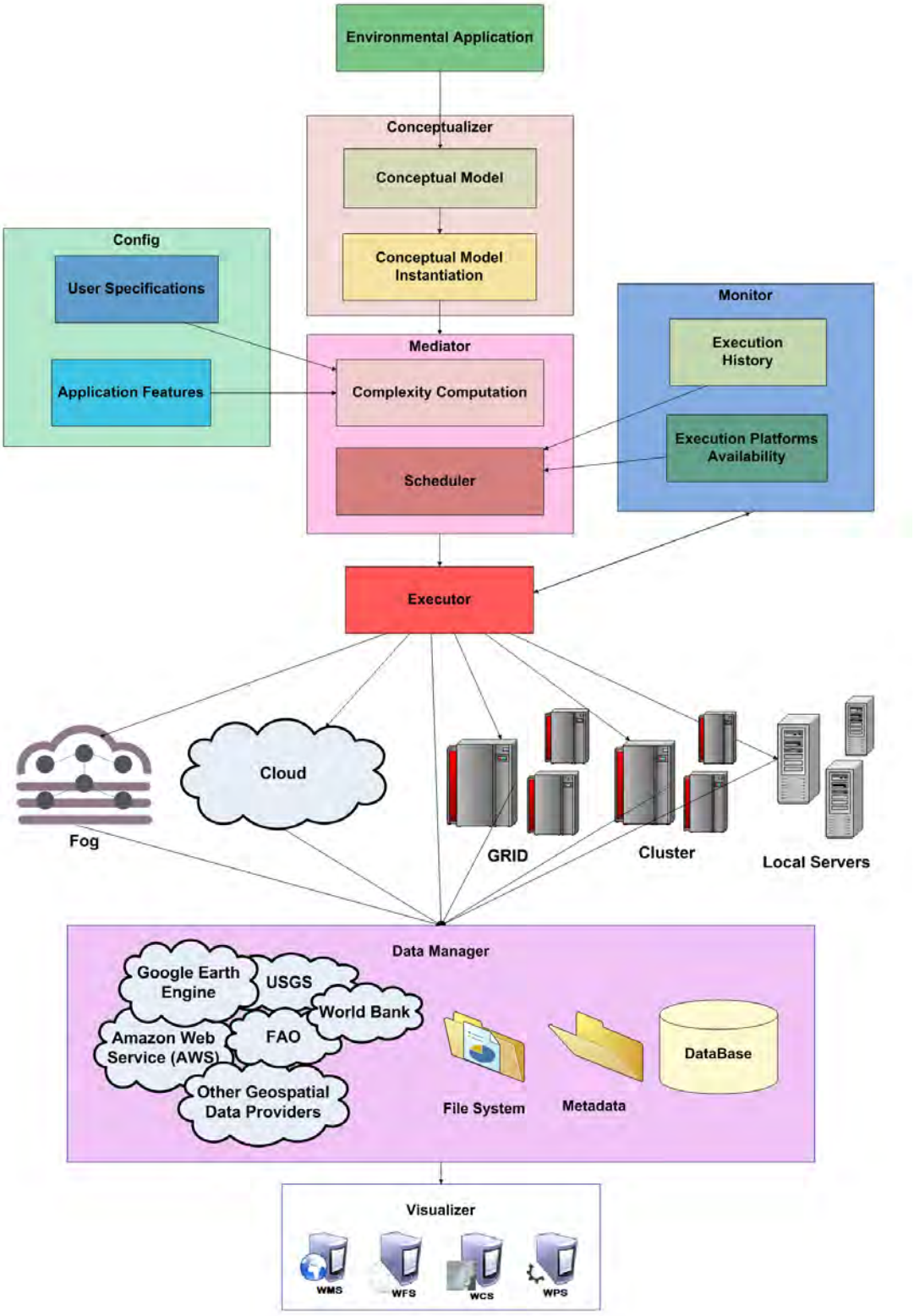

Figure 77: ENV2CE - System Architecture

The System Architecture is a typical multi-tier client server architecture, customized to meet the proposed system requirements and to follow the proposed methodology with focus on the interoperability and openness of the included components. Figure 77 illustrates a possible implementation scenario.

The Client Layer provides end users capabilities to insert new applications into the framework, to create conceptual descriptions of the applications, to instantiate conceptual models to real use cases data, to insert applications specifications, execution preferences and execution constrains, as well as to visualize monitoring reports and resulting data. The visualization of geospatial data can be done through a set of OGC standards implementations such as Web Map Service (WMS), Web Coverage Service (WCS), Web Feature Service (WFS) and also to be post processed using the Web Processing Service.

The Application Layer contains the core services performed by the system and is composed of several important modules: Conceptualizer, Mediator, Executor and Monitor.

10.4.3. Components Description

10.4.3.1. Conceptualizer Component

The Conceptualizer is responsible for generating the conceptual model of an application, instantiating it with specific data and computing resources and managing the application execution and data flows. At this stage, the workflow of the application is build and the workload is divided into sets of independent tasks. The Conceptualizer is therefore responsible for managing the application workflows, including the metadata and any other additional information related to the application workflow.

This component includes a parser responsible for parsing the application conceptual model described by the user (detailed in section Chapter 8:). At this step, all the information regarding the data and the execution workflow of the application are extracted and incorporated in an internal data structure. All the conceptual models (applications workflows at this stage) are stored in the database and can be further on used for other executions. After the scheduling solution was decided (Scheduler Component), the application workflow is instantiated with actual input data and actual execution resources.

10.4.3.2. Mediator Component

The main goal of the Mediator Component is to achieve the integration and the interoperability between Environmental field (Environmental data and applications) and High Performance Computing (the Hybrid Computing Environment). The interoperability between two systems in general, defined as the state in which the two systems can work, communicate and interchange information while taking advantage of both systems capabilities and functionalities, is a challenging research topic. There exist two main approaches to achieve this state of interoperability between two systems, which are not compatible: 1) development and implementation of common standards that would allow an easy communication and exchange of information, and 2) a mediation (broker) layer, capable to mediate the differences between the two systems. The adoption of common standards is definitely the best way to achieve interoperability but in this ideal case, the two systems must also support the adherence of these new common standards. Both the definition and the implementation of standards are long and complex processes and involve many organizations and collaborations. The mediation approach involves the insertion of a new component capable to mediate the communication between the two systems and to easily explore their capabilities and functionalities.

The Mediator Component is developed on the broker pattern, which separates users of services (clients) from providers of services (servers) by inserting a new, intermediary level, called broker. When a client needs a service, it has to interact with the broker, using a service interface. The broker then forwards the client's service request to the appropriate server, which will process that request and send back the result. This component incorporates the Complexity Computation and the Scheduler Components and supports protocols and data models found in environmental domain.

An important functionality of the Mediator component is to decide on which infrastructure to submit which tasks and this implies much more than just analyzing detailed performance metrics of each available computing infrastructure. The motivation of developing this component was to allow users who have access to a limited local set of resources and to a potentially much larger pool of distributed resources, to easily make use of all of them in an efficient manner while also hiding the complexity of the underling computational infrastructures' mechanisms.

The functionalities exposed by each of the available distributed computing infrastructures (DCIs) within the HCE can provide many benefits, which are visible only in certain conditions. These infrastructures (Grid, cluster, Cloud, etc.) are not just powerful platforms that execute any process faster by making use of the parallel architectures on which they are built. Many other aspects have to be considered regarding the complexity they hide. In most of the cases, the functionality offered by these infrastructures can improve the execution time of a complex computation by taking advantage of the parallelization techniques. There are still cases in which the DCIs can have negative consequences, increasing the execution time of a process due to the overhead introduced by the management that lies behind (i.e. job creation, scheduling, submission and management, monitoring, result collection, etc.). These are usually the cases for simple processes, which do not need a high amount of resources. The cases in which DCIs and implicitly a HCE comes with an improvement in the execution time for environmental applications are particular. These cases are described for Grid infrastructure in (Rodila and Gorgan, 2010 [195], 2011 [197]). Before using a HCE and particularly the composing DCI, one should consider the full picture and look at the complexity that hides beyond each of these infrastructures. They can offer great advantages for performing large and complex computations but it slows down the small processes. The analysis of the boundary above which a DCI is beneficial in executing service requests is therefore an important part in the migration of environmental applications on a HCE and the Mediator component tries to find appropriate solutions.

The identified cases in which a DCI can improve the performances obtained while executing an environmental application/service request are:

- The time required to execute a request on a local server is considerably larger than the overhead introduced by job creation and management on any of the available DCIs;
- Several requests are made in parallel at the same time;
- The request (environmental application) is a complex request that can be split into several independent sub-requests, depending either on the requested data or on the requested functionality.

Before porting an environmental application on a HCE, all these cases should be taken into consideration and analyzed to decide whether the given request falls in one of the cases.

Should it not be the case, the process of migration for this special case is useless and the request should be better executed on a local server. This analysis should be made for each individual request and should be part of the extended service either as an incorporated or as an additional component.

The Mediator Component is responsible for the following activities:

- Analyze the environmental application, base both on the user specifications in configuration file and the data and execution flows extracted from the application conceptual model;
- Schedule all the application tasks based on different criteria: complexity estimation, execution history, user preferences, etc.;
- Provide functionality for data parallelism at the task level if necessary, based on data requirements;
- Applications and resources management.

10.4.3.3. Complexity Computation

The Complexity Computation Component is responsible for estimating the computational complexity provided by the application into runtime and/or costs estimates on a specific resource. The complexity analysis is a complex task and must consider different inputs such as: the type of the applications (as specified by the user in the configuration file), the data and the execution flows (as derived from the application conceptual model), the complexity estimated by the user, etc. Base on the estimations made by this component, the scheduling process adjusts the selection of resources for each application task to obtain better performances.

Estimates of the relative complexities of the tasks to be scheduled are used to identify potential scheduling blocks. An initial hybrid mix of computing resources is determined based on user/system defined objectives, policies and constrains. Tasks that are computationally intensive may be more suitable for a cluster or a HPC Grid resource while tasks that require quick turnaround may be more suitable for a Cloud resource. A low-level complexity task or even application (simple execution or serial execution flows in the application conceptual model) may also be more appropriate to be executed on local servers instead of migrating it on one or more of the available DCIs within the HCE. High complexity tasks can also be further analyzed before

scheduling, to analyze the data parallelism option if the task input data is quite large. The allocation of tasks and the scheduling policy can change at runtime.

The Complexity Computation component acts thus as a filter for the incoming requests (applications and tasks) and based on the results, it assigns different priorities for the available DCIs. Complex requests have to be further on analyzed before scheduling them for execution, to determine the level of data parallelism that can be further on applied.

We define the complexity of an application as a function of different input parameters:

EnvAppComplexity = f (InputDataSize, InputDataNrOfFiles, NrOfWorkflowTasks, UserEstimatedComplexity)

Similarly, we define the complexity of a task as:

TaskComplexity = f (InputDataSize, RequiredProcessingPower, RequiredProcessingMemory, UserEstimatedComplexity)

Based on these initial estimations, we compute an estimation matrix in which we estimate the optimization criteria from mapping each task on each available and possible computing resource. The optimization criteria chosen here were the total execution time and the total execution cost.

10.4.3.4. Scheduler

As described in Chapter 9 scheduling is one of the most important problems in computing system. It is a dynamic activity and has to be described with both static and dynamic properties, at different instances of time. This activity can sometimes take a significant amount of time, especially when the complexity of the scheduling problem becomes sufficiently large. We should make sure that the execution time saved by selecting the optimal schedule, considering the specified context, is not less than the increase in time required to find that schedule.

Shortly, scheduling is the process of assigning tasks to resources (machines) and then ordering the tasks for execution on each resource. To map *n* tasks (T_1, T_2, …, T_n) to *m* resources (R_1, R_2, …, R_m) (Figure 78) we have to consider the tradeoff between the quality of the solution and the time needed to get that solution.

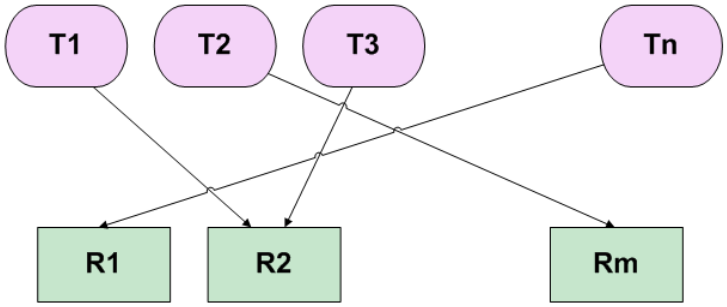

Figure 78: Task Scheduling on Resources – Abstract View

A scheduling problem may have one or more scheduling solutions and in this section we will consider a range of possible solutions. At execution, the goal of the scheduling algorithm is to always choose the solution that is closer to the optimum one, from the range of available solution at that time, in the given context.

Within a scheduling problem there are two important phases that have to be considered (Hanh and Simonenko, 2015 [102]): the processing requirements required by a task (CPU, memory, etc.) and the optimization requirements (minimum parallel executing time, minimum execution cost, high resource utilization, etc.). Based on these two phases, the first step is to reduce the number of resources available for executing the tasks (reducing in the same time the space of scheduling solution) to the set of resources satisfying the processing requirements. Based on the new set of available resources, the second step consists in choosing the scheduling solutions satisfying the optimization requirements. Among these solutions (a reduce space) we can find the one that converges towards the optimal solution, considering the defined context and requirements (Figure 79).

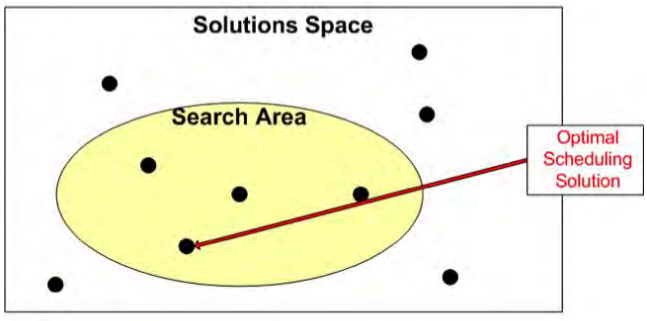

Figure 79: Scheduling Solutions Space - Abstract View

Let us define in the following the scheduling problem together with its important component: the workload, the resources and the scheduling goal and requirements. All these concepts were described in detail in Chapter 9.

We consider therefore the scheduling problem SP (Figure 80) defined as a set of tasks, a set of resources and a set of scheduling requirement:

$$\mathbf{SP = (T, R, Z)} \quad \mathbf{(1)}$$

where:

SP = scheduling problem;

- $T = \{T_1, T_2, \ldots T_n\}$, $n > 0$ – list of tasks defining the workload;
- $R = \{R_1, R_2, \ldots, R_m\}$, $m > 0$ – list of heterogeneous resources (belonging to different computing infrastructures);
- $Z = \{Z_1, Z_2, \ldots\}$ – list of scheduling requirements that must be fulfilled (optimality criteria).

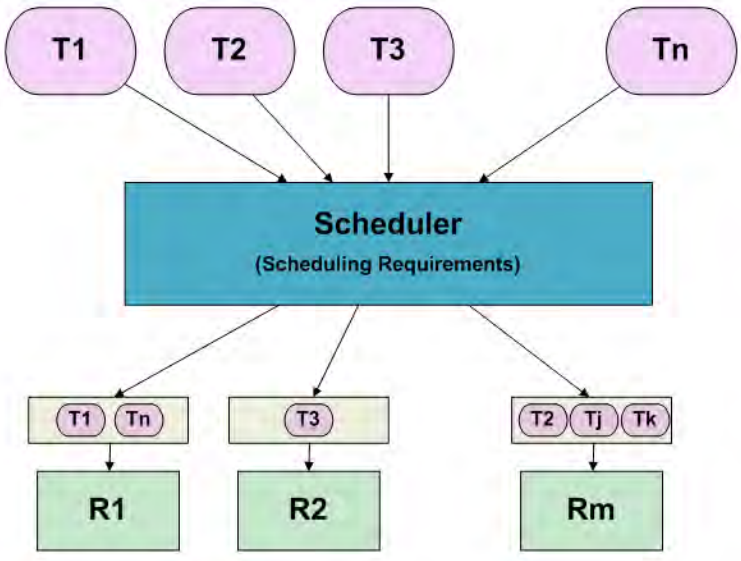

Figure 80: Scheduler Model

The set of tasks is application dependent and it's extracted based on the conceptual model of the application, defined in Chapter 8: . During the application execution, the conceptual model of the application is build and the workflow with the corresponding tasks is extracted. The tasks of the workflow should be assigned (mapped) to available required resources, which can belong

to any of the composed distributed computing infrastructures (DCIs) within the HCE. Based on the final set (pool) of selected resources, the final execution environment is determined at runtime and the Scheduler submits and executes the tasks according to the resources' type. The central point of this execution flow is the appropriate selection of the resources (from the highly heterogeneous pool) to map the current set of workflow tasks.

The set of resources depends on the HCE and the composed distributed computing infrastructures. Each infrastructure has different types of resources, which have a high degree of heterogeneity even within that specific infrastructure leading therefore to an even higher degree within the HCE. For example Grids have active resources (CEs) that continuously execute submitted jobs while desktop Grids have volatile resources that can become unavailable (they go offline) anytime the owner wants to use that computer. Clouds have virtual resources that have to be deployed before being used, affecting the load and the waiting time of the task. When defining the set of heterogeneous resources within a HCE, we have to take into account that all types of resources introduce a certain amount of delay that affects the execution time of a submitted task.

The resource selection should take into account all the common and specific parameters of each computing infrastructure, described in details in Chapter 9: , but will follow also specific rules:

- Each used distributed computing infrastructure (DCI) will be assigned a certain priority weight. This weight is computed based on several factors:
    - User's access to the DCI;
    - DCI resource availability (obtain the current load either through information system interrogation or through estimation);
    - DCI execution cost (which can vary based on the task type);
    - User budget (especially when assigning weights to public Cloud infrastructures);
- The infrastructure with the highest weight will be selected as the first pool of resources to map the application jobs. Selection of resources from another infrastructure (having a lower priority weight) is done in the following cases:
    - When resources of the current infrastructure become exhausted – the current infrastructure cannot cope with the current workload to be executed;

- When task requirements (hardware and/or software dependencies) cannot be satisfied with the resources from the current infrastructure – no computational resource is found that has the specific required properties (encountered especially when complex software need to be executed). Installing the missing software or libraries might not always be possible and regarding the hardware dependencies, this is impossible to be modified;
- When workload distribution is needed – need to access a larger pool of resources to obtain a higher throughput in terms of resource usage (depends on the user budget). This case is more appropriate to independent tasks when no communication is needed.

A. Workflow description based on the application conceptual model

A workflow is an abstract representation of an application, deduced in our case from the environmental application conceptual model, and it's composed from a set of tasks, with precedence constrains between them, which can be executed in sequential mode, in parallel mode or a combination of them. We represent a workflow using the notation:

$$\mathbf{W = (T(W), B(W), maxST(W), ET(W))} \quad \mathbf{(2)}$$

Where:

- $T(W) = \{T_1, T_2, \ldots, T_n\}$ – List of workflow tasks;
- B(W) = User defined workflow budget (defined for public Cloud resource usage);
- maxST(W) = User defined maximum start time of the execution;
- ET(W) = User defined estimation of total execution time.

B. Task description

A task is the description of a compute or data transfer process, its dependencies (hardware and software requirements), and it's constrains. We represent a task using the notation:

T_i = (Res(T_i), Exe(T_i), Dep(T_i), In(T_i), Out(T_i), maxST(T_i), ET(T_i), RT(T_i), D(T_i), P(T_i)) (3)

Where:

- Id(T_i) – Task id;
- Res(T_i) = Resource attributes for the task: resource type, resource count, …
- Exe(T_i) = Executable of the task;
- Dep(T_i) = List of dependencies: libraries, software, operating system, …;
- In(T_i) = Inputs of the task;
- Out(T_i) = Outputs of the task;
- maxST(T_i) = User defined maximum start time of the task execution. All maxST(t) must satisfy the condition: maxST(T_i) > = maxST(W), i = 1,n;
- ET(T_i) = User defined estimation of task execution time;
- RT(T_i) = The time when the task becomes ready for execution (all dependencies are completed, all the inputs are available) – this time is computed during runtime;
- D(T_i) = Deadline of the task;
- P(T_i) = Priority of the task.

We define additional temporary task parameters such as:

- ST(T_i) – actual start time of the task, which must satisfy the condition:

$$RT(T_i) <= ST(T_i) <= maxST(T_i);$$

- WT(T_i) – waiting time of the task – defined as: WT(T_i) = ST(T_i) – RT(T_i);
- Weight (T_i) = P (T_i)/$e^{D(Ti)}$ – weight of the task – the tasks with the higher weight (higher priority and a tighter deadline) should be mapped first. The exponential function is used to increase the impact factor of the deadline in the weight.

At the execution, a task can be in one of the following states:

- **Submitted** – the task has been sent to execution to a chosen resource;
- **Wait** – the task is in the waiting list to be mapped to a resource or the task is already; mapped to a resource but the resource is not yet available;
- **Running** – the task is being executed;
- **Done** – the execution of the task is successfully terminated;
- **Cancelled** – the execution of the task has been cancelled from different reasons.

C. Resource description

A resource is a reusable entity that is used to fulfil a task or resource request. This can be a machine, network, or some service that is synthesized using a combination of machines, networks, and software.

We define the following characteristics for a resource R_j:

$$\mathbf{R_j = (Id(R_j), P(R_j), CPU(R_j), Type(R_j), C(R_j))} \quad \mathbf{(4)}$$

- Id (R_j) – resource id
- *MIPS(R_j) – Million Instructions Per Second*
- *NFC (R_j) – number of free cores*
- *NPC (R_j) – Total number of processing cores*
- Performance P (R_j) = NFC(R_j) * (MIPS(R_j) / Nt(R_j)). For simplification, the performance can also be defined as the time taken to execute a task in seconds.
- CPU – the computing capacity of the resource – numbered of processes instructions per second.
- Type(R_j) – Resource type: 1 (cluster), 2 (Grid), 3 (Cloud);
- C(R_j) – cost per hour. This property if defined for Cloud resources, for all the other resources, this value is 0.

Other resource characteristics could be taken into consideration (memory, bandwidth, latency delay, etc.), but for the sake of simplicity we will not include all of them in our approach.

The workflow of the application can be divided into execution stages called jobs, based on the composed tasks and their interdependencies. At each stage of the workflow, the workflow manager determines the tasks that have to be executed at current stage as well as the relative computational complexity of each task. The number of tasks per stage and the complexity of each task can vary, making the runtime scheduling a complex process. Once the number of tasks to be scheduled within a stage have been identified, the scheduler analysis the tasks and their complexities to determine the appropriate mix of resources from each available infrastructure.

Further improvements in the scheduling mechanisms can be done by:

- Clustering tasks based on their complexities to generate blocks of tasks for scheduling;
- Estimating the runtime of each block on the available resources – using a cost estimator service;
- Determine the allocations and the scheduling policies based on runtime estimates and overall objectives and resource specific policies and constraints.

Once the resource allocations have been determined, the desired resources are provisioned and the workers are started, using the mechanisms appropriate for each individual computing infrastructure. During the execution, the workflow manager also monitors the execution of the tasks to determine the progress and to orchestrate the execution of the overall workflow. The scheduler also monitors the status of the resources and determines the progress to ensure that the scheduling objectives, the policies and constrains are satisfied. If this is not the case, the scheduler can dynamically change the resource allocation and scheduling policy as required.

Our scheduler will put all tasks in a queue and will decide how to schedule them in order to meet the following requirements (goals):

- Respect the time constrains of each task T_i, i = 1,n;
- Enforce task dependencies – initially all $RT(T_i)$, i = 1,n are set to 0; once the dependencies of a task T_i becomes available (processing requirements, input, …), $RT(T_i)$ is updated to the current time of the system;
- Respect the workflow constrains (time and budget) (user defined); Each constrain will have associated an importance weight (w_i) such as the sum of all weight equals 1

$$(\sum w_i = 1)$$

- Follow the defined selection rules.

A list of possible mapping solution (tasks over resources) can be obtained offline, prior to actual execution of the workflow within a HCE. Within this range of possible solutions, the Scheduler should be able to select at runtime the solution closer to the optimum one. For this, we define in the following the concepts of solution, optimum solution, distance between two solutions and convergence rules.

A scheduling solution (SS) is defined using the equation:

$$\mathbf{SS = (M (T, R), C)} \quad \mathbf{(5)}$$

Where:

- $T = \{T_1, T_2, \ldots, T_n\}$, $n>0$ – list of tasks needed to be scheduled;
- $R = \{R_1, R_2, \ldots, R_s\}$, $s>0$ – list of final selected resources;
- $M (T_i, R_j)$ – mapping of task T_i to resource R_j;
- C - estimated total cost for the selected resources.

Depending on the user requirements, specifications and constrains on the list of tasks (composing the application workflow), we define the cost using the following notation:

$$\mathbf{C = (Time (W), B)} \quad \mathbf{(6)}$$

Where:

- Time(W) –Execution time of the workflow (set of tasks);
- B – Budge spent on resources (for public Cloud resources).

$$\mathbf{Time\ (W) = \sum_i^g\ (Time\ (PTG_i))}, \quad \mathbf{(7)}$$

where Time(PTG) is the execution time of a parallel task group

$$\mathbf{Time\ (PTG) = max_j^p\ Time\ (T_j)} \quad \mathbf{(8)}$$

The total workflow execution time – Time(W) – is the sum of all the execution times of the groups of parallel tasks within that workflow – Time(PTG). A group of parallel tasks is a group of tasks that can be executed in parallel. The total execution time of such a group is actually the maximum execution time of the composed tasks. Now the execution time of a task $Time(T_i)$ is defined in our case using the formula:

$$\mathbf{Time\ (T_i) = Time\ (schedule) + Time\ (waiting) + Time(submission) + Time\ (execution) + Time(result\ collection)} \quad \mathbf{(9)}$$

An optimal scheduling solution (OSS) is a solution having a minimum cost (minimum time execution and minimum budget):

$$\mathbf{OSS = (M (T, R), minC)} \quad \mathbf{(10)}$$

The distance between a scheduling solution (SS) and the optimal scheduling solution (OSS) is defined as:

$$\mathbf{d = min \sqrt{\sum \alpha_i (Metric_i)^2}} \quad \mathbf{(11)}$$

Where

- α – represents a weight;
- Metrics – represents a selected optimal metric – in our case this will be the Time and the Budget. Other metrics can be added later.

D. Proposed Scheduling Algorithm:

Table 2: Scheduling Algorithm

Input: List of Task descriptions T, list of resources descriptions R and list of scheduling constrains Z.

Output: List of allocated resources R (which can come from different infrastructures)

```
1  FOR ALL Rj in R DO
2       eliminate Ri which do not satisfy any task performance
requirements
3  END FOR
4  FOR ALL Ti in T DO
5       FOR ALL Ri in R DO
6            compute estimation execution time EET(Ti, Rj)
7            compute estimation execution cost EEC(Ti, Rj)
8       END FOR
9  END FOR
10 PTG = extractGroupsOfParallelTasks()
11 finalPoolOfResources = []
12 computeInfrastructuresWeights()
13 infrastructures = sortAvaiableInfrastructures()
14 infrastructureNumber = getInfrastructureCount
(infrastructures)
15 FOR ALL TG in PTG DO
16      requestedResources = ResourcesRequest(TG, R)
17      count = getResourceCount(requestedResources)
18      i = 0
19      WHILE (i < infrastructureNumber) AND count > 0 DO
20           infrastructureId = getId(infrastructures(i))
21           reservationId = getReservation(requestedResources,
infrastructureId, count)
22           IF (reservationId < 0) DO // not enough resources
or not satisfying resources from the current infrastructure
23                (partialCount, partialResources,
reservationId) = getPartialReservation(requestedResources,
infrastructureId)
24                resourceIds = getResources(partialResources,
partialCount, infrastructureId)
25                addResourcesToReservation(finalReservation,
resourceIds)
26                count = count - partialCount
27                requestedResources = requestedResources -
partialResources
28                i = i + 1
29           ELSE DO // already reserved enough resources
30                count = 0 // exit loop
31           END IF
32      END WHILE
33      R = getAvailableResources()
34 END FOR
35 RETURN finalReservation
```

The first step of the algorithm is to eliminate the resources that do not match any task performance requirements (line 2). In this way we reduce the scheduling solutions space (Figure 79) by keeping only the resources that would possibly satisfy the optimization criteria. In line 6 and 7 we compute an estimation matrix (both for execution time and execution cost). Each cell of the matrix (i,j) contains an estimation of tasks T_i over the resource R_j. This matrix will be use to reserve resources for a group of tasks in such a way to maximize the optimization criteria (execution time and/or cost). In line 10 we extract the groups of parallel tasks from the application workflow. For each such group, we have to also compute the maximum task execution time in the group, for equation (7) and (8). In line 12 we compute a certain weight for each available computing infrastructure. This weight is based on the user access to the infrastructure, user preferences on some infrastructures, optimization criteria, and estimation matrix.

InfrastructureWeight = f (UserAccees, UserPreferences, OptimizationCrateria, EstimationMatrix)

The selection of the resources always starts with the resources satisfying the optimization criteria and coming from the infrastructure with the highest weigh. When all the resources from the current selected infrastructure are finished or are not capable of satisfying tasks requirements, we move to the next infrastructure (next lower weight) (line 28). The weights of the infrastructures can also be manually set by the user before the execution.

For each group of tasks (line 15) we select (reserve) the required set of resources (line 21) from the current infrastructure. If the resources are not enough (line 22) we move to the next infrastructure until we get the necessary set of resources or we no longer have available resources. For each group of parallel tasks, we recompute the available resources in line 33 as after each execution, most of the resources become available. In the case of Cloud resources, a script has to be scheduled to release the virtual machine once the execution is ready. The ideal case would be to also compute the costs and time needed to keep running the virtual machines instead of shutting them down and turning them on again at the next possible execution.

The algorithm returns a final set of resources, for executing all the groups of parallel tasks, and this set of resources represents a scheduling solution. The goal is to choose a solution as close as possible to the optimal one and we try to achieve this using the formula (11) based on the minimum distance between solutions.

10.4.3.5. Executor Component

The execution of an application on one infrastructure cannot be ported and applied on another one. The ideal case would be to have a standard way for the execution of an application on distributed resources, regardless of their type and membership. This would be a step forward in solving a large set of environmental problems.

The Executor component proposed in this research is responsible for actually launching into execution the workflow of the application (with the composing tasks) based on the defined scheduling policy. This component is formed from different infrastructure specific Adaptors, which are responsible for provisioning the resources on their specific infrastructure, configuring workers as execution agents on these resources and assigning the corresponding tasks to these workers. A platform Adaptor is a complex module, which has to handle and to mediate all the incompatibilities between the executed application and the execution platform. It has to consider specific platform issues such as communication protocols, connectivity, resource management, execution monitoring, fault recovery, etc. These platform Adaptor components are specific for each integrated execution platform and represent an essential point for the execution flow in the presented mapping framework.

10.4.3.6. Fault Handling

Distributed systems are large collections of heterogeneous loosely coupled resources and therefor is inevitable that some of them may fail due to numerous and diverse reasons. As failures are inevitable, some mechanisms are needed to solve faults related problems such as:

- Failures detection;
- Failures transparency for users;
- Failures recovery;
- Build Redundancy – redundancy is the best way to deal with failures and it's achieved by duplicating data so that if one component crashes, another one will be able to provide the required information.

10.4.3.7. Data Level

The data level contains all the programs that hold the actual data on which the application operate. These programs can be different databases able to incorporate environmental data (such

as PostgreSQL, RASDAMAN) or different environmental data providers (their databases) such as USGS, Google Earth Engine, Amazon AWS, World Bank, FAO, etc. Besides the databases and the data providers, in the data level we can also have different file systems as well as metadata catalogs. An important aspect about this level is that data is persistent, meaning that the data will be stored even when the system is not in used and can be accessible at any moment later. The data level is not only responsible for holding the data but also for making it consistent. This means that whenever some information changes, the metadata are updated too. Another important aspect about this level is that it's independent of the processing level. The execution of an application is not affecting the structure of the data at this level and neither the other way around.

Environmental applications can operate on different complex data types, some of which can be modeled in terms of relations but some others are more easily modeled in terms of objects. This is one of the main reasons why the selection of the database cannot be just a simple relational database but instead choose a mix and have also object oriented or object-relational databases.

10.5. ENV2CE Execution Flow

A flow of the execution of an environmental application on such a HCE has the following steps and is illustrated in Figure 81.

1) Insert application specific information: application type, execution preferences, constrains;
2) Create a conceptual model of the application;
3) Validate the conceptual model;
4) Instantiate the conceptual model with actual application data;
5) Validate the instantiated conceptual model (existence and availability of the mapped data);
6) Create an internal representation of the application based on the instantiated conceptual model;
7) Intelligent resource allocation and scheduling for each application tasks based on preferences, constrains, execution history, resource availability, etc;
8) Execution of application based on the proposed schedule;

9) Error detection at the execution level and application of error recovery mechanisms if necessary;
10) Execution monitoring and reporting; and
11) Execution History Management.

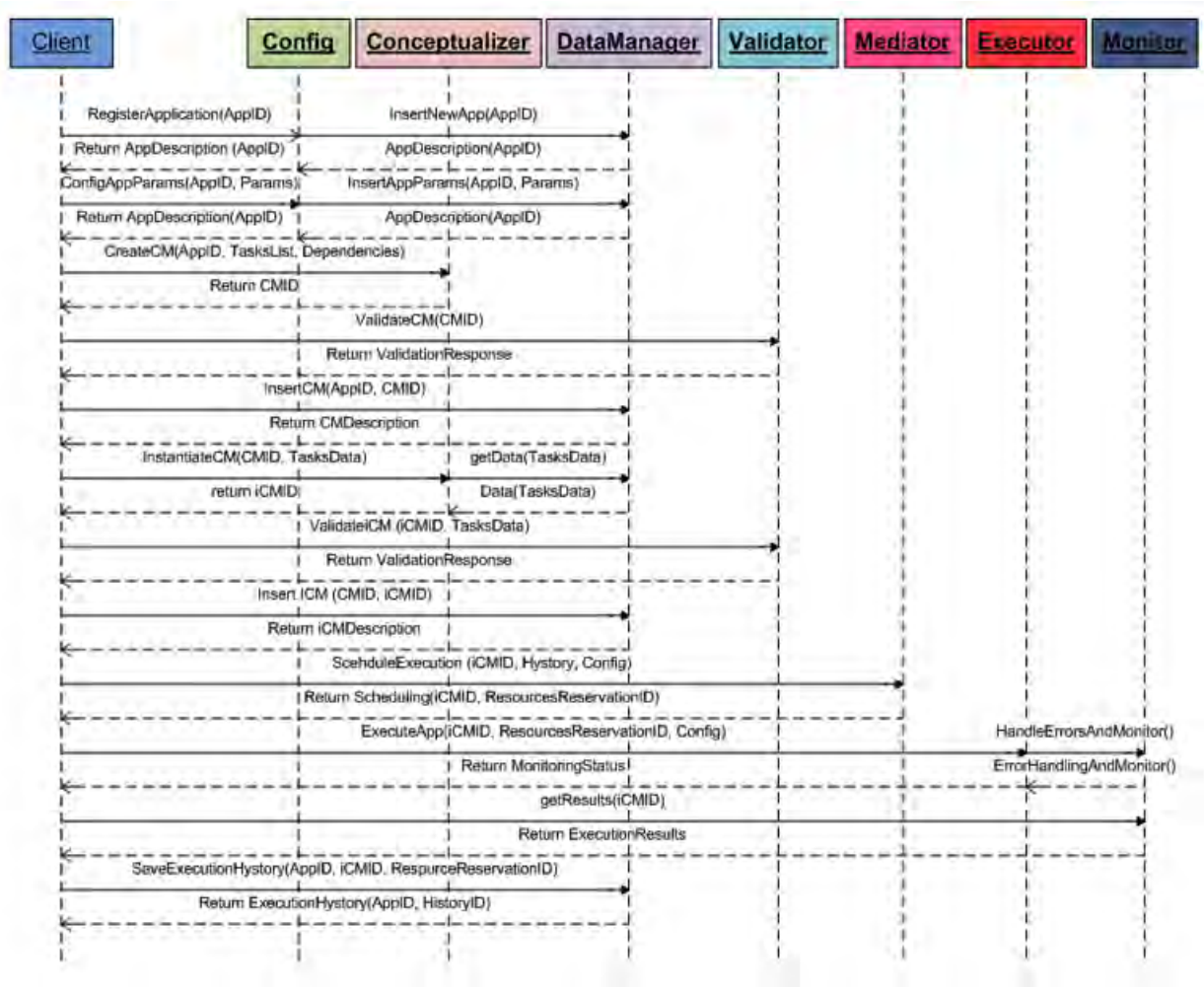

Figure 81: ENV2CE - Execution Flow

10.6. Conclusions

Production computational infrastructures will soon start to provide hybrid computing environments that integrate different computing infrastructures such as traditional HPC clusters, Grids, Clouds and possible even other new concepts. Understanding the potential benefits of such a hybrid infrastructure, and the mechanism behind, is important. In this chapter we have

experimentally investigated possible ways of integrating heterogeneous resources from different computing infrastructures to improve different objectives as well as how environmental applications can benefit from such a hybrid computing environment (HCE). We have introduced a new methodology – ENV2CE to efficiently execute large-scale environmental applications on a HCE. We have also proposed a framework architecture able to manage such an execution and we presented in detail its components and functionalities. The proposed methodology is based on the definition of a conceptual model, which will facilitate and simplify not only the understanding of the structure of the application but also the general execution and data flow. This conceptual model hides the complexity of different types of environmental applications and provides and easy and flexible way to map these applications on different heterogeneous computational resources.

The usage of a mix set of resources from different computing infrastructure is not just a case of gathering more resources for a problem but rather an experiment to demonstrate how different resource types and underlying resource management paradigms can complement one another to obtain better results (better execution times, better performance costs, etc.). The optimum execution environment is chosen based on a computed complexity of the received request, on the availability of the accessible platforms but also on the previous executions (history). The complexity of a request takes into account the application features, the parameters of the request (amount of data to be retrieved or processed, processes to be performed), the user specifications/preferences, etc. After selecting the optimum executing environment, the requests have to be adapted to be able to run on the chosen platform/platforms. This step is achieved for each available platform by the corresponding adaptor. A platform adaptor is a complex module, which has to handle and to mediate all the incompatibilities between the executed application and the execution platform. It has to consider specific platform issues such as communication protocols, connectivity, resource management, execution monitoring, fault recovery, etc. These platform adaptor components are specific for each integrated execution platform and represent an essential point for the execution flow in the presented mapping framework.

The definition of such a methodology (and supporting infrastructure) came as a response to the urgent need of the environmental community in using as much computational resources as possible to efficiently and effectively analyze and process an increasing amount of data.

10.7. Personal Contributions

- Introduce a new methodology and framework (ENV2CE) for efficiently porting and executing environmental applications on a Hybrid Computing Environment (HCE).
- Propose a Mediation solution for the interoperability between environmental applications and different distributed computing infrastructures (HCE).
- Execution of application tasks based on platform specific adaptors.
- Propose a new scheduling algorithm for a HCE, considering user specifications, application complexity, history, and optimization criteria.
- Published Papers:
    - ***Rodila, D.,*** *Ray, N., and Gorgan, D. (2015). Conceptual Model for Environmental Science Applications on Parallel and Distributed Infrastructures, Environmental System Research, Vol. 4/23, 2015, http://dx.doi.org/10.1186/s40068-015-0050-1.*
    - ***Rodila, D.,*** *Gorgan, D., Ray, N., and Lehmann, A. (2016). ENV2CE: Environmental Application Conceptualization and Execution on a Hybrid Computing Environment - Framework and Methodology Proposal, (to be submitted).*
    - ***Rodila, D.,*** *and Gorgan, D. (2012). Mapping Geospatial Applications onto Parallel and Distributed Environments, in ePaMuS 2012 – 5th International Workshop on Engineering Parallel and Multi-Core Systems – The Multi-Core Workshop, Palermo, Italy, 4-6 July, 2012, pp. 443 – 448, http://dx.doi.org/10.1109/CISIS.2012.152.*
    - ***Rodila, D.,*** *Bacu, V., and Gorgan, D. (2012). Geospatial Applications on Different Parallel and Distributed Systems in enviroGRIDS Project, in European Geosciences Union - General Assembly EGU 2012, Vienna, Austria, April 22-27, 2012, (abstract).*
    - ***Rodila, D.,*** *and Gorgan, D. (2011). A Mediation Approach in Geospatial Web Services Gridification, in ICCP2011 – IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 25-27, 2011, pp. 541-548, http://dx.doi.org/10.1109/ICCP.2011.6047928.*

- *Gorgan, D., Stefanut, T., Bacu, V., Mihon, D., and* ***Rodila, D.*** *(2010). Grid based Environment Application Development Methodology, in Large-Scale Scientific Computing, Springer Journal LNCS 5910, ISBN 978-3-642-12534-8, pp.499-506, http://dx.doi.org/10.1007/978-3-642-12535-5_59.*

Chapter 11: Framework Evaluation and Validation

11.1. Introduction

In Computer Science, most of the times, immediately after we ask ourselves the question on how to solve a problem, we think about the efficiency of the solution and the question becomes: how to solve the problem efficiently. Before finding an answer to this question, one should understand how to compare the relative efficiency of different solutions.

In this thesis, we have explored and analyzed solutions for easily porting and executing environmental applications on different parallel and distributed infrastructures, such as cluster, Grid and Cloud. Based on our theoretic and heuristic approach, we proposed a new methodology and a new framework architecture for easily and efficiently executing environmental applications on a Hybrid Computing Environment (HCE). We considered that it is important here to mention that the execution efficiency in this case is relative to the standard execution procedures used in Environmental Sciences, where the applications and the processing are usually performed on local machines. We have shown in the thesis that the capabilities of different distributed computing infrastructures, forming the HCE, bring an efficiency improvement in the execution of applications only in certain cases and we have treated and filtered these cases through a Mediator component and using estimations of the complexities of application and its processes/tasks.

11.2. ENV2CE – Components Validation

Considering the high complexity of a Hybrid Computing Environment (HCE), regarding technical settings, installation, and management, the overall validation of the proposed methodology and framework was hard to achieve. We consider therefore a component based validation and evaluation, concentrating our efforts more on the validation process than on the contextual technical aspects. We present in the following how we achieved this.

The ENV2CE methodology and framework were developed based on a heuristic approach achieved through a list of experiments evaluating the execution of different environmental applications on different parallel and distributed infrastructures. These

experiments were carried in different national and international research projects (MedioGRID, SEE-GRID-SCI, GiSHEO, enviroGRIDS, enviroPAD, GAR, UNEP Live, LiMES) and are part of a series of functional and tested environmental applications within these projects: gProcess, ESIP, gSWAT, GreenLand, EDAP and LiMES. These experiments were also validated through publications in different peer reviewed journals and conference proceedings papers, project deliverables and scientific presentations (the full list of publication is attached at the end of the thesis).

Versions of the conceptual models presented in this research have been validated within the gProcess platform (Bacu et al., 2009 [15], Rodila et al., 2009 [194]), which was further used in functional environmental applications such as ESIP (Gorgan et al., 2009 - 24), gSWAT (Bacu et al., 2011b [17], 2013 [18], Rodila and Gorgan, 2012 [198], Rodila et al., 2012 [199]), GreenLand (Mihon et al., 2010a [150]) and GreenView (Mihon et al., 2010b [151]). The gProcess platform was validated especially through the development and execution of applications for satellite images processing.

The Mediator component was tested and validated inside the enviroGRIDS project (http://www.envirogrids.net/), mostly in the gridification of the OGC Web services (Rodila et al., 2010 [196], 2011 [197], 2012 [198]) and in the execution of SWAT hydrological model on the Grid infrastructure, using the gSWAT application (Bacu et al., 2011b [17], 2013 [18], Rodila and Gorgan, 2012 [198], Rodila et al., 2012 [199]). Within this component the Complexity computation functionality was also evaluated in effectively filtering the geospatial services requests and selecting the appropriate execution environment base on the requested data and performed processing.

For the Scheduler component we considered a simulation-based approach. To create a simulation environment as close as possible to the real situation, we developed a Python framework able to automatically and randomly generate the parameters and the simulation conditions. This framework was used to test the performance of the proposed scheduling algorithm. Due to the NP-Complete nature of scheduling algorithms and the complexity of a HCE, reasonable assumptions are hard to make for obtaining optimal solutions. For this reason we focus on finding sub-optimal solutions (approximate and heuristic approaches). Instead of searching the entire solution space for an optimal solution, we limited the number of solutions by eliminating the resources that do not satisfy the processing requirement conditions. Within this

smaller set of solutions, we want to select the solution that converges to the optimal one, considering our defined context of the problem. For this approach, a metric to evaluate solutions is needed but the efficiency is determine by several factors: availability of a function to evaluate a solution, the time required to evaluate a solution, the ability to judge the value of an optimal solution based on some metrics, a mechanism for intelligently pruning the solution space.

To be able to evaluate an algorithm, certain efficiency indictors or performance metrics have to be defined. For our evaluation, we considered the following:

- **Total Time to Completion (TTC)** – the total execution time for an application workflow. TTC is composed of different parts: time to create the application workflow and to divide it into set of tasks (stages), the scheduling time, the queueing time (or the time needed to start VMs if in the Cloud), and the actual runtime of the workflow. We can also add the time for results collection.
- **Total Cost of Completion (TCC)** – the total (public Cloud) cost for the entire application workflow.
- **Overall Throughput (OT)** – the time it takes for the HCE to run a batch of tasks.
- **Scheduling Algorithm Duration (SAD)** – a measure of how fast the tasks are map to resources. Considering a complex computing environment like HCE, this is an important indicator especially for the cases where the tasks have to be mapped as fasts as possible, avoiding bottleneck situations.
- **Makespan (M)** – the time it takes for all the resources to finish executing all the tasks. The executions of tasks on a resource are done in parallel with the other resources. This implies that the makespan is the total task execution time for the slowest resource (the one that finishes the last).
- **Load imbalance (LI)** – measures how balanced the load across the resources is. This indicator is defined as the difference between the time it takes for the last resource to finish execution (makespan) and the earliest time in which a resource finished its execution (executing the assigned tasks). The load is perfectly balanced if this indicator has a value of 0, implying that no resource is excessively overloaded and lowering the chances of resource failure due to overloading.
- **Success Rate (SR)** – the weighted percentage of the tasks that finished before their deadlines.

The task object contains a number of parameters that can be used to estimate the time needed for execution. This estimation is done also based on a number of resource attributes such as the CPU power, memory and the processing time needed to complete. Normally we have to take into account the fact that if another task starts executing on the same processing unit before others completes, an interruption mechanism is used to re-estimate the time needed for the remaining tasks to complete. For simplification reasons, the simulator was forced to run just one task per CPU at a time, by matching the memory required by the task with the memory of the CPU. In this way we can avoid the simulation of the interruption mechanism needed when a new task is assigned to a CPU, to re-compute the estimations times of the currently running tasks on that CPU.

The scheduler receives as input sets of tasks with dependencies and, using the provided scheduling algorithm and the underlying modeled configuration of heterogeneous resources (coming from different available computing infrastructures), it assigns each tasks to a corresponding processor for execution, according to the implemented scheduling policies. Using the simulation framework we have randomly generated the number of tasks and the number of resources, within a certain range. We have performed experiment also with a fixed configuration of tasks (application workflow) and resources (from different infrastructures).

The conclusion was that simulations tools can be very useful and very powerful, considering especially the complexity, the diversity, the dynamism and the costs of a HCE but the real conditions of such a computing environment are hard to map. There are a large number of errors that can appear in such an environment, from different reasons and having different effects. These errors play an important role in the efficiency and the successful termination of an execution. Our experiments were conducted for a small number of tasks and a limited number of processing resources. Further analysis on larger and more complex workflows might lead to different results and we plan to further evaluate these aspects in our future work. We also plan to consider more complex scheduling policies and more improvement scheduling criteria constraints in the future.

11.3. Conclusions

The aim of this chapter was to evaluate and validate the ENV2CE methodology and framework at component level. The overall evaluation of such a methodology is a complex, long and expensive process, considering the complexity, the diversity, the dynamism and the costs of a Hybrid Computing Environment.

The components proposed in the ENV2CE framework were individually evaluated and validated in a series of experiments, parts of different research projects and different functional environmental applications used by the Environmental Sciences community. Further analysis on larger and more complex applications workflows is also planned, in our future work, to further evaluate the overall methodology.

Conclusions

Chapter 12: Final Conclusions

Data Revolution is no longer a novelty as we live in a world in which data gets bigger and more complex, and data access is faster than ever. Governments, agencies, research groups and citizens at large are dealing with an unprecedented exponential increase in the volume and types of data that is strongly influenced by the fast advances in technology. According to the United Nations (UN, 2014 [228]), 90% of the data in the world has been created in the last two years. How prepared are we and how fast can we adapt to this new world, considering resources, capacities, knowledge and state of mind? Many questions still need answers and many issues need urgent solutions. An increasing quantity of data is never used because it is not freely available, not well documented, not following standards, or because not enough storage or computing resources are available to extract useful information from it for better decision-making. There is a clear need, at all scales, to discover, access, integrate, use and share spatial data from different sources to support well-informed decision-making and management of the growing national, regional, and global issues that are threatening our environment and the world at large. The vital importance, capabilities, benefits, and possibilities of using digital geographic information for sustainable development have been discussed and analyzed (Nebert, 2005 [161]) but appropriate specialized resources and services able to fully utilize these ever-growing sets of (big) data and information are not always available or suitable enough. Our capacity to handle large quantities of spatial information and the power to process it has increased exponentially in the last 30 years as a result of development in SDI, GIS technologies, Internet, and IT technologies.

In this context, the main goal of this research was: **To analyze and explore the ways in which Information Technology, and especially parallel and distributed high-performance systems, can improve the major challenges and needs the Environmental Sciences are facing in the process of extracting understandable and useful information from raw environmental data, leading in the end to a better informed decision making towards a sustainable development and a sustainable planet.**

To be able to achieve this goal, we have formulated in the introduction of this thesis some associated research questions and in the following we will give an overview of our findings during this research work:

1. **What are the current urgent *needs and challenges* of Environmental Sciences (focusing on environmental data and environmental applications)?**

During our research we have identified a series of issues and challenges that the Environmental Sciences community is facing. Most of them are related with understanding the complex Earth System and its components, their processes and interactions. Understanding such a system and predicting future behavior is a complex and challenging mission that requires managing and processing huge amounts of environmental data, using complex techniques and massive computing and storage resources. Solving these challenges is an essential condition to respond to the current global changes. The major identified challenges are:

- The dynamic and interdependent nature of environmental factors (both human and natural ones) and their impact;
- Integration of social, economic and environmental considerations;
- Decision and policies are based on existing knowledge, understanding, and observations, and often prove to fall short in terms of intended outcomes;
- The need of a holistic thinking – assessing a problem in the context of the larger system – of systems – within which it occurs;
- Understanding that complexity is a direct function of the problem statement, decision objectives, system understanding, and data availability;
- The importance of merging the knowledge domains into coherent and appropriately complex representations of the relevant system (Laniak et al., 2013 [130]);
- How to obtain knowledge from environmental data?
- How to apply this knowledge to explain, explore, and predict environmental-system responses to natural and human induced factors (stressors)?
- The diversity of stakeholders in Environmental Sciences;

- Discovery, accessing, processing, and preparation of data is particularly challenging due to a combination of cross-disciplinary, volume, and disparate sources, presenting data with varying formats and semantics;
- Sensor data gathering, quality assurance and dissemination must be optimized;
- Model developer have to deal with auxiliary observations of low and even unknown data quality;
- Need for appropriate protocols, standards, sharing permissions, and removal of language and scale barriers between various environmental fields;
- Interdependent elements: applications, science, technology and community;
- Interoperability between different systems (such as information technology components, software applications, organizations) has to be seen as the ability to exchange information accurately, effectively and consistently and to use that information;
- Environmental applications require large volumes of data and Big Data challenges have to be addressed effectively. Petabytes of geospatial data already exists and this amount grows every day at an increasing rate due to improving fidelity in data capture technologies (Mahdavi-Amiri et al., 2015 [144]);
- Need of modeling services that could intelligently integrate user-generated observations (from smart devices), measurements (from sensor stations) and referenced data (from institutional data repositories) in a generic way (Granell et al., 2016 [95]). Access to crowd-sourced environmental observations demonstrates the value of sharing small and localized observations, that when aggregated in Big Data repositories, can built a deeper and broader understanding of environmental phenomena (Granell et al., 2016 [95]);
- Data processing at varying aggregation levels;
- Information access is not equally divided over the globe, or within countries or communities – environmental knowledge and environmental information flows should not become monopolized or controlled by only a few people/sectors (Karpouzoglou et al., 2016 [120]);
- Need of openness: combination of transparency, cooperation and collaboration. This will allow open sharing of products of individual research and development efforts supporting a wider access and enabling innovation;

- Need for new directions to design and develop environmental software applications to make the best use of both geospatial and Future Internet technologies (holistic, flexible and scalable solutions that enable multi-disciplinary approach);
- Increasing demands for timely and contextually aware information delivery (Granell et al., 2016 [95]);
- The need to cross artificial boundaries between current sectors, to interconnect already existing systems, and to break technical as well as organizational barriers (Granell et al., 2016);
- There is no roadmap or single methodology for generating actionable knowledge, therefore a certain degree of flexibility, openness to alternatives and iterative learning is needed;
- High volume or high velocity collection of geospatial data is hard to fully exploit in many applications because of limited processing power (Lee and Kang, 2015 [133]);
- Need for fast and reliable tools and components to remotely access geo-referenced data;

2. What is the *state of the art* in High Performance Computing (HPC) landscape (cluster, Grid, Cloud, Hybrid Computing)?

The scientific applications of high-performance and distributed computing are widely spread and the most popular parallel and distributed platforms among scientists are Multicore processors, clusters, Grids and Clouds systems. Due to the integration of many-core technologies, these infrastructures are undergoing revolutionary changes, providing orders-of-magnitude speed improvements. Although each of these systems is straightforward to program, due to many existing platform specific tools, creating applications able to run on an optimum combination of such systems becomes difficult. The programming complexity of the applications, running on such heterogeneous and hierarchical platforms, has vastly increased also because of data distribution, need of scalability and software heterogeneity. These issues lead most of the times to the need of simultaneously using multiple platforms at the same time. Within this thesis we have identified the need of integrating different distributed computing infrastructures within a Hybrid Computing Environment and we have highlighted the challenges as well as the advantages of such a computing environment.

3. **What are the *lessons learned* based on a heuristic approach of integrating environmental applications with HPC?**

The execution of environmental applications usually requires large computational and storage resources due to the massive amount of data, high resolutions, and large geographical areas they are using. Different parallel and distributed infrastructures, such as cluster, Multicore, Grid, and Cloud satisfy mostly the necessary requirements for running such applications. Depending on application features, data model, and processing requirements, one of such infrastructures could be more appropriate and efficient, and could offer better performances than other ones. Porting applications and services to parallel and distributed infrastructures has been a concern in many research communities and it is still a very difficult task to accomplish since each platform requires particular details to take into account when integrating applications that have not been designed to run on parallel and distributed infrastructures in the first place. These applications have to be modified to have a particular structure or to use particular programming API for accessing the resources from each individual infrastructure, without knowing too much details about the running platform. We drew all these conclusions from a long set of experiments in which we have executed different environmental applications on different parallel and distributed infrastructures. Based on these conclusions we set our goal to define a generic conceptual model for environmental applications which will allow an easy integration with any computing infrastructure and finally to propose a general methodology for porting scientific applications on different parallel and distributed infrastructures. This is a complex and challenging task that has the potential to help solving many scientific issues in many research areas.

4. **What are the flexible solutions to *integrate* environmental applications and environmental data with HPC infrastructures?**

Computing resources available for the execution of an application may vary from different number of cores or computing nodes to different sizes of memory or storage nodes. The performance goal of an application may also vary, depending on different functions such as energy, throughput, cost, etc. Another variable item is the program's workload. All these elements of variability compose the execution environment of a program. For an application to

efficiently execute, it has to adapt to this variable execution environment in a flexible manner, which implies that the program should not be encoded with a single static parallelism configuration. Taking into account the growing need for computational speed, storage and scalability, that environmental applications demand, the users usually tend to use or to switch more than one execution infrastructure for obtaining the necessary resources. To be able to easily switch between these infrastructures we introduced an application conceptual model that hides the complexity of different types of environmental applications and that provides an easy and flexible way to map an environmental application to a computing infrastructure. Using this model, a user can easily describe the structure, the data flow as well as the execution flow (workflow) of the application. The aim of the proposed application conceptual model was to provide a platform-independent, robust, convenient and easy to use methodology, which would allow a user to execute an application on a heterogeneous computing environment.

5. **How can we solve the *interoperability* between Environmental Sciences and Computer Science?**

The interoperability between two systems in general, and between Environmental Sciences and Computer Science in our particular case, can be addressed using two general approaches:

- Standardization – defining common specification for interfaces, metadata and data models. This is a long, slow process, which requires commitment and adoption from the participating systems, high ICT expertise, and most of the times complex specifications.
- Mediation – adapting and harmonizing heterogeneous interfaces, metadata and data models. This is possible only if the harmonization and adoption is theoretically and practically feasible. This implies that distinct data models must be mapped to a higher-level conceptual model, able to reconcile the heterogeneous implementations.

In this thesis we proposed a solution to solve the interoperability between Environmental Sciences (environmental data and environmental applications) and Computer Science (a Hybrid Computing Environment composed from several parallel and distributed computing infrastructures) based on a mediation approach, through the introduction of an intermediate

"broker" layer (mediator), able to hide the complexity of the computing environment and to provide access to its functionalities and capabilities in an easy and flexible manner.

6. **Is there an efficiently way to take advantage of all the available *heterogeneous computing* infrastructures *simultaneously*?**

There is a large number of applications with interesting workload characteristics and resource requirements, which can take advantage of multiple computing infrastructures at a time, to reduce execution time, reduce cost (currency or resource allocation) or handle unexpected runtime situations (unexpected delays or unexpected failures) (Kim et al., 2009 [123]). In many realistic scientific research areas, domain experts are actually forced to concurrently use multiple computing infrastructures. All of these infrastructures are undergoing many changes due to the integration of core technologies, providing speed improvements but becoming also more heterogeneous, complex and hierarchical. The simultaneous execution on multiple computing infrastructures is a complex and challenging task as it involves not only the interoperability of scientific applications with different parallel and distributed infrastructures but also the interoperability and the coexistence of these infrastructures. The need to use multiple computing platforms for running an application is due to cases in which the reservation of a sufficient number of computing nodes in a single platform is impossible but also due to the distributed nature of the input data, heterogeneity of the software, ad-hoc availability of the resources etc. (Seinstra et al., 2011 [209]).

In this thesis, we have proposed the integration of different distributed computing infrastructures such as cluster, Grid and Cloud in a Hybrid Computing Environment. We have analyzed and presented the major challenges of such an environment and possible solutions.

7. **How can we solve the *interoperability between different HPC infrastructures*? What are the challenges and how this solution can be *applied* to Environmental Sciences?**

The computing is always changing to increase performance, changes which are reflected in microprocessor design and networks. Parallel and distributed high performance computing infrastructures, such as Multicore processors, clusters, Grids Clouds, became popular because they are able to offer the necessary resources needed for running large-scale scientific applications. These infrastructures can satisfy the need for processing power and storage

capacity, can improve accessibility to distributed storage and heterogeneous computing resources and can provide a reliable and secure environment.

Interoperable distributed infrastructures should allow applications to use them simultaneously. For this, commonly agreed protocols are required for information exchange and overall management. This is still hard to achieve and it will take a long time till all the distributed systems will adopt and implement these agreed protocols. The interoperability of different parallel and distributed infrastructures and the creation of a Hybrid Computing Environment was achieved in this thesis through a mediation approach, by introducing an intermediate "broker" layer (Mediator), able to manage the interoperation between the heterogeneous set of resources coming from different computing infrastructures and to allow users to access and use the functionalities of these infrastructures in a transparent manner.

In most of the cases, the functionality offered by distributed computing infrastructures can improve the execution time of a complex computation by taking advantage of the parallelization techniques. There are still cases in which the distributed infrastructures can have negative consequences, increasing the execution time of a process due to the overhead introduced by the specific management behind each infrastructure (job creation, submission and management, monitoring, result collection, etc.). These are usual the cases for simple processes which do not need such a high amount of resources. The analyze of the boundary above which distributed systems are beneficial in general in executing different processes but also the decision of which computing infrastructure is more appropriate than others for executing certain processes are two important challenges in the integration of environmental applications with a Hybrid Computing Environment. To achieve this integration we have introduced a new methodology and framework (ENV2CE) for efficiently porting and executing environmental applications on a Hybrid Computing Environment (HCE) and we have proposed a Mediation solution for the interoperability between environmental applications and different distributed computing infrastructures (HCE). The selection of the optimum set of heterogeneous resources needed for executing a specific application is a major challenge. In our proposed methodology, this set of heterogeneous resources, coming from different computing infrastructures, is chosen based on a computed complexity of the received requests, on the availability and accessibility of the computing infrastructures, on user preferences but also on the previous executions (history). The complexity of a request takes into account the application features, the parameters of the request

(amount of data to be retrieved or processed, processes to be performed), the user specifications/preferences, etc. At this level, we can define different complexity metrics and we can assign different weights to each input, considering application and user priorities.

9. How can we *evaluate and validate* the proposed solutions?

A set of defined metrics and experiments helped us evaluate the proposed methodology and framework and the conclusions are the following:

- The real case of HCE is hard to achieve and depends on a lot of factors;
- The evaluation of the ENV2CE methodology and framework was done at the component level;
- The evaluation and validation of the ENV2CE framework components were performed within the experiments done in different functional environmental applications and research projects;
- The simulation of the scheduling algorithm was realized within a developed simulation framework.

During our research we also discovered and assessed some different solutions and directions for the current environmental issues and we considered it is important to emphasize them:

- Open technologies approaches – remove institutional and geographical barriers associated with information flow (Karpouzoglou et al., 2016 [120]);
- Encourage widespread re-use of data for different purposes and data brokers and fusion of heterogeneous data;
- Bridge the gap between qualitative data (local observations, citizen generated data and perception data) and the quantitative data systems (typically used in decision support tools);
- Encourage multi-disciplinary approach in research and development activities towards developing effective geo-spatial data analytics;
- Data analytics research and development activities are needed to deal with the very high rates (speed) characterizing the new geospatial data flows;

- Provide algorithms and workflows capable to deal with huge quantities of data arriving at very high speed (Nedelcu, 2015 [163]);
- Visual representation of information is a fundamental component to interact indirectly with environmental observations and models. It helps display complex data in a more understandable way (visualization can increase the human capacity to process and retain complex information and reduce cognitive workloads);
- Interlink contemporary models from different disciplines to optimize the search for answers to increasingly complex questions (slow process due to the slow development and adoption of standards in the data and model sharing);
- The implementation of the SDGs towards a sustainable development will provide a tangible political "trigger" to foster and accelerate (Scott and Rajabifard, 2016 [208]):
    - the development and adoption of legal, technical, geospatial and statistical standards;
    - openness and exchange of data and metadata;
    - interoperability of data and information systems; and
    - integration of statistical and geospatial information (both management and exchange).
- Collaboration between organization is key to data management but also to data access and processing services;
- Provide a common agreed framework that allows easy and seamless integration of data from different sources, giving access to services that can collaborate/work together to provide new meaningful and understandable knowledge and information (Giuliani, 2011a [84]);
- Maximize the opportunities given by new technologies (such as HPC computing) towards a sustainable environment. Parallel architectures such as Grid, Cloud, Multicore, etc. seem to provide the necessary functionalities for solving most of the problems related to the geospatial data: handle complex computations through parallelism, both at data level and processing level, support data management and data security.

12.1. Concluding Remarks

- Timely access and easy integration of environmental data are essential.
- Transforming raw data into understandable information is an essential task that SDIs cannot fully satisfy. Therefore computational needs to process large data sets and efficient access to environmental data through OGC services will strongly influence the future success of SDIs.
- The usage of Parallel and Distributed infrastructures in the development and execution of environmental applications can offer promising opportunities through substantially lowering the development and execution costs.
- We need a good and easy to use methodology able to port the necessary applications and services to parallel and distributed architectures to obtain high performances.
- The Hybrid Computing Environment (HCE) proposed in this thesis should be seen (is) a proof of concept, demonstrating the feasibility and usability of heterogeneous distributed computing resources in the environmental context.
- The proposed HCE and the ENV2CE methodology do not offer a full replacement of the existing environmental information systems (which has a long history for offering strong support for geospatial data and processing). The intention is rather to complement the existing tools and services and to explore to the maximum the capabilities offered by different available distributed infrastructures.
- Choosing the optimum execution environment or the optimum combination of heterogeneous computing resources becomes an important issue for obtaining better performances.
- Porting applications and services to parallel and distributed infrastructures is a very difficult task to accomplish since each platform requires particular details to take into account when integrating applications that have not been designed to run on parallel and distributed platforms in the first place. These applications have to be modified to have a particular structure or to use particular programming API for accessing the resources from each individual architecture, without knowing too much details about the running platform.

12.2. Personal Contributions

The main contribution of this PhD thesis is the new methodology and framework (ENV2CE) designed for efficiently porting and executing environmental applications on a Hybrid Computing Environment.

Within this major contribution, we can identify several important original sub-contributions, highlighted also at the end of each thesis chapter:

- Identification of major environmental challenges related to environmental data, from an engineering point of view, based on literature review:
    - Metadata, Open Data, Linked Data, Data Interoperability;
    - Environmental Big Data – issues and challenges;
    - Challenges in transforming raw data into meaningful and understandable information.
- Heuristic approach on the technical details of SDI components:
    - Standards implementations;
    - Testing and working with different GIS and OGC services: WMS, WFS, WCS, WPS (PyWPS), TJS.
- Identification of major environmental challenges related to environmental applications, from an engineering point of view;
- Identification of general characteristics of environmental applications, mainly from hydrological and remote sensing fields.
- Development of a new environmental application (Environmental Data Acquisition and Processing – EDAP).
- Contributions to the development of different functional environmental applications mostly in the remote sensing and hydrological fields: gProcess, gSWAT, GreenLand, LiMES. The personal contributions are mainly oriented to:
    - Overall system architecture;
    - Processing components, using different geospatial technologies;
    - Scheduling and execution on different computing infrastructures;
    - Database design and implementation.

- Analyzing and working with two environmental applications (hydrological models SWAT and GFM - Continuum) in different projects.
- Execute different environmental applications on different distributed infrastructures:
    - SWAT calibration on gLite (Grid) vs. Multicore (UTCN);
    - SWAT calibration on Baobab cluster (UNIGE);
    - SWAT calibration on OpenStack Cloud (UNIGE/HEPIA);
    - SWAT calibration on OpenStack Cloud (UNIGE/SwissACC);
    - SWAT calibration on Windows Azure (UNIGE/SwissACC);
    - Global Flood Model on Baobab cluster (UNIGE);
    - OGC services Gridification on gLite (Grid) (UTCN);
    - Landsat 8 Data Acquisition on local servers (UNEP/Grid-Geneva).
- Observe the behavior of each infrastructure and understand what are their strengths and their weaknesses in porting environmental applications.
- Explore and analyze alternative solutions to connect different computing backend using GC3Pie tool, provided by the GC3 team (ETH – Zurich).
- Propose a conceptual model of environmental applications, based on theoretical knowledge and practical experience gained from different execution experiments (Chapter 7). The proposed model is a key component in a general methodology for porting these applications on different parallel and distributed infrastructures.
- Analyze and explore challenges and possible solutions in a Hybrid Computing Environment (HCE), composed from different distributed computing Infrastructures (DCIs): cluster, Grid, and Cloud.
- Introduce a new methodology and framework (ENV2CE) for efficiently porting and executing environmental applications on a Hybrid Computing Environment (HCE).
- Propose a Mediation solution for the interoperability between environmental applications and different distributed computing infrastructures (HCE).
- Execution of application tasks based on platform specific adaptors.
- Propose a new scheduling algorithm for a HCE, considering user specifications, application complexity, history, and optimization criteria.

12.3. Perspectives and Future Work

The future of Earth System as well as our future not just as scientists but also as human beings will depend on our ability to find and implement effective solutions to the current and future environmental problems. Regional-scale land use management, impacts of global climate change, valuation of ecosystem services, fate and transport of nanomaterial, life-cycle analysis, biodiversity, etc. are just few of the environmental problems that we are facing today. Our long-term goal should be to foster an increased understanding of the Earth systems well enough to predict future outcomes, impacts and consequences, to increase awareness and detection of unintended consequences of decisions and policies, and to reduce the perception of "black box" modeling of the Earth System. The rapid technological developments in information technology, telecommunications, sensors, networks etc. have greatly improved our ability to deal with complex environmental problems and challenges, especially regarding the huge amounts of environmental data and the environmental services, tools and methods needed to manage this data, but we still need to make large improvements in harnessing the full power and the potential of the technological progress to achieve solutions never possible before.

In this thesis we have proposed solutions to solve not only the interoperability between two different scientific research fields: i.e. Environmental Sciences and Computer Science, but also the interoperability between different parallel and distributed computing infrastructures, forming a Hybrid Computing Environment, using a Mediation approach. To this extent, we have introduced a new methodology and framework (ENV2CE) for efficiently porting and executing environmental applications on a Hybrid Computing Environment.

The research presented in this thesis offers a background for future research activities, among which we emphasize the followings:

- Integration of a more details analysis of the application complexity. Based on this, better task estimations can be done, improving therefore the scheduling algorithm and the overall performance.
- Improve the application conceptual model to integrate other application flows of particular cases.
- A more detailed experimental framework, focusing on better understanding the specific infrastructures' parameters, their interdependencies, correlation as well as a sensitive analysis to determine a better weight of their importance within a Hybrid Computing

Environment. It would also be interesting to extend the concepts and understanding gained in this work to other computational platforms and other metrics to improve.

- A better validation of the proposed methodology and framework through more intense experiments and different real use cases.

We believe that the ENV2CE methodology and framework is a step forward into a standardized way of accessing large heterogeneous storage and computing facilities for complex environmental applications. This achievement is an important element in better understanding the Earth System, predicting its future behavior and responding to the current global changes (climate, land use and demographic changes, loss of biodiversity, pollution, urbanization) that are threatening the natural environment and the society at large.

12.4. Results

- SCIEX Swiss Scholarship (18 months) – University of Geneva – Technical University of Cluj-Napoca – enviroPAD: Efficient Development and Execution of Environmental Applications on Parallel and Distributed Infrastructures;
- 14 ISI journal papers (4 as first author);
- 27 national and international conference proceedings papers (9 as first author);
- 1 book chapter;
- Presentations at different trainings and workshop events.
- Member in research projects, such as: enviroPAD, enviroGRIDS, MedioGRID, GiSHEO, SEE-GRID-SCI, mEducator;
- Guest Editor, "International Journal of Embedded Systems (IJES)", Inderscience, ISSN: 1741-1068 (http://www.inderscience.com/browse/index.php?journalID=45);
- Program Committee Co-Chair of the "5th International Workshop on Engineering Parallel and Multi-Core Systems ePaMuS – 2012 – The Multi-Core Workshop";
- Program Committee Member at the "Groupware and Online Campuses" track from the "3rd International Conference on Emerging Intelligent Data and Web Technologies EIDWT-2012";

- Program Committee Member at the "SeDiS – 2012 Workshop", held in conjunction with the "3rd International Conference on Emerging Intelligent Data and Web Technologies EIDWT - 2012" (http://sedis.hpc.pub.ro/).

Bibliography

[1] Abbaspour, K.C., Johnson, A., and van Genuchten, M.Th. (2004). Estimating uncertain flow and transport parameters using a sequential uncertainty fitting procedure. Vadose Zone J., Vol. 3, pp. 1340–1352.

[2] Abbaspour, K.C., Yang, J., Maximov, I., Siber, R., Bogner, K., Mieleitner, J., Zobrist, J., and Srinivasan, R. (2007). Spatially distributed modelling of hydrology and water quality in the pre-alpine/alpine Thur watershed using SWAT. Journal of Hydrology, Vol. 333, pp. 413-430.

[3] Abbaspour, K.C., Vejdani, M., Srinivasan, R. (2010). SWAT-CUP: A calibration and uncertainty analysis program for SWAT. Proceedings of the SWAT Conference 2010, Seoul, Korea.

[4] Ackoff, R. L. (1989). From data to wisdom. Journal of Applied Systems Analysis, Vol. 15, pp. 3-9.

[5] Adams, B., and Gahegan, M. (2014). Emerging Data Challenges for Next-generation spatial Data Infrastructure. Winter, S., Rizos, C.(eds.) Research@Locate 2014, Canberra, Australia, April 7-9, pp. 118-129, http://ceur-ws.org/Vol-1142/paper13.pdf.

[6] Agarwal, D., Karamati, S., Puri, S., and Prasad, S.K. (2014). Towards an MPI-Like Framework for the Azure Cloud Platform, in cluster, Cloud and Grid Computing (CCGrid), 14th IEEE/ACM International Symposium on, 26-29, May 2014, pp.176-185, https://dx.doi.org/10.1109/CCGrid.2014.100.

[7] Alexandrescu, A. (2012). Applications of Artificial Intelligence in Heterogeneous Computing Systems, PhD Dissertation, PhD School of the Faculty of Automatic Control and Computer Engineering, Gheorghe Asachi Technical University of Iasi, Iasi, Romania, 2012.

[8] Amalarethinam, G. D. I., and Josphin, M. A. (2015). Dynamic Task Scheduling Methods in Heterogeneous Systems: A Survey. International Journal of Computer Applications, Vol. 110(6), pp. 12-18.

[9] Amedro, B., Baude, F., Caromel, D., Delbe, C., Filali, I., Huet, F., Mathias, E., and Smirnov, O. (2010). Chapter: An efficient framework for running applications on

clusters, grids and clouds, Cloud Computing: Principles, Systems and Applications, series Computer Communications and Networks, pp. 163–178, Springer, 2010. ISBN: 978-0-387-47656-8.

[10] Analysis of Environmental Data, http://www.umass.edu/landeco/teaching/ecodata/schedule/environmental.data.pdf.

[11] Andrews, G. R. (1999). Foundations of Multithreaded, Parallel, and Distributed Programming. Addison-Wesley.

[12] Andrikopoulos, V., Tobias, B., Leymann, F., and Strauch, S. (2013). How to adapt applications for the Cloud environment. In: Computing, Springer, Vol. 95(6), pp. 493-535.

[13] Arnold, J.G., Srinivasan, R., Muttiah, R.S., and Williams, J.R. (1998). Large area hydrologic modeling and assessment, Part 1: Model Development. JAWRA Journal of the American Water Resources Association, Vol. 34 (1), pp. 73-89.

[14] Atzori, L., Iera, A., and Morabito, G. (2010). The Internet of things: a survey. Computer Networks, Vol. 54(15), pp. 2787-2805.

[15] Bacu V., Stefanut, T., Rodila, D., and Gorgan, D. (2009). Process Description Graph Composition by gProcess Platform, HiPerGRID - 3rd International Workshop on High Performance Grid Middleware, 28 May, Bucharest. Proceedings of CSCS-17 Conference, Vol.2., pp. 423-430, ISSN 2066-4451.

[16] Bacu V., Mihon D., Rodila D., Stefanut T., and Gorgan D. (2011a). Grid based architectural components for SWAT model calibration, in 2011 International Conference on High Performance Computing and Simulation (HPCS), Istanbul, Turkey, pp. 193-199, https://dx.doi.org/10.1109/HPCSim.2011.5999824.

[17] Bacu, V., Mihon, D., Rodila, D., Stefanut, T., Gorgan, D. (2011b). gSWAT Platform for Grid based Hydrological Model Calibration and Execution' in ISPDC 2011 - 10th International Symposium on Parallel and Distributed Computing, Cluj-Napoca, Romania.

[18] Bacu, V., Mihon, D., Stefanut, T., Rodila, D., Abbaspour, K., Rouholahnejad, E., and Gorgan, D. (2013). Calibration of SWAT Hydrological Models in a Distributed Environment Using the gSWAT Application, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 66–74, ISSN 2158-107X.

[19] Badger, M.L., Grance, T., Patt-Corner, R., Voas, J.M. (2012). Cloud Computing Synopsis and Recommendations – Recommendations of the National Institute of Standards and Technology. NIST Special Publication 800-146 (2012).

[20] Baker, M., Buyya, R., and Laforenza, D. (2002). Grids and grid technologies for wide-area distributed computing. Journal of Software-Practice & Experience, Vol. 32(15), pp. 1437-1466.

[21] Basaeed, E., Bhaskar, H., and Al-Mualla, M. (2012). Beyond Pan-sharpening: Pixel-level Fusion in Remote Sensing Applications, 2012 International Conference on Innovations in Information Technology (IIT), pp. 139 - 144, https://dx.doi.org/10.1109/INNOVATIONS.2012.6207718.

[22] Baumann, P. (2009). Array databases and raster data management, in T. Ozsu, L. Liu (eds.), Encyclopedia of Database Systems. Springer, 2009.

[23] Baumann, P. (2014). rasdaman: Array databases boost spatio-temporal analytics, in Computing for Geospatial Research and Application (COM.Geo), 2014 Fifth International Conference on, Aug 2014, pp. 54–54.

[24] Béjar, R., Latre, M. A., Nogueras-Iso, J., Muro-Medrano, P. R., and Zarazaga-Soria, F. J. (2009). Systems of Systems as a Conceptual Framework for Spatial Data Infrastructures." International Journal of Spatial Data Infrastructures Research, Vol. 4, pp. 201-217.

[25] Belgacem M. B. and Chopard B. (2015). A hybrid HPC/cloud distributed infrastructure: Coupling EC2 cloud resources with HPC clusters to run large tightly coupled multiscale applications. Future Generation Computer Systems, Vol 42(0), pp. 11-21.

[26] Bellinger, G., Castro, D., Mills, A. (2014). Data, Information, Knowledge, and Wisdom, The Way of Systems, http://www.systems-thinking.org/dikw/dikw.htm.

[27] Blagojevic, F., Nikolopoulos, D. S., Stamatakis, A., Antonopoulos, C. D. and Curtis-Maury, M. (2007). Runtime scheduling of dynamic parallelism on accelerator based multi-core systems. Parallel Computing, Vol. 33(10-11), pp. 700-719.

[28] Blanco, C.V., Huedo, E., Montero, R.S., Llorente, I.M. (2009). Dynamic Provision of Computing Resources from Grid Infrastructures and Cloud Providers, Workshops at the Grid and Pervasive Computing Conference, pp. 113-120.

[29] Bochenina, K. (2014). A comparative study of scheduling algorithms for the multiple deadline – constrained workflows in heterogeneous computing systems with time windows, Procedia computer science, Vol. 29, pp. 509–522.

[30] Bonomi, F., Milito, R., Zhu, J., Addepalli, S. (2012). Fog Computing and Its Role in the Internet of Things. Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, pp. 13–16.

[31] Borgdorff, J., Lorenz, E., Hoekstra, A.G., Falcone, J., and Chopard, B. (2011). A Principled Approach to Distributed Multiscale Computing, from Formalization to Execution. In e-Science Workshops (eScienceW), 2011 IEEE Seventh International Conference on, pp. 97–104.

[32] Borgdorff, J., Falcone, J., Lorenz, E., Bona-Casas, C., Chopard, B., and Hoekstra, A. G. (2013). Foundations of distributed multiscale computing: Formalization, specification, and analysis. Journal of Parallel and Distributed Computing, Vol. 73(4), pp. 465–483.

[33] Braun, R., Siegel, H., Beck, N., Boloni, L., Maheswaran, M., Reuther, A., Robertson, J., Theys, M., Yao, B., Hensgen, D. and Freund, R. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Parallel and Distributed Computing, Vol. 61(6), pp. 810-837.

[34] Bridges, M., Vachharajani, N., Zhang, Y., Jablin, T. and August D. (2007). Revisiting the sequential programming model for multi-core. In Proceedings of the 40th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), pp. 69-84.

[35] Calatrava, A., Molto, G., and Hernandez, V. (2011). Combining Grid and Cloud Resources for Hybrid Scientific Computing executions, in Proc. 2011 IEEE Third International Conference on Cloud Computing Technology and Science, pp. 494-501.

[36] Carrion, J. V., Molto, G., De Alfonso, C., Caballer, M. and Hernandez, V. (2010). A Generic Catalog and Repository Service for Virtual Machine Images, in 2nd International ICST Conference on Cloud Computing (CloudComp 2010), pp. 1-15.

[37] Caron, E., and Desprez, F. (2006). DIET: A Scalable Toolbox to Build Network Enabled Servers on the Grid, International Journal of High Performance Computing Applications, Vol. 20(3), pp. 335-352.

[38] Caron, E., Toch, L., Rouzaud-Cornabas, J. (2013). Comparaison de performance entre OpenStack et OpenNebula et les architectures multi-Cloud: Application à la cosmologie, Inria Research report n 8421, Project Team Avalon, Dec. 2013.

[39] Cavallaroa, G., Riedela, M., Benediktssona, J.A., Goetza, M., Runarssona, T., Jonassona, K., and Lippertb, T. (2014). Smart data analytics methods for remote sensing applications, In Geoscience and Remote Sensing Symposium (IGARSS), 2014 IEEE International, IEEE (2014), pp. 1405–1408.

[40] Chang, F. and Karamcheti, V. (2001). A framework for Automatic Adaptation of Tunable Distributed Application, Cluster Computing, Vol. 4(1), pp. 49-62.

[41] Chen, C., and Zhang, C.-Y. (2014). Data-intensive applications, challenges, techniques and technologies: a survey on big Data. Inf. Sci., Vol. 275, pp. 314 - 347.

[42] CIMA Resource Foundation – International Center on Environmental Monitoring, http://www.cimafoundation.org.

[43] Colceriu, V., Mihon, D., Minculescu, A., Bacu, V., Rodila, D., and Gorgan, D. (2013). Workflow Based Description and Distributed Processing of Satellite Images, in International Journal of Advanced Computer Science and Applications (IJACSA), ISSN 2158-107X, pp. 50–57.

[44] Coleman, D. J., Y. Georgiadou, and J. Labonte (2009). Volunteered geographic information: the nature and motivation of producers. International Journal of Spatial Data Infrastructures Research, Vol. 4(1), pp. 332–358.

[45] Comert, C. (2004). Web Services and National Data Infrastructure (NSDI), ISPRS Conference, 6p., http://cartesia.org/geodoc/isprs2004/comm4/papers/365.pdf.

[46] CometCloud Project. http://www.cometcloud.org/.

[47] Copernicus Big Data Workshop, March 2014, Bruxelles. http://www.copernicus.eu/library/detail/212.

[48] Costan, A. (2010). Autonomic Behavior of Large Scale Distributed Systems based on Monitoring Information, PhD Thesis, Politechnic University of Bucharest, Bucharest, Romania.

[49] CSW – OGC Catalogue Service Specification, http://www.opengeospatial.org/standards/cat.

[50] Curtis-Maury, M., Dzierwa, J., Antonopoulos, C. D., and Nikolopoulos, D. S. (2006). Online power-performance adaptation of multithreaded programs using hardware event-based prediction. In Proceedings of the 20th International Conference on Supercomputing (ICS), pp. 157-166.

[51] De Bono, A. (2016). Statistical Data Management for the UNEP Environmental Data Explorer (EDE), UNEP/GRID-Geneva, Draft Internal Document.

[52] Demchenko, Y., Zhao, Z., Grosso, P., Wibisono, A., De Laat, C. (2012). Addressing Big Data challenges for Scientific Data Infrastructure. IEEE 4th International Conference on Cloud Computing Technology and Science (CloudCom 2012). Taipei, Taiwan: IEEE Computing Society, based in California, USA, 2012, pp. 614-617.

[53] Di, L., Chen, A., Yang, W., and Zhao, P. (2003). The Integration of Grid Technology with OGC Web Services (OWS) in NWGISS for NASA EOS Data. Seattle, USA: Proceedings of the Eighth Global Grid Forum (GGF8), June 24-27.

[54] Di, L. (2004). The Development of Geospatially-enabled Grid Technology for Earth Science Applications, Proceedings of NASA Earth Science Technology Conference 2004. June 22-24, Palo Alto, CA, USA.

[55] Diaz-Montes, J., AbdelBaky, M., Zou, M., and Parashar, M. (2015). CometCloud: Enabling Software-Defined Federations for End-to-End Application Workflows.IEEE Internet Computing, Vol. 19(1), pp. 69-73.

[56] Digital Earth – ISDE – International Society for Digital Earth, http://www.digitalearth-isde.org/.

[57] DMTF, Cloud Management Standards, http://www.dmtf.org/standards/cloud.

[58] Dozier, J., and Gail, W. B. (2009). The emerging science of environmental applications. The Fourth Paradigm: Data Intensive Scientific Discovery, edited by Tolle, K., Tansley, S., and Hey, T., pp. 13 -19, Microsoft Research, Redmond, WA.

[59] EarthZine (2014) - Fostering Earth Pbservation & Global Awarness, The Group on Earth Observations Looks Toward a Second Decade of Data Sharing, http://earthzine.org/2014/01/06/the-group-on-earth-observations-looks-toward-a-second-decade-of-data-sharing/, January 2014.

[60] EGEE - Enabling Grid for e-Science project, http://www.eu-egee.org/.

[61] EMI – European Middleware Initiative project, http://www.eu-emi.eu/.

[62] ER-flow – Building a European Research Community through Interoperable Workflows and Data, EU FP7 project, http://www.erflow.eu/, 2013.

[63] ETSI TS 102 827: GRID; Grid Component Model (GCM); GCM Interoperability Deployment, European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France, 2008.

[64] ETSI TS 102 828: GRID; Grid Component Model (GCM); GCM Application Description, European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France, 2010.

[65] ETSI TS 102 829: GRID; Grid Component Model (GCM); GCM Fractal Architecture Description Language (ADL), European Telecommunications Standards Institute (ETSI), Sophia-Antipolis, France, 2009.

[66] European Strategy Forum on Research Infrastructures (ESFRI) e-IRG (e-Infrastructure Reflection Group) Report on Data management, November 2009, http://ec.europa.eu/research/infrastructures/pdf/esfri/publications/esfri_e_irg_report_data_management_december_2009_en.pdf.

[67] European Union, 2007 Directive of the European Parliament and of the Council establishing an Infrastructure for Spatial Information in the European Community (INSPIRE), Brussels.

[68] Evangelidis, K., Ntouros, K., Makridis, S., Papatheodorou, C. (2014). Geospatial services in the Cloud, in Computers and Geosciences, February 2014, Vol. 63, pp. 116-122.

[69] Evans, D. (2011). The Internet of Things. How the Next Evolution of the Internet is Changing Everything, CISCO White Paper.

[70] Ferrari, T. and Gaido, L. (2011). Resources and Services of the EGEE Production Infrastructure. Journal of Grid Computing, Vol. 9(2), pp. 119–133.

[71] Forti, A. (2006). DAG Scheduling for Grid Computing systems. PhD thesis, University of Udine - Italy.

[72] Fortuin, K. P. J., van Koppen, C. S. A., and Leemand, R. (2011). The Value of Conceptual Models in Cooping with Complexity and Interdisciplinarity in Environmental Sciences Education, Bioscience 61 (2011)10. – ISSN 0006-3568 – pp. 802-814.

[73] Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared, in: Grid Computing Environments Workshop, 2008, GCE-08, 2008, pp. 1–10, https://dx.doi.org/10.1109/GCE.2008.4738445.

[74] Fustes, D., Cantorna, D., Dafonte, C., Arcay, B., Iglesias, A., and Manteiga, M. (2014). A cloud-integrated web platform for marine monitoring using GIS and remote sensing. Application to oil spill detection through SAR images. Future Generation Computer Systems 34, pp. 155-160.

[75] Future Earth – research for global sustainability, http://www.futureearth.org/.

[76] Garlasu, D., Sandulescu, V., Halcu, I., Neculoiu, G., Grigoriu, O., Marinescu, M., Marinescu, V. (2013). A big data implementation based on Grid computing, Roedunet International Conference (RoEduNet) 2013 11th, Sinaia, Romania, https://dx.doi.org/10.1109/RoEduNet.2013.6511732.

[77] Gassman, P.W., Reyes, M.R., Green, C.H., and Arnold, J.G. (2007). The soil and water assessment tool: Historical development, applications, and future research directions. Transactions of the Asabe, Vol. 50(4), pp. 1211-1250.

[78] GC3Pie - https://code.google.com/p/gc3pie/.

[79] Gentzsch, W., Girou, D., Kennedy, A., Lederer, H., Reetz, J., Riedel, M., Schott, A., Vanni, A., Vazquez, M., and Wolfrat, J. (2011). DEISA-Distributed European Infrastructure for Supercomputing Applications. Journal of Grid Computing, Vol. 9(2), pp. 259–277.

[80] GEO, 2005. GEOSS: 10-Year Implementation Plan Reference Document. ESA Publications Division, http://www.earthobservations.org/documents/10-YearPlanReferenceDocument.pdf.

[81] GEO, 2007. Strategic Guidance for Current and Potential Contributors to GEOSS, Printed by JAXA on behalf of GEO Architecture and Data Committee, Available at: https://www.earthobservations.org/documents/portal/25_Strategic%20Guidance%20Document.pdf, Oct 2007.

[82] GeoTIFF data format, http://trac.osgeo.org/geotiff/.

[83] Gil, Y., Chan, M., Gomez, B., and Caron, B. (2014). EarthCube: Past, Present, and Future, EarthCube Project Report EC-2014-3.

[84] Giuliani, G. (2011a). Spatial Data Infrastructures for Environmental Sciences. PhD thesis, University of Geneva – Switzerland.

[85] Giuliani, G. (2011b). Bringing GEOSS Services into Practice, enviroGRIDS Workshop, Delft, Netherlands, 18th of April, 2011.

[86] Giuliani, G., Ray, N., Schwarzer, S., De Bono, A., Dao, H., Peduzzi, P., Beniston, M., Van Woerden, J., Witt, R., and Lehmann, A. (2011). Sharing environmental data through GEOSS. International Journal of Applied Geospatial Research, Vol. 2(1), pp. 1-17.

[87] Giuliani, G., Nativi, S., Lehmann, A., and Ray, N. (2012). WPS mediation: an approach to process geospatial data on different computing backends. Computers and Geosciences, Vol. 47, pp. 20-33.

[88] Giusti, L.C.D., Chichizola, F., Naiouf, M.R., Giusti, A.D., and Luque, E.(2010). Automatic Mapping Tasks to Cores – Evaluating AMTHA Algorithm in Multicore Architectures, International Journal of Computer Science Issues, Vol. 7, Issue 2, March 2010. Available online at: http://arxiv.org/abs/1004.3254.

[89] Goodchild, M.F., Guo, H., Annoni, A., Bian, L., de Bie, K., Campbell, F., Craglia, M., Ehlers, M., van Genderen, J., Jackson, D., Lewis, A.J., Pesaresi, M., Remetey-Fulopp, G., Simpson, R., Skidmore, A., Wang, C., Woodgate, P. (2012). Next-generation digital earth. Proc. Natl. Acad. Sci. Vol. 109 (28), pp. 11088-11094.

[90] Gore, A. (1998). The Digital Earth: Understanding our planet in the 21st Century, http://portal.opengeospatial.org/files/?artifact_id=6210.

[91] Gorgan, D., Bacu, V., Stefanut, T., Rodila, D., and Mihon, D. (2012). Earth Observation application development based on the Grid oriented ESIP satellite image processing platform, in Journal on Computer Standards & Interfaces, Published by Elsevier B.V., https://dx.doi.org/10.1016/j.csi.2011.02.002, ISSN: 0920-5489, pp: 541–548.

[92] Gorgan, D., Giuliani, G., Ray, N., Lehmann, A., Cau, P., Abbaspour, K., Charvat, K., Jonoski, A. (2013). Black Sea Catchment Observation System as a Portal for GEOSS Community. International Journal of Advanced Computer Science and Applications EnviroGRIDS Special Issue on Building a Regional Observation System in the Black Sea Catchment, pp. 9-18.

[93] gLite middleware, http://glite.cern.ch/.

[94] GML – OGC Geographic Markup Language Specification, http://www.opengeospatial.org/standards/kml.

[95] Granell, C., Havlik, D., Schade, S., Sabeur, Z., Delaney, C., Pielorz, J., Uslander, T., Mazzetti, P., Schleidt, K., Kobernus, M., Havlik, F., Bodsberg, N.R., Berre, A., and Mon, J.L. (2016). Future Internet technologies for environmental applications. Environmental Modelling and Software, Vol. 78, pp.1-15.

[96] GreenLand application, http://cgis.utcluj.ro/applications/greenland.

[97] Gropp, W., Lusk, E., and Skjellum, A. (1999). Using MPI: portable parallel programming with the message.passing interface, 2nd ed. Cambridge, MA: MIT Press.

[98] Grothe, M., and Brentjens, T. (2013). Joining Tabular and Geographic Data – Merits and Possibilities of the Table Joining Service, Report, Geonovum, http://www.geonovum.nl/sites/default/files/Report%20Geonovum-Table%20Joining%20Service%20v1.1.pdf.

[99] gSwat application, http://cgis.utcluj.ro/applications/gswat.

[100] Hadoop Documentation, http://hadoop.apache.org/.

[101] Hadoop Distributed File System (HDFS) - http://hortonworks.com/hadoop/hdfs/.

[102] Hanh, P.H., and Simonenko, V. (2015). Objective-oriented algorithm for job scheduling in parallel heterogeneous systems, Chapter: Job Scheduling Strategies for Parallel Processing, Volume 1291 of the series Lecture Notes in Computer Science, pp. 193-214.

[103] Hajibaba, M. and Gorgin, S. (2014). A Review on Modern Distributed Computing Paradigms: Cloud Computing, Jungle Computing and Fog Computing, Journal of Computing and Information Technology, Vol. 22(2), http://dx.doi.org/10.2498/cit.1002381.

[104] Harrison, K., Lavrijsen, W. T. L. P., Tull, C. E., Mato, P., Soroko, A., Tan, C. L., Brook, N. and Jones, R. W. L. (2003). GANGA: a user-Grid interface for Atlas and LHCb. In in Proceedings of Computing in High Energy and Nuclear Physics, La Jolla.

[105] Hierarchical Data Format (HDF) Group, http://www.hdfgroup.org/.

[106] Hierarchical Data Format for Earth Observing System (HDF-EOS), http://hdfeos.org/.

[107] Hoeing, A. (2010). Orchestrating Secure Workflows for Cloud and Grid Services, PhD Thesis, Electrical Engineering and Computer Science, Technical University of Berlin, Berlin, Germany.

[108] Ibis eScience software framework, http://www.cs.vu.nl/ibis/.

[109] IEEE Standards Association, P2301 - Guide for Cloud Portability and Interoperability Profiles (CPIP), http://standards.ieee.org/develop/project/2301.html.

[110] IEEE and the Open Group - The Open Group Base Specifications Issue 6 IEEE Std 1003.1, 2004 Edition. 2004.

[111] INSPIRE (2007), Infrastructure for Spatial Information in the European Community, http://inspire.ec.europa.eu/, May 2007.

[112] IPCC, 2014: Climate Change 2014: Synthesis Report. Contribution of Working Groups I, II and III to the Fifth Assessment Report of the Intergovernmental Panel on Climate Change [Core Writing Team, R.K. Pachauri and L.A. Meyer (eds.)]. IPCC, Geneva, Switzerland, 2014, http://www.ipcc.ch/pdf/assessment-report/ar5/syr/SYR_AR5_FINAL_full_wcover.pdf.

[113] ISO - International Organization for Standardization, http://www.iso.org/.

[114] ISO 19115 – 1:2014, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=53798.

[115] ISO/TS 19139: 2007, http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=32557.

[116] Jensen, J.R. (2000). Remote Sensing of the Environment: An Earth Resource Perspective. Prentice Hall, Jan. 2000.

[117] Kahanwal, B. and Singh, T.P. (2012). The Distributed Computing Paradigms: P2P, Grid, cluster, Cloud and Jungle, International Journal of Latest Research in Science and Technology, July-August 2012, Vol. 1(2), pp. 183-187.

[118] Karmas, A., Tzotsos, A., and Karantzalos, K. (2015). Scalable Geospatial Web Services through Efficient, Online and Near Real-time Processing of Earth Observation Data, in IEEE First International Conference on Big Data Computing Service and Application, pp. 194-201, https://dx.doi.org/10.1109/BigDataService.2015.49.

[119] Karoczkai, K., Kertesz, A., and Kacsuk, P. (2015). Brokering Solution for Science Gateways using Multiple Distributed Computing Infrastructures, 7th International Workshop on Science Gateways, Budapest, Hungary, pp. 28-33, https://dx.doi.org/10.1109/IWSG.2015.12.

[120] Karpouzoglou, T., Zulkafli, Z., Grainger, S., Dewulf, A., Buytaert, W., & Hannah, D. M. (2016). Environmental Virtual Observatories (EVOs): prospects for knowledge co-creation and resilience in the Information Age. Current Opinion in Environmental Sustainability, 18, pp. 40–48, http://dx.doi.org/10.1016/j.cosust.2015.07.015.

[121] Katz, D., Callaghan, S., Harkness, R., Jha, S., Kurowski, K., Manos, S., Pamidighantam, S., Pierce, M., Plale, B., Song, C., and Towns, J. (2010). Science on the TeraGrid. Computational Methods in Science and Technology, 05/2010. Special Issue 2010.

[122] Kaur, R. (2015). A Review of Computing Technologies: Distributed, Utility, cluster, Grid and Cloud Computing. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 5(2).

[123] Kim, H., el-Khamra, Y., Jha, S., and Parashar, M. (2009). An Autonomic Approach to Integrated HPC Grid and Cloud Usage, in 2009 Fifth IEEE International Conference on e-Science. IEEE, pp. 366–373.

[124] Kloh, H., Schulze, B., Mury, A., and Pinto, R. (2010). A scheduling model for workflows on grids and clouds, in Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science, no. December. ACM, pp. 3.

[125] Kramer, M., and Senner, I. (2015). A modular software architecture for processing of big geospatial data in the cloud, Computers and Graphics, Vol. 49, pp. 69-81, http://dx.doi.org/10.1016/j.cag.2015.02.005.

[126] Krauter, K., Buyya, R. and Maheswaran, M. (2002). A taxonomy and survey of grid resource management systems for distributed computing. Software, Practice, and Experience, Vol. 32(2), pp. 135–164.

[127] Kruger, A., and Kolbe, T.H. (2008). Mapping Spatial Data Infrastructures to a Grid Environment for optimized Processing of Large Amounts of Spatial Data, ISPRS Congress Beijing, Beijing, China, pp. 1559-1566.

[128] LaBarge, R. and McGuire, T. (2012). Cloud Penetration Testing, International Journal on Cloud Computing: Services & Architecture, Vol. 2(6), pp. 43.

[129] Landsat News 2014, NASA-USGS Landsat 8 Satellite Celebrates First Year of Success, http://www.nasa.gov/content/goddard/nasa-usgs-landsat-8-satellite-celebrates-first-year-of-success/#.UyIv1vl_spo.

[130] Laniak, G.F., Olchin, G., Goodall, J., Voinov, A., Hill, M., Glynn, P., Whelan, G., Geller, G., Quinn, N., Blind, M., Peckham, S., Reaney, S., Gaber, N., Kennedy, R., and Hughes, A. (2013). Integrated environmental modeling: A vision and roadmap for the future, Environmental Modeling and Software, Vol. 39, pp. 3-23.

[131] Lee, C. B., Schwartzman, Y., Hardy, J., and Snavely, A. (2005). Are user runtime estimates inherently inaccurate? Springer LNCS, Vol. 3277, pp. 253- 263.

[132] Lee, C., and Percivall, G. (2008). Standards-Based Computing Capabilities for Distributed Geospatial Applications, Computer, Vol. 41(11), pp. 50-57, Nov. 2008, https://dx.doi.org/10.1109/MC.2008.468.

[133] Lee, J., and Kang, M. (2015). Geospatial Big Data: Challenges and Opportunities. Elsevier, Big Data Research, 2015, pp. 74-81.

[134] Leiserson, C. E. (2009). The Cilk++ concurrency platform. In Proceedings of the 46th ACM/IEEE Design Automation Conference (DAC), pp. 522-527.

[135] LHC – CERN Large Hadron Collider, http://home.web.cern.ch/topics/large-hadron-collider.

[136] LiMES – Live Monitoring of Earth Surface (2015), http://limes.grid.unep.ch/.

[137] Lloyd, W., David, O., Lyon, J., Rojas, K., Ascough, J., Green, T., Carlson, J. (2012). The Cloud Services Innovation Platform – Enabling Service-Based Environmental Modelling Using IaaS Cloud Computing, In: Proc. iEMSs 2012 Int. Cong on Env.Modeling and Software, Germany, July 2012, pp. 8.

[138] Loekken, S., and Farres, J. (2014). ESA Earth Observation Big Data R&D Past, Present, & Future Activities, Copernicus Big Data Workshop, http://www.copernicus.eu/ .

[139] Lopes, R.V., and Menasce, D. (2015). A Taxonomy of Job Scheduling on Distributed Computing Systems, IEEE Transactions on Parallel and Distributed Systems, vol. 0 (0).

[140] Lu, Y., Nakicenovic, N., Visbeck, M., and Stevance, A.-S. (2015). Five priorities for the UN Sustainable Development Goals. Nature, Vol. 520 (7548), pp. 432-433.

[141] Maassen, J., Drost, N., Bal, H. E., and Seinstra, F. J. (2011). Towards jungle computing with Ibis/Constellation, Proc. of the 2011 workshop on Dynamic distributed data-intensive applications, programming abstractions, and systems (3DAPAS). New York, NY, USA: ACM, 2011, pp. 7-18.

[142] Madsen, H., Albeanu, G., Burtschy, B., Popentiu-Vladicescu, F. (2013). Reliability in the Utility Computing Era: Towards Reliable Fog Computing. International Conference on Systems, Signals and Image Processing (IWSSIP), Bucharest, Romania: IEEE, pp. 43–46.

[143] Maffioletti, S., and Murri, R. (2012). GC3Pie: A Python framework for high-throughput computing, Proceedings of the EGI Community Forum 2012/EMI Second Technical Conference (EGICF12-EMITC2). 26-30 March, 2012. Munich, Germany. Published online at, id. 143. Vol. 1. 2012.

[144] Mahdavi-Amiri, A., Alderson, T., and Samavati, F. (2015). A Survey of Digital Earth, Computers and Graphics, Vol 53, Part B, pp. 95-117.

[145] Maity, S., Bonthu, S. R., Sasmal, K., and Warrior, H. (2013). Role of Parallel Computing in Numerical Weather Forecasting Models. IJCA Special Issue on International Conference on Computing, Communication and Sensor Network CCSN2012 (4), pp. 22-27.

[146] Mateescu, G., Gentzsch, W. and Ribbens, C. J. (2011). Hybrid Computing—Where HPC meets grid and Cloud Computing. Future Generation Computer Systems, 27(5), pp. 440-453, http://dx.doi.org/10.1016/j.future.2010.11.003.

[147] Matott, L.S., Babendreier, J.E., and Purucker, S.T. (2009). Evaluating uncertainty in integrated environmental models: a review of concepts and tools. Water Resources Research 45, W06421. http://dx.doi.org/10.1029/2008WR007301.

[148] McKee, L. (2010). 18 Reasons for Open Publication of Geoscience Data, http://earthzine.org/2010/08/04/18-reasons-for-open-publication-of-geoscience-data/.

[149] McKee, L. (2015). OGC Information Technology Standards for Sustainable Development, OGC White Paper, https://portal.opengeospatial.org/files/?artifact_id=60920.

[150] Mihon, D., Bâcu, V., Stefanut, T., and Gorgan, D. (2010a). Tehnici de interactiune utilizator în aplicatiile de prelucrare a imaginilor satelitare – exemplificare pe baza aplicatiilor GreenView si GreenLand. RoCHI2010 - Conferinta Nationala de Interactiune Om-Calculator, 3-5 Sept. Bucuresti, ISSN 1843-4460, pp. 37-44.

[151] Mihon, D., Bacu, V., Gorgan, D., Mészáros, R., and Gelybó, G. (2010b). Practical considerations on the greenview application development and execution over see-grid. Earth Science Informatics, Vol. 3(4), pp. 247–258.

[152] Mohammadi, H., Rajabifard, A., and Williamson, I. (2008): Spatial Data Integrability and Interoperability in the Context of SDI. The European Information Society - Taking Geoinformation Science One Step Further. Amsterdam.

[153] Moscicki, J. T. (2003). DIANE - Distributed Analysis Environment for GRID-enabled Simulation and Analysis of Physics Data. Nuclear Science Symposium Conference Record, pp. 1617 – 1620.

[154] Moscicki, J.T., Brochu, F., Ebke, J., Egede, U., Elmsheuser, J., Harrison, K., Jones, R.W.L., Lee, H.C., Liko, D., Maier, A., Muraru, A., Patrick, G.N., Pajchel, K., Reece, W., Samset, B.H., Slater, M.W., Soroko, A., Tan, C.L., van der Ster, D.C. and Williams M. (2009). Ganga: A tool for computational-task management and easy access to grid resources. Computer Physics Communications, Vol. 180(11), pp. 2303 – 2316.

[155] Moss, R.H., Edmonds, J.A., Hibbard, K.A., Manning, M.R., Rose, S.K., van Vuuren, D.P., Carter, T.R., Emori, S., Kainuma, M., Kram, T., Meehl, G.A., Mitchell, J.F.B., Nakicenovic, N., Riahi, K., Smith, S.J., Stouffer, R.J., Thomson, A.M., Weyant, J.P., and Wilbanks, T.J. (2010). The next generation of scenarios for climate change research and assessment, Nature, Vol. 463(7282), pp. 747–756, https://dx.doi.org/10.1038/nature08823.

[156] Muller, M., Bernard, L., and Kadner, D. (2013). Moving code – Sharing geoprocessing logic on the Web. ISPRS Journal of Photogrammetry and Remote Sensing, March 2013, http://dx.doi.org/10.1016/j.isprsjprs.2013.02.011.

[157] Muresan, O., Pop, F., Gorgan, D., and Cristea, V. (2006). Satellite Image Processing Applications in MedioGRID. 5th International Symposium on Parallel and Distributed Computing, 6-9 July, pp. 253-262, https://dx.doi.org/10.1109/ISPDC.2006.42.

[158] Mylopoulos, J. (1992). Conceptual modeling and Telosí, in Conceptual modeling, databases, and CASE, chapter 2, P. Loucopoulos and R. Zicari, editors, pp. 49-68. Wiley.

[159] Nativi, S., Mazzetti, P., and Geller, G. (2013). Environmental model access and interoperability: the GEO Model Web initiative, Environmental Modelling and Software, Vol. 39, pp. 214-228, http://dx.doi.org/10.1016/j.envsoft.2012.03.007.

[160] Nativi, S., Mazzetti, P., Santoro, M., Papeschi, F., Craglia, M., and Ochiai, O. (2015). Big Data challenges in building the Global Earth Observation System of Systems. Environmental Modelling and Software, Vol. 68, pp. 1-26, http://dx.doi.org/10.1016/j.envsoft.2015.01.017.

[161] Nebert, D.D. (2005). Developing Spatial Data Infrastructure: The SDI Cookbook, Technical Working Group, Global Spatial Data Infrastructure.

[162] Nebert, D., Whiteside, A. and Vretanos, P. A. (2007). Open GIS Catalogue Services Specification. OpenGIS Publicy Available Standard, Open GIS Consortium Inc.

[163] Nedelcu, I. (2015). Current Challenges in Design, Development and Implementation of Geospatial Information Systems, 10th International Scientific Conference "Defence Resources Management in the 21st Century", pp. 196 – 204, Brasov, Romania, November 13th 2015.

[164] Network Common Data Form, http://www.unidata.ucar.edu/software/netcdf/docs/index.html.

[165] Nikolaou, C., Kyzirakos, K., Bereta, K., Dogani, K., Giannakopoulou, S., Smeros, P., Garbis, G., Koubarakis, M., Molina, D., Dumitru, O., Schwarz, G., and Datcu, M. (2014). Big, linked and open data: Applications in the German aerospace center, in The Semantic Web: ESWC 2014 Satellite Events, ser. Lecture Notes in Computer Science. Springer International Publishing, 2014, pp. 444–449.

[166] OGF – Open Grid Forum, OCCI – Open Cloud Computing Interface Working Group, http://forge.ogf.org/sf/projects/occi-wg.

[167] OGC – Open Geospatial Consortium, http://www.opengeospatial.org/.

[168] Ondieki, C. M., and Murimi, S. (2004). Applications of Geographic Information Systems, in Environmental Monitoring, edited by Prof. Hilary I. Inyang and Dr. John Daniels, in Encyclopedia of Life Support Systems (EOLSS), developed under the auspices of UNESCO, EOLSS. Publishers, Oxford, UK, http://www.eolss.net, 2004.

[169] Open Grid Forum – OGF, www.ogf.org.

[170] Ostermann, S., Prodan, R., and Fahringer, T. (2009). Extending Grids with cloud resource management for scientific computing, in 10th IEEE/ACM International Conference on Grid Computing, Oct. 13-15, 2009, pp. 42-49.

[171] Otepka, J., Ghuffar, S., Waldhauser, C., Hochreiter, R., and Pfeifer, N. (2013). Georeferenced point clouds: a survey of features and point cloud management. ISPRS International Journal Geo-Information 2013, Vol. 2(4), pp. 1038–65.

[172] Padberg, A., and Greve, K. (2009). Gridification of OGC Web Services: Challenges and Potential, GIS.Science, Vol. 14(3), pp. 77- 81.

[173] Padberg, A., and Kiehle, C. (2009). Towards a Grid-Enabled SDI: Matching the Paradigms of OGC Web Services and Grid Computing, International Journal of Spatial Data Infrastructures Research, Special Issue GSDI-11, pp. 174-178.

[174] Palankar, M. R., Iamnitchi, A., Ripeanu, M., and Garfinkel, S. (2008). Amazon S3 for science Grids: a viable solution? In International Workshop on Data-aware Distributed Computing (DADC'08) in conjunction with HPDC 2008, pp. 55-64, New York, NY, USA, 2008. ACM.

[175] Parekh, V. (2005). Applying Ontologies and Semantic Web technologies to Environmental Sciences and Engineering, Master of Science, University of Maryland, Baltimore County, 2005.

[176] Piegorsh, W. W., and Bailer, A. J. (2005). Analyzing Environmental Data. John Wiley and Sons, Mar 11, 2005, Mathematics, ISBN:0-470-84836-7.

[177] Pinedo, M.L. (2008). Scheduling: Theory, Algorithms, and Systems, 3rd edn. Springer.

[178] Pop, F. (2008), Optimization of Decentralized Scheduling Strategies in Grid Environments, PhD Thesis, Computer Science and Engineering Department, University Polytechnic of Bucharest, Romania.

[179] Rajabifard, A., and Williamson, I.P. (2001). Spatial Data Infrastructures: Concept, SDI Hierarchy and Future directions. Geomatics'80, Tehran, Iran.

[180] Rajabifard, A., and Williamson, I.P. (2004). SDI Development and Capacity Building, GSDI-7 Conference on Spatial Data Infrastructure for a Sustainable Future, Bangalore, India, 12p.

[181] Raman, A. (2011). A System for Flexible Parallel Execution. PhD thesis, Department of Computer Science, Princeton University, Princeton, New Jersey, United States, December 2011.

[182] Raman, A., Kim, H., Oh, T., Lee, J. W., and August, D. I. (2011). Parallelism orchestration using DoPE: the degree of parallelism executive. In Proceedings of the

32nd ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI).

[183] RAMSAR Convention – the Convention on Wetlands of International Importance, http://www.ramsar.org/.

[184] RDF – Resource Description Framework, https://www.w3.org/RDF/.

[185] Reed, C., Buehler, K., and McKee, L. (2015). OGC Consensus: How Successful Standards Are Made, ISPRS International Journal of Geo-Information, 2015, 4, 1693-1706, https://dx.doi.org/10.3390/ijgi4031693.

[186] Reid, J., Waites, W., and Butchart, B. (2012). An Infrastructure for Publishing Geospatial Metadata as Open Linked Metadata. Multidisciplinary Research on Geographical Information in Europe and Beyond, Proceedings of the AGILE'2012 International Conference on Geographic Information Science, Avignon, April, 24-27, 2012.

[187] Reinders, J. (2007). Intel Threading Building Blocks. O'Reilly & Associates, Inc., 2007.

[188] Reynoso, L., Grosclaude, E., Sanchez, L., and Alvarez, M. (2014). Towards a social interaction-based cognition model: An analysis of Spatial Data Infrastructure, IEEE 13th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC), pp. 459-467, https://dx.doi.org/10.1109/ICCI-CC.2014.6921499.

[189] RightScale, http://www.rightscale.com/.

[190] Rings, T., Caryer, G., Gallop, J., Grabowski, J., Kovacikova, T., Schulz, S. and Stokes-Rees, I. (2009). Grid and Cloud Computing: Opportunities for Integration with the Next Generation Network, Journal of Grid Computing: Special Issue on Grid Interoperability, JOGC, Vol. 7(3), pp. 375 – 393.

[191] Rings, T., Grabowski, J., and Schulz, S. (2011). A Testing Framework for Assessing Grid and Cloud Infrastructure Interoperability, International Journal On Advances in Systems and Measurements (SysMea11v4n12), Vol. 3, (1&2), pp. 95–108.

[192] Rings, T. and Grabowski, J. (2012). Pragmatic integration of cloud and grid computing infrastructures. In Proceedings of the 2012 IEEE fifth International Conference on Cloud Computing CLOUD '2012.

[193] Rodero-Merino, L., Vaquero, L. M., Gil, V., Galan, F., Fontan, J., Montero, R. S., and Llorente, I. M. (2010). From infrastructure delivery to service management in clouds,

Future Gener. Comput. Syst., Vol. 26(8), pp. 1226-1240, http://dx.doi.org/10.1016/j.future.2010.02.013, (RESERVOIR Project).

[194] Rodila, D., Bacu, V., and Gorgan, D. (2009). Integration of Satellite Image Operators as Workflows in the gProcess Application, in Proceedings of ICCP2009 - IEEE 5th International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 27-29, 2009, ISBN: 978-1-4244-5007-7, pp. 355-358, http://dx.doi.org/10.1109/ICCP.2009.5284737.

[195] Rodila, D., and Gorgan, D. (2010). Integration of Spatial Data Infrastructures with Grid Environment, 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, pp. 269-277.

[196] Rodila, D., Gorgan, D., and Bacu, V. (2010). The Interoperability between OGC Services and Grid Environment in enviroGRIDS project. MiDiS-2010, The First International Workshop on Middleware for Large Scale Distributed Systems, November 4-6, 2010, Fukuoka, Japan, pp. 387-392.

[197] Rodila, D., and Gorgan, D. (2011). A Mediation Approach in Geospatial Web Services Gridification, 12th IEEE International Conference on Intelligent Computer Communication and Processing, ICCP2011, pp. 541-548.

[198] Rodila, D., and Gorgan, D. (2012). Geospatial and Grid Interoperability through OGC Services Gridification, in International Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS), Vol. 5/6, December 2012, https://dx.doi.org/10.1109/JSTARS.2012.2217115, ISSN: 1939-1404, pp. 1650 - 1658.

[199] Rodila, D., Bacu, V., and Gorgan, D. (2012). Comparative Parallel Execution of SWAT Hydrological Model on Multicore and Grid Architectures, in International Journal of Web and Grid Services (IJWGS), Vol. 8/3, September 2012, pp. 304 – 320, https://dx.doi.org/10.1504/IJWGS.2012.049172.

[200] Rouholahnejad, E., Abbaspour, K.C., Vejdani, M., Srinivasan, R., and Lehmann, A. (2011). Parallelizing SWAT Calibration in Windows using SUFI2 Program, in Environmental Modelling and Software, Vol. 31, pp. 28–36.

[201] Rouholahnejad, E., Abbaspour, K. C., Srinivasan, R., Bacu, V., and Lehmann, A. (2014). Water resources of the Black Sea Basin at high spatial and temporal resolution, Water

Resources Research, Vol. 50(7), pp. 5866–5885, https://dx.doi.org/10.1002/2013WR014132.

[202] Roy, D.P., Wulder, M.A., Loveland, T.R., Woodcock, C.E., Allen, R.G., Anderson, M.C., Helder, D., Irons, J.R., Johnson, D.M., Kennedy, R., Scambos, T.A., Schaaf, C.B., Schott, J.R., Sheng, Y., Vermote, E.F., Belward, A.S., Bindschadler, R., Cohen, W.B., Gao, F., Hipple, J.D., Hostert, P., Huntington, J., Justice, C.O., Kilic, A., Kovalskyy, V., Lee, Z.P., Lymburner, L., Masek, J.G., McCorkel, J., Shuai, Y., Trezza, R., Vogelmann, J., Wynne, R.H., and Zhu, Z. (2014). Landsat-8: Science and product vision for terrestrial global change research, in Remote Sensing of Environment, Vol. 145, pp. 154-172.

[203] Ruth, P., McGachey, P., and Xu, D. (2005). VioCluster: Virtualization for dynamic computational domain. In IEEE International on Cluster Computing (Cluster 2005), pages 1-10, Burlington, USA, September 2005.IEEE.

[204] Ryan, B. (2016). The Benefits from Open Data are Immense, Geospatial World, pp.72-73 http://geospatialworld.net/uploads/magazine/January-2015-Geospatial-World-Magazine/files/72.html, January 2016.

[205] Sadashiv, N., and Kumar, D. (2011). Cluster, Grid and Cloud Computing: A Detailed Comparison, in Proceedings of the 6th International Conference on Computer Science & Education (ICCSE). IEEE.

[206] Santos, N. and Gonçalves, G. (2014). Remote sensing applications based on satellite open data (Landsat8 and Sentinel-2), Conferência Nacional de Geodecisão. Instituto Politécnico de Setúbal (ESTB/IPS), 15-16 May, 2014.

[207] Savenije, H.H.G. (2009). The art of hydrology, in Hydrology and Earth System Sciences, Vol. 13, pp. 157–161.

[208] Scott, G., and Rajabifard, A. (2016). Integrating Geospatial Information into Sustainable Development Goals, Geospatial World, January 2016, pp. 68 – 71.

[209] Seinstra, F. J. et al. (2011). Jungle computing: Distributed supercomputing beyond clusters, grids, and clouds. In M. Cafaro and G. Aloisio, editors, Grids, Clouds and Virtualization, Computer Communications and Networks, pp. 167-197. Springer.

[210] Seltenrich, N. (2014). Remote-Sensing Applications for Environmental Health Research, Environmental Health Perspective, Vol. 122(10), A268-A296, https://dx.doi.org/10.1289/ehp.122-A268.

[211] SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs (SHIWA) EU FP7 project, http://www.shiwa-workflow.eu/, 2012.

[212] Silberstein, M., Sharov, A., Geiger, D., and Schuster, A. (2009). Gridbot: execution of bags of tasks in multiple grids. In Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis, pp. 1-12, http://dx.doi.org/10.1145/1654059.1654071.

[213] Silvestro, F., Gabellani, S., Delogu, F., Rudari, R., and Boni, G. (2013). Exploiting remote sensing land surface temperature in distributed hydrological modelling: the example of the Continuum model, Hydrol. Earth Syst. Sci., 17, pp. 39-62, https://dx.doi.org/10.5194/hess-17-39-2013.

[214] Silvestro, F., Gabellani, S., Delogu, F., Rudari, R., Laiolo, P., and Boni, G. (2015). Uncertainty reduction and parameter estimation of a distributed hydrological model with ground and remote-sensing data, Hydrol. Earth Syst. Sci., 19, pp. 1727-1751, https://dx.doi.org/10.5194/hess-19-1727-2015.

[215] Silvestro, F., Campo, L., Rudari, R., De Angeli, S., D'Andrea, M., Rodila, D., and Gabellani, S. (2016). Impacts of EC-Earth Global Climate Model RCP4.5 climate change scenario on maximum daily streamflow quantiles at global scale, Journal of Climate, (submitted article).

[216] Snir, M., Otto, S., Huss-Lederman, S., Walker, D., and Dongarra, J. (1998). Eds., MP1-The Complete Reference, 2nd ed. Cambridge, MA, US MIT Press Cambridge, ISBN:0262692155.

[217] Sokolowski, J.A., and Banks, C.M. (2010). Modeling and Simulations Fundamental.

[218] Suleman, M. A., Qureshi, M. K., Khubaib, and Patt, Y. N. (2010). Feedback-directed pipeline parallelism. In Proceedings of the 19th International Conference on Parallel Architecture and Compilation Techniques (PACT), pp. 147 - 156.

[219] Sun, J., Yang, J., Zhang, C., Yun, W., and Qu, J. (2013). Automatic remotely sensed image classification in a Grid environment based on the maximum likelihood method, in Mathematical and Computer Modelling, Vol. 58, Issue 3-4, pp. 573 – 581, August 2013, https://dx.doi.org/10.1016/j.mcm.2011.10.063.

[220] SWAT – Soil and Water Assessment Tool, http://swat.tamu.edu/.

[221] SWAT Theoretical Documentation, Version 2009, http://twri.tamu.edu/reports/2011/tr406.pdf.

[222] Tak, B., Urgaonkar, B., and Sivasubramaniam, A. (2011). To Move or not to Move: The Economics of Cloud Computing. In: Third USENIX Workshop on Hot Topics in Cloud Computing (HOTCLOUD 2011).

[223] Tanenbaum, A. S., Steen, M. V. (2006). Distributed systems: Principles and Paradigms, (2nd Edition). Pearson Prentice Hall.

[224] Taylor, S.J.E., Popescu, G.V., Pullen, J.M., Turner, S.J. (2004). Panel on distributed simulation and the grid. Proceeding of 8th IEEE International Symposium on Distributed Simulation and Real-Time Applications (DS-RT 2004), Budapest, Hungary October 21-23, 2004. IEEE Comp Soc Tech Comm Parallel Process; IEEE Comp Soc Tech Comm Simulat; IEEE Comp Soc Tech Comm Comp Architecture; pp. 144-149, https://dx.doi.org/10.1109/DS-RT.2004.14.

[225] TJS – OGC Table Joining Service, http://www.opengeospatial.org/standards/tjs.

[226] Tumwizere, R.P., and Jeong, K. (2012). A survey on Computing Technology Application in Remote Sensing, 2012 8th International Conference on Computing and Networking Technology (ICCNT), 27-29 August, 2012, pp. 77-80.

[227] Tusa, F., Celesti, A., Paone, M., Villari, M. and Puliafito A. (2011). How clever-based clouds conceive horizontal and vertical federations, in Proceedings of the 2011 IEEE Symposium on Computers and Communications, ser. ISCC '11. Washington, DC, USA: IEEE Computer Society, pp. 167-172, http://dx.doi.org/10.1109/ISCC.2011.5984011, (CLEVER Project).

[228] UN (2014), A World that counts, mobilizing the Data Revolution for Sustainable Development, the Independent Expert Advisory Group on a Data Revolution for Sustainable Development, United Nations, 32p. http://www.undatarevolution.org/wp-content/uploads/2014/12/A-World-That-Counts2.pdf.

[229] United Nations – Sustainable Development Goals (SDGs) - https://sustainabledevelopment.un.org/sdgs.

[230] Vitolo, C., Elkhatib, Y., Reusser, D., and Macleod, C.J.A. (2015). Web technologies for environmental big Data. Environ. Model. Softw. Vol. 63, pp. 185-198.

[231] Viviroli, D., Zappa, M., Gurtz, J., Weingartner, R. (2009). An introduction to the hydrological modelling system PREVAH and its pre- and post-processing-tools. Environmental Modelling & Software 24 (10), pp. 1209-1222, https://dx.doi.org/10.1016/j.envsoft.2009.04.001.

[232] Vrugt, J.A., Nuallian, B. O., Robinson, B. A., Bouten, W., Dekker, S. C., and Sloot, P. M. A. (2006). Application of parallel computing to stochastic parameter estimation in environmental models. Computers and Geosciences, 32(8), pp. 1139 - 1155, https://dx.doi.org/10.1016/j.cageo.2005.10.015.

[233] W3C – World Wide Web Consortium, https://www.w3.org/.

[234] Wand, Y., Weber, R. (2002). Research Commentary: Information Systems and Conceptual Modeling – A Research Agenda, in Information Systems Research, Vol. 13(4), pp. 363—376.

[235] Wang, Z., and O'Boyle, M. F. (2009). Mapping parallelism to multi-cores: A machine learning based approach. In Proceedings of the 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP), pp. 75-84.

[236] Wang, P., Wang, J., Chen, Y., and Ni, G. (2013). Rapid processing of remote sensing images based on cloud computing. Future Generation Computer Systems 29 (8), pp. 1963-1968.

[237] WATERS – Watershed Assessment, Tracking and Environmental Results System, https://www.epa.gov/waterdata/waters-watershed-assessment-tracking-environmental-results-system.

[238] Weissman, J.B. (2002). Predicting the Cost and Benefit of Adapting Data Parallel Applications in Clusters. Journal of Parallel and Distributed Computing, volume 62, issue 8, August 2002, pp. 1248-1271.

[239] Werder, S., and Kruger, A. (2009). Benefit from Parallelization in Grid-Computing, http://ifgi.uni-muenster.de/archives/agile/submissions/AGILE-GridWorkshop-Werder.pdf.

[240] WCS – OGC Web Coverage Service Specification, http://www.opengeospatial.org/standards/wcs.

[241] WFS – OGC Web Feature Service Specification, http://www.opengeospatial.org/standards/wfs.

[242] WMS – OGC Web Map Service Specification, http://www.opengeospatial.org/standards/wms.

[243] WPS – OGC Web Processing Service Specification, http://www.opengeospatial.org/standards/wps.

[244] Wulder, M.A., Masek, J.G., Cohen, W.B., Loveland, T.R., and Woodcock, C.E. (2012). Opening the archive: how free data has enabled the science and monitoring promise of Landsat: Remote Sensing of Environment, Vol. 122, pp. 2-10, http://dx.doi.org/10.1016/j.rse.2012.01.010.

[245] Xhafa, F., Paniagua, C., Barolli, L. and Caball, S. (2010). A parallel grid-based implementation for real-time processing of event log data of collaborative applications, in International Journal of Web and Grid Services, Vol. 6, No.2, pp.124-140.

[246] Xiao, L.B., Zhong, Er.S., and Liu, J.Y. (2001). A discussion on basic problems of 3D GIS. Journal of Image and Graphics 2001:6(A) (9): 842-848.

[247] Yamini, L., LathaSelvi, G., and Mukherjee, Saswati (2011). Efficient Metascheduling in a Cloud Extended Grid Environment, in IEEEInternational Conference on Recent Trends in Information Technology, pp. 667-672.

[248] Yang, C., Goodchild, M., Huang, Q., Nebert, D., Raskin, R., Xu, Y., Bambacus, M., and Fay, D. (2011). Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? International Journal of Digital Earth, Vol. 4 (4), pp. 305-329.

[249] Yang, C., Xu, Y., and Nebert, D. (2013). Redefining the possibility of Digital Earth and geosciences with spatial cloud computing. International Journal of Digital Earth, Vol. 6(4), pp. 297-312.

[250] Yang, C., and Huang Q. (2013). Spatial Cloud Computing: A Practical Approach, CRC Press, ISBN 9781466593169.

Annexes

Published Articles

Journal Papers

1. **Rodila, D.,** Gorgan, D., Ray, N., and Lehmann, A. (2016). ENV2CE: Environmental Application Conceptualization and Execution on a Hybrid Computing Environment - Framework and Methodology Proposal, (to be submitted).
2. **Rodila**, **D.,** Ray, N., and Gorgan, D. (2015). Conceptual Model for Environmental Science Applications on Parallel and Distributed Infrastructures, Environmental System Research, Vol. 4/23, 2015, https://dx.doi.org/10.1186/s40068-015-0050-1.
3. **Rodila**, **D.,** Bacu, V., and Gorgan, D. (2012). Comparative Parallel Execution of SWAT Hydrological Model on Multicore and Grid Architectures, in International Journal of Web and Grid Services (IJWGS), Vol. 8/3, September 2012, pp. 304 – 320, https://dx.doi.org/10.1504/IJWGS.2012.049172.
4. **Rodila**, **D.,** and Gorgan, D. (2012). Geospatial and Grid Interoperability through OGC Services Gridification, in International Journal of Selected Topics in Applied Earth Observations and Remote Sensing (JSTARS), Vol. 5/6, December 2012, pp. 1650 – 1658, ISSN: 1939-1404, https://dx.doi.org/10.1109/JSTARS.2012.2217115.
5. Giuliani, G., Dao, H., De Bono, A., Chatenoux, B., Allenbach, K., De Laborie, P., **Rodila, D.,** Alexandris, N., and Peduzzi, P. (2016). Live Monitoring of Earth Surface (LiMES): a framework for monitoring environmental changes from Earth Observations, (to be submitted).
6. Silvestro, F., Campo, L., Rudari, R., De Angeli, S., D'Andrea, M., **Rodila**, **D.,** and Gabellani, S. (2016). Impacts of EC-Earth Global Climate Model RCP4.5 climate change scenario on maximum daily streamflow quantiles at global scale, Journal of Climate (submitted article).
7. Mihon, D., Colceriu, V., Bacu, V., Allenbach, K., **Rodila**, **D.,** Giuliani, G., and Gorgan, D. (2013). OGC Compliant Services for Remote Sensing Processing over the Grid Infrastructure, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 32–40, ISSN 2158-107X.

8. Colceriu, V., Mihon, D., Minculescu, A., Bacu, V., **Rodila**, **D.,** and Gorgan, D. (2013). Workflow Based Description and Distributed Processing of Satellite Images, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 50–57, ISSN 2158-107X.
9. Bacu, V., Mihon, D., Stefanut, T., **Rodila**, **D.,** Abbaspour, K., Rouholahnejad, E., and Gorgan, D. (2013). Calibration of SWAT Hydrological Models in a Distributed Environment Using the gSWAT Application, in International Journal of Advanced Computer Science and Applications (IJACSA), pp. 66–74, ISSN 2158-107X.
10. Gorgan, D., Bacu, V., Stefanut, T., **Rodila**, **D.,** and Mihon, D. (2012). Earth Observation application development based on the Grid oriented ESIP satellite image processing platform, in Journal on Computer Standards & Interfaces, Published by Elsevier B.V., pp. 541–548, ISSN: 0920-5489, https://dx.doi.org/10.1016/j.csi.2011.02.002.
11. Gorgan, D., Bacu, V., Mihon, D., Stefanut, T., **Rodila**, **D.,** Cau, P., Abbaspour, K., Giuliani, G., Ray, N., and Lehmann, A. (2012). Software platform interoperability throughout enviroGRIDS portal, in International Journal of Selected Topics in Applied Earth Observations and Remote Sensing - JSTARS, Vol. PP/99, pp. 1-11.
12. Gorgan, D., Bacu, V., Mihon, D., **Rodila**, **D.,** Abbaspour, K., and Rouholahnejad, E. (2012). Grid based calibration of SWAT hydrological models, in Journal of Nat. Hazards Earth Syst. Sci., Vol. 12/7, pp. 2411-2423, https://dx.doi.org/10.5194/nhess-12-2411-2012.
13. Mihon, D., Bacu, V., **Rodila**, **D.,** Stefanut, T., Abbaspour, K., Rouholahnejad, E., and Gorgan, D. (2012). Grid Based Hydrologic Model Calibration and Execution, Chapter in the book: Advanced in Intelligent Control Systems and Computer Science, Dumitrache I. (Ed.), Springer-Verlag, Vol. 187, pp. 279-293, ISBN 978-3-642-32548-9, https://dx.doi.org/10.1007/978-3-642-32548-9_20.
14. Gorgan, D., Bacu, V., **Rodila**, **D.,** Pop, F., and Petcu, D. (2010). Experiments on ESIP - Environment Oriented Satellite Data Processing Platform, in Earth Science Informatics Journal, Springer, Dec. 2010, Vol.3/4, pp. 297-308, ISSN:1865-0473, https://dx.doi.org/10.1007/s12145-010-0065-0, http://www.springerlink.com/content/v463w6872266j282/.

Conference Papers

1. **Rodila**, **D.,** and Gorgan, D. (2012). Mapping Geospatial Applications onto Parallel and Distributed Environments, in ePaMuS 2012 – 5th International Workshop on Engineering Parallel and Multi-Core Systems – The Multi-Core Workshop, Palermo, Italy, 4-6 July, 2012, pp. 443 – 448, http://dx.doi.org/10.1109/CISIS.2012.152.
2. **Rodila, D.,** Bacu, V., and Gorgan, D. (2012). Geospatial Applications on Different Parallel and Distributed Systems in enviroGRIDS Project, in European Geosciences Union - General Assembly EGU 2012, Vienna, Austria, April 22-27, 2012, (abstract).
3. **Rodila**, **D.,** Bacu, V., and Gorgan, D. (2011). Comparative Analysis of Distributed and Grid Based Execution of SWAT Model, in 3PGCIC 2011 - Sixth International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Barcelona, Spain, October 26-28, 2011, pp. 273-278, http://dx.doi.org/10.1109/3PGCIC.2011.49.
4. **Rodila, D.,** and Gorgan, D. (2011). A Mediation Approach in Geospatial Web Services Gridification, in ICCP2011 – IEEE International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 25-27, 2011, pp. 541-548, http://dx.doi.org/10.1109/ICCP.2011.6047928.
5. **Rodila, D.,** Bacu, V., Ardelean, V., Borlea, C., and Gorgan, D. (2011). Geospatial Web Services Gridification in enviroGRIDS, in European Geosciences Union - General Assembly EGU 2011, Vienna, Austria, April 03-08, 2011, (abstract and presentation), http://meetingorganizer.copernicus.org/EGU2011/EGU2011-11469.pdf .
6. Gorgan, D., Mihon, D., Bacu, V., **Rodila, D.,** Stefanut, T., Colceriu, V., Allenbach, K., Balcik, F., Giuliani, G., Ray, N., and Lehmann, A. (2013). Flexible Description of Earth Data Processing over HPC Architectures, Big Data From Space Symposium, Frascati, Italy, 5-7 June, Abstract Book, pp. 39.
7. Bacu, V., **Rodila, D.,** Pop, O., Dumitru, C., and Gorgan, D. (2012). Graphical Simulation and Visualization of Human Organs, MEI 2012 - International Conference on Medical Education Informatics, 6-7 April, Thessaloniki, http://www.mei2012.org/content/graphical-simulation-and-visualization-human-organs.
8. Bacu, V., Mihon, D., Stefanut, T., **Rodila, D.,** Gorgan, D., Cau, P., and Manca, S. (2011). Grid Based Services and Tools for Hydrological Model Processing and Visualization, in

SYNASC 2011 – 13th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, September 26-28.

9. Bacu, V., Mihon, D., **Rodila, D.,** Stefanut, T., and Gorgan, D. (2011). Grid Based Architectural Components for SWAT Model Calibration, in HPCS 2011 - International Conference on High Performance Computing and Simulation, Istanbul, Turkey, July 4-8, pp. 193-198, ISBN 978-1-61284-381-0.
10. Bacu, V., Mihon, D., **Rodila, D.,** Stefanut, T., and Gorgan, D. (2011). gSWAT Platform for Grid based Hydrological Model Calibration and Execution, in ISPDC 2011 - 10th International Symposium on Parallel and Distributed Computing, Cluj-Napoca, Romania, July 6-8, 2011, pp. 288-291.
11. Gorgan, D., Bacu, V., Mihon, D., Stefanut, T., **Rodila, D.,** Kokoszkiewicz, L., Rouholahnejad, E., Abbaspour, K., and van Griensven, K. (2011). Grid Based Hydrological Model Calibration and Execution by gSWAT Application, 2011 International SWAT Conference, 15-17 June, 2011, Toledo, Spain.
12. Gorgan, D., Bacu, V., Manca, S., Giuliani, G., **Rodila, D.,** Stefanut, T, Mihon, D., Abbaspour, K., Rouholahnejad, E., van Griensven, A., Kokoszkiewicz, L., and Cau, P. (2011). Tools and Applications in enviroGRIDS Project for Spatial Data Processing of Black Sea Catchment Basin, EGI User Forum, Vilnius, 11-15 April, https://www.egi.eu/indico/contributionDisplay.py?contribId=154&sessionId=16&confId=207.
13. Bacu, V., **Rodila**, **D.,** Kokoszkiewicz, L., Rouholahnejad, E., Mihon, D., van Griensven, A., Abbaspour, K., and Gorgan, D. (2011). Calibration and Execution of SWAT Models over GRID Architecture, Presentation to ESSI-12, EGU 2011, Vienna, 3-8 April, 2011.
14. Gorgan, D., Mihon, D., Stefanut, T., Bacu, V., **Rodila, D.** (2011). Grid Based Environment Oriented Tools and Applications for Black Sea Catchment Basin, Presentation to HS12.8, EGU 2011, Vienna, 3-8 April, 2011.
15. **Rodila, D.,** and Gorgan, D. (2010). Integration of Spatial Data Infrastructures with Grid Environment, in SYNASC 2010 – 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, September 23-26, 2010, pp. 269-277, http://dx.doi.org/10.1109/SYNASC.2010.63.

16. **Rodila, D.,** Gorgan, D., and Bacu, V. (2010). The Interoperability between OGC Services and Grid Environment in EnviroGRIDS Project, in 3PGCIC 2010 – International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, Fukuoka, Japan, November 4-6, 2010, IEEE Computer Press, pp. 387-392, ISBN: 978-0-7695-4237-9, http://dx.doi.org/10.1109/3PGCIC.2010.65.
17. Gorgan, D., Stefanut, T., Bacu, V., Mihon, D., and **Rodila, D.** (2010). Grid based Environment Application Development Methodology, in Large-Scale Scientific Computing, Springer Journal LNCS 5910, pp.499-506, ISBN 978-3-642-12534-8, https://dx.doi.org/10.1007/978-3-642-12535-5_59.
18. Gorgan, D., **Rodila**, **D.,** Bacu, V., Giuliani, G., and Ray, N. (2010). OGC and Grid Interoperability in enviroGRIDS Project, in European Geosciences Union - General Assembly EGU 2010, May 02-07, 2010, Vienna, Austria (abstract and presentation), http://meetingorganizer.copernicus.org/EGU2010/EGU2010-13457.pdf .
19. Gorgan, D., **Rodila, D.,** Bacu, V., Giuliani, G., Ray, N., Charvat, K., and Lehman, A. (2010). Geospatial and Grid infrastructures interoperability in enviroGRIDS, 5th EGEE User Forum, April 12-16, 2010, Uppsala, Sweden (2010). Book of Abstracts, http://indico.cern.ch/contributionDisplay.py?contribId=161&confId=69338.
20. Bacu V., Gorgan, V., **Rodila, D.,** Pop, F., Neagu, G., and Petcu, D. (2010). gProcess and ESIP Platforms for Satellite Imagery Processing over the Grid, EGU2010 - European Geosciences Union General Assembly, 02 – 07 May 2010, Vienna, Austria (2010). Geophysical Research Abstracts, Vol. 12, EGU2010-14125-1, eISSN: 1607-7962 (2010). http://meetingorganizer.copernicus.org/EGU2010/EGU2010-14125-1.pdf
21. Bacu, V., **Rodila, D.,** Mihon, D., Stefanut, T. and Gorgan, D. (2010). Error prevention and recovery mechanisms in the ESIP platform, in ICCP2010 – IEEE 6th International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 26-28, 2010, pp. 411 – 417.
22. **Rodila, D.,** Bacu, V., and Gorgan, D. (2009) Semantic Annotation based Service Composition for Grid Workflow Description and Execution, in SYNASC 2009 – 11th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, September 26-29, 2009, pp.245-253, ISBN: 978-0-7695-3964-5, http://dx.doi.org/10.1109/SYNASC.2009.11.

23. **Rodila, D.,** Bacu, V., and Gorgan, D. (2009). Integration of Satellite Image Operators as Workflows in the gProcess Application, in Proceedings of ICCP2009 - IEEE 5th International Conference on Intelligent Computer Communication and Processing, Cluj-Napoca, Romania, August 27-29, 2009, pp. 355-358, ISBN: 978-1-4244-5007-7, http://dx.doi.org/10.1109/ICCP.2009.5284737.
24. Gorgan, D., Bacu, V., **Rodila, D.,** Pop, F., and Petcu, D. (2009). Experiments on ESIP - Environment oriented Satellite Data Processing Platform. SEE-GRID-SCI User Forum, Bogazici University, Istanbul, Turkey, December 9-10, pp. 157-166, ISBN: 978-975-403-510-0.
25. Gorgan, D., Bacu, V., Stefanut, T., **Rodila, D.,** and Mihon, D. (2009). Grid based Satellite Image Processing Platform for Earth Observation Applications Development, IDAACS'2009 - IEEE Fifth International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, 21-23 September, Cosenza, Italy, IEEE Published in Computer Press, pp. 247-252, ISBN 978-1-4244-4901-9.
26. Bacu V., Stefanut, T., **Rodila, D.,** and Gorgan, D. (2009). Process Description Graph Composition by gProcess Platform, HiPerGRID - 3rd International Workshop on High Performance Grid Middleware, 28 May, Bucharest. Proceedings of CSCS-17 Conference, Vol.2., pp. 423-430, ISSN 2066-4451.
27. Gorgan, D., Bacu, V., Stefanut, V., **Rodila, D.,** and Petcu, D. (2009). Grid based Satellite Data Processing Platform in Earth Observation, 4th GRID & e-Collaboration Workshop – Digital Repositories, 25-26 Feb 2009, ESA/ESRIN Frascati, Italy, http://www.congrex.nl/08M16/.

Technical Notes

1. **Rodila, D.,** Chatenoux, B., and Giuliani, G. (2016). Landsat 8 resources - a short comparison of different data access providers and methods (submitted paper).