WPS mediation: An approach to process geospatial data on different computing backends

Giuliani, Gregory; Nativi, Stefano; Lehmann, Anthony; Ray, Nicolas

Computers & Geosciences 47 (2012) 20–33

Contents lists available at SciVerse ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

WPS mediation: An approach to process geospatial data on different computing backends

Gregory Giuliani [a,b,c,*], Stefano Nativi [d], Anthony Lehmann [a,b,c], Nicolas Ray [a,b,c]

[a] University of Geneva, Institute for Environmental Sciences, Climatic Change and Climate Impacts, enviroSPACE Lab., Battelle—Building D, 7 route de Drize, CH-1227 Carouge, Switzerland
[b] United Nations Environment Programme, Division of Early Warning and Assessment, Global Resource Information Database—Geneva, International Environment House, 11 chemin des Anémones, CH-1219 Châtelaine, Switzerland
[c] University of Geneva, Forel Institute, 10 route de Suisse, CP 416, CH-1290 Versoix, Switzerland
[d] Institute of Methodologies for Environmental Analysis, C.da S. Loja—Zona Industriale, I-85050 Tito Scalo, Italy

ARTICLE INFO

Article history:
Received 14 March 2011
Received in revised form
11 October 2011
Accepted 12 October 2011
Available online 20 October 2011

Keywords:
Web Processing Service
Mediation
Grid computing
Geospatial data
PyWPS
Ganga

ABSTRACT

The OGC Web Processing Service (WPS) specification allows generating information by processing distributed geospatial data made available through Spatial Data Infrastructures (SDIs). However, current SDIs have limited analytical capacities and various problems emerge when trying to use them in data and computing-intensive domains such as environmental sciences. These problems are usually not or only partially solvable using single computing resources. Therefore, the Geographic Information (GI) community is trying to benefit from the superior storage and computing capabilities offered by distributed computing (e.g., Grids, Clouds) related methods and technologies. Currently, there is no commonly agreed approach to grid-enable WPS. No implementation allows one to seamlessly execute a geoprocessing calculation following user requirements on different computing backends, ranging from a stand-alone GIS server up to computer clusters and large Grid infrastructures.

Considering this issue, this paper presents a proof of concept by mediating different geospatial and Grid software packages, and by proposing an extension of WPS specification through two optional parameters. The applicability of this approach will be demonstrated using a Normalized Difference Vegetation Index (NDVI) mediated WPS process, highlighting benefits, and issues that need to be further investigated to improve performances.

1. Introduction

Spatial Data Infrastructure (SDI) is a widely accepted concept to facilitate and coordinate the exchange and sharing of geospatial data among different organizations through network technologies (Kiehle et al., 2006). A SDI offers a spatially enabled Service Oriented Architecture (SOA) in which standardized interfaces provide access to functionalities as a set of independent and interoperable services (Granell et al., 2009). The objective of this architectural approach is to promote loosely coupled, standard-based distributed computing so that developed components can be reused. Different standards proposed by the Open Geospatial Consortium (OGC), the International Organization for Standardization (ISO), the World Wide Web Consortium (W3C), and other standardization bodies are used in order to enable interoperability between geospatial data and services. Brauner et al. (2009) subdivide the services that handle geospatial data into three categories: catalog, data, and processing services.

Currently SDIs are mainly concerned with catalog and data services allowing data discoverability, retrieval, and visualization (Baranski, 2008; Schaeffer, 2008). However, the real added value in geospatial data handling is to turn data into usable information to answer a complex query or support a decision. This requires: finding and retrieving data, applying specific calculations, and finally visualizing the result. Commonly, users still process data on their desktop computers using Geographic Information Systems (GIS) software, like ArcGIS[1] or GRASS[2] (Kiehle et al., 2006).

The increasing computational power and network capabilities enable processing of distributed geospatial data over the web (Brauner et al., 2009) using SOA principles and web services technologies. Web-based geoprocessing services can therefore be seen as the next logical step to extend SDI capabilities (Friis-Christensen et al., 2007; Kiehle et al., 2007) by providing access to a collection of geospatial calculations (like in a standalone desktop GIS software) delivering some concrete functionality (Granell

* Corresponding author at: University of Geneva, Institute for Environmental Sciences, Climatic Change and Climate Impacts, enviroSPACE Lab., Battelle—Building D, 7 route de Drize, CH-1227 Carouge, Switzerland.
Tel.: +41 22 917 84 17; fax: +41 22 917 80 29.
E-mail address: gregory.giuliani@unige.ch (G. Giuliani).

[1] http://www.esri.com/software/arcgis/.
[2] http://grass.osgeo.org/.

0098-3004/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cageo.2011.10.009

et al., 2009). Li et al. (2010) have successfully developed a prototype to make available GRASS modules and algorithms using Simple Object Access Protocol (SOAP)-based web services. These authors highlighted that: (a) the interoperability of web services improves the sharing of geospatial data by applications on different platform and (b) the modularity of web services enables the sharing of specific geospatial processes by a wide range of users.

In 2007, the OGC has introduced the Web Processing Service (WPS) specification with the aim to propose a standardized interface for publishing and performing geoprocessing tasks in a web services environment (Open Geospatial Consortium, 2007). In the last years, different implementations have been proposed that demonstrated the applicability of the WPS approach (Kiehle et al., 2006; Stollberg and Zipf, 2007; Brauner and Schaeffer, 2008; Diaz et al., 2008). In particular, the reusability and the possibility to chain processing services and solve specific and complex problems have been emphasized. In addition, these authors showed that servers are in general more powerful than desktop computers allowing users: (a) to process more rapidly a given data set and (b) to process larger data sets (in term of spatial resolution, spatial extent or file size).

However, users can experience a lack of computing power when they process large data sets—such as the global ASTER Digital Elevation Model (DEM) (Hayakawa et al., 2008) at 30 m resolution—or run complex simulations (e.g., dynamic climate models) requiring several CPU hours or days of calculations. In such situation the use of distributed computing appears to be an interesting solution (Lee and Percivall, 2009). Distributed computing is a form of computation in which many calculations are carried out simultaneously on several computing elements linked over a network. The term "distributed" should be distinguished from "parallel" computing that commonly refers to processing tasks that are executed simultaneously on multiple processors on a single computer. Various distributed computing platforms are available such as Grids, Clouds, and Clusters.

Different approaches have been developed to extend SDIs capabilities to use either Grids (Di et al., 2003; Gorgan et al., 2009; Mazzetti et al., 2009; Folino et al., 2010) or Clouds (Baranski et al., 2009). All these authors showed benefits in term of high calculation performance and improved availability of services but also highlighted differences (e.g., service description, service interface, service state, security) between SDIs and distributed computing infrastructures (Padberg and Kiehle, 2009).

Several attempts to implement the WPS specification in a distributed computing environment have been successfully made. Nevertheless, they are in general dependent on the middleware used by the distributed computing infrastructure: some implementations are working on gLite[3] (Muresan et al., 2008; Mazzetti et al., 2009), Globus[4] (Di et al., 2008), or Unicore[5] (Baranski, 2008). In theory, a developed process might be reused across different WPS frameworks.In practice, this is limited due to the use of different programming languages and Application Programming Interfaces (APIs). In other words, a service provider who wants to share a geoprocessing task using the WPS specification must develop a specific version of that process for each specific backend supported by a dedicated WPS implementation.

This means that the scalability in term of execution and reusability of a given WPS process on different computing backends is currently restricted. This situation can potentially limit the development, adoption, and diffusion of WPS.

The aim of this paper is to present a proof of concept to enhance WPS usability allowing one to execute a given geoprocessing task, with a dedicated WPS implementation, independently of the computing backends (e.g., local server, cluster or different Grids/Clouds), thus avoiding the need to rewrite processes by making WPS processes as scalable and flexible as possible.

2. Web Processing Service and distributed computing

The OGC Web Processing Service specification (Open Geospatial Consortium, 2007) provides a standardized way to access geo-processing algorithms in a web service environment, which consequently extends SDIs analysis capabilities (Kiehle et al., 2006; Schaeffer, 2008).

Brauner et al. (2009) reported that performance and processing power are crucial in the context of geoprocessing services, especially in the case of large-scale data sets. To leverage the full potential of WPS, a high performance-computing environment is consequently a key requirement. Distributed computing promises to support SDIs, especially Grids, and Clouds (Foerster and Schaffer, 2007; Baranski, 2008; Baranski et al., 2009; Lee and Percivall, 2009).

2.1. Web Processing Service

In respect of traditional geo-processing implementations, WPS-enabled processes are flexible and remotely accessible algorithms available through web services (Kiehle et al., 2006) that can be reused in different scenarios.

The principal element of a WPS is the notion of process that is a geospatial calculation with defined inputs and outputs (Granell et al., 2009). This implies the following steps: (a) to find suitable geospatial data needed to run the algorithm, (b) to initiate the process, (c) to control the output, and (d) to make the results available to the client.

To work in a web service environment, a WPS instance must offer various operations accessible through web communication. In particular, descriptions of geoprocessing tasks with the help of metadata that are accessible, usable and understandable both by humans and other web services (Kiehle et al., 2006) are key elements to build chains of services (Schaffer and Foerster, 2008). The interface is based on three operations (Fig. 1) which can be called using either HTTP-GET in combination with key-value pair (KVP), or HTTP-POST and eXtended Markup Language (XML)-encoding (Schaffer and Foerster, 2008).

The *GetCapabilities* operation provides an XML document defining service metadata (e.g., server provider, contact information), abstracts and a list of available processes offered by the queried WPS instance. Once users have selected a required process, they can perform a *DescribeProcess* operation to retrieve process metadata within an XML document including parameters descriptions (e.g., input and output parameters). Finally, with the *Execute* operation it is possible to initiate the selected geoprocessing service with all necessary input data. The WPS instance will run the calculation and send back the result to the client.

These three operations are invoked through Uniform Resource Locator (URL) as shown in following examples:

(1) *GetCapabilities* request: `http://localhost/cgi-bin/wps?service=WPS&request=getcapabilities`
(2) *DescribeProcess* request for the "buffer" process:
`http://localhost/cgi-bin/wps?service=WPS&version=1.0.0&identifier=buffer&request=describeprocess`
(3) *Execute* request using "cities.gml" data set as input:
`http://localhost/cgi-bin/wps?service=WPS&version=1.0.0&identifier=buffer&request=execute&datainputs=[data=http://foo.bar/cities.gml;width=5]`

[3] http://glite.web.cern.ch/glite/.
[4] http://www.globus.org/toolkit/.
[5] http://www.unicore.eu/.

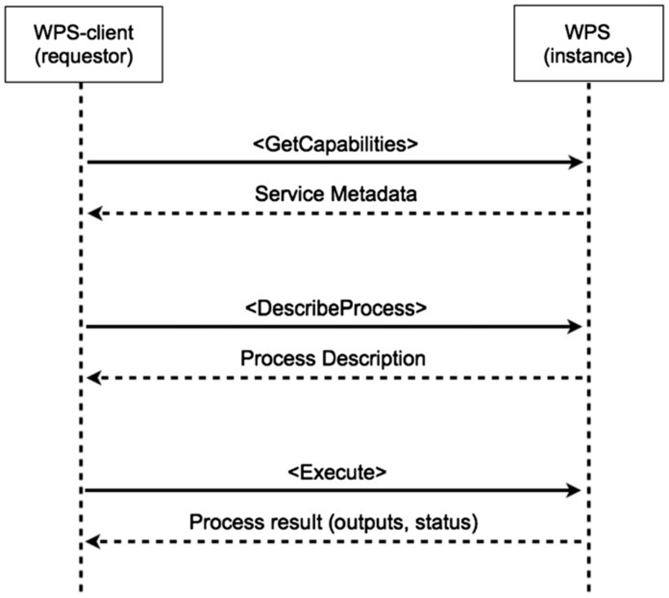


Fig. 1. Communication pattern between a WPS-client and a WPS-instance.

The required input data can either be delivered across a network or made available on a server. The WPS specification discerns three input/output data types:

(1) *LiteralData*: can be any character string like string, float, integer or Boolean.
(2) *ComplexData*: used for vector and raster data by sending directly the data to the server or by referencing a remote data source.
(3) *BoundingBox*: to be used within a specific area.

The output of a geoprocessing service can be obtained either by a direct response to the request (e.g., result sent directly to the client) or as resource stored on the server and accessible through the web using URLs (Schaeffer, 2008). In the latter case, the *Execute* response will be an XML document providing the URLs to access each stored output.

To ensure that processes are reusable, the specification defines WPS Application Profiles enabling optimization of interoperable client user interface behavior, as well as the semantic discovery (i.e., publish/find/bind pattern) and orchestration (Open Geospatial Consortium, 2009; Lanig and Zipf, 2010). Such level of interoperability can be realized only if each process is described in a dedicated Application Profile. Consequently, a WPS Application Profile is a document describing how a WPS shall be configured to serve as a process that is recognized as an OGC WPS process (Open Geospatial Consortium, 2009; Lanig and Zipf, 2010). An application profile consists of: (1) an OGC Uniform Resource Name (URN) to uniquely and unequivocally identify a process (mandatory), (2) a reference response to a DescribeProcess request for that specific process (mandatory), and optionally (3) a human-readable document describing the process and its implementation, as well as (4) a Web Service Definition Language (WSDL) description. Such documentation allows one to formally describe inputs/outputs/semantic of the various geoprocessing operators maintained in dedicated repositories (e.g., web service registries) that are structured following a semantically defined hierarchy of processes, each defined by a URN. This allows the definition of each unique process within the repository and each WPS instance to refer to that URN (Open Geospatial Consortium, 2009; Lanig and Zipf, 2010).

2.2. Distributed computing

The lack of performance (i.e., calculation speed, latencies) when integrating and processing large-scale data sets on a single computer backend limits the types of analyses that can be carried out. Therefore a high performance computing environment is required to extend SDI capabilities. Methods and technologies related to distributed computing have been reported to support SDIs (Brauner et al., 2009). A distributed computing infrastructure can be thought of various autonomous computers that communicate with each other through a network to achieve a common task. When applicable, an incoming task is divided into several smaller sub-tasks executed simultaneously that are merged into an overall result at the end of the process.

There are several categories of distributed computing systems and the most commonly used are: Grids, Clouds, and Clusters. A Grid can be defined as a parallel processing architecture in which computational resources are shared across a network offering access to unused CPU and storage capacities to all participating servers (Foster et al., 2008). Resources can be provided dynamically to users that are looking for computing power. In a Grid environment, users are grouped into specific communities, called Virtual Organization (VO) where they have similar data-intensive goals. Secure authentication is performed using a certificate that allows identifying users unequivocally and grant access to resources only to those how are authorized to use them. Cloud represents the Internet or whatever large network infrastructures where data computation and storage are moved away from local computers being "outsourced" and operated by third-party distributed facilities (Foster et al., 2008; Baranski et al., 2009). This allows users dynamically (i.e., "on-demand") accessing resources (e.g., computational power, storage) without the need to manage the underlying infrastructure. Clouds have an economy-based model focusing on delivering computing resources as services. Finally, a computer cluster is understood as a group of locally managed and linked computers acting like a single powerful computer (Foster et al., 2008). Despite the increasing popularity of Cloud computing, Grids have been specifically targeting the scientific community and consequently are more employed to support SDIs so far. To our knowledge, no significant attempts have been made to use WPS on a cluster. Therefore we concentrate our effort on Grid; it is the most complex infrastructure (for instance in terms of security and non-uniformity of resources), is widely used in the scientific community, and is potentially available to any users who do not have access to local facilities like clusters but who have a valid certificate.

To group distributed computing elements in a Grid infrastructure, a piece of software called "middleware" is required. This middleware enables sharing resources and acts as a layer between heterogeneous hardware and specific user applications. Various Grid middleware are available (e.g., gLite, Globus, Unicore); they must all deal with: job submission, VO management, data management, and security. Applications that use and benefit from a Grid infrastructure are defined as "grid-enabled" (or gridified). For Baranski (2008), the term "gridification" means *"the adaptation of existing applications and services to the requirements and expectations of a grid environment"*. Although this term refers clearly to the Grid environment, we extend this definition in this paper to any application or services that might be adapted to distributed computing environments (e.g., Grids, Clouds, Clusters).

Within the GI community, a clear challenge is to make use of the secure sharing and processing mechanisms offered by Grids while implementing the necessary OGC specifications in terms of interfaces and services for the interoperability sake. Due to their respective nature (e.g., targeted audience, technicalities, standards) connecting Grids and SDIs is not a trivial task, and this procedure requires extensions and customizations (Di et al., 2008). In

Table 1
Encapsulation vs. integration approaches.

	Encapsulation	Integration
Implementation	Easy	Difficult
Gridification level	Low	High
Middleware	Independent	Dependent
Proxy	Not needed	Needed
Level of OWS adaptation	Low, not a Grid service	High, becomes a Grid service
Level of OWS interface extension	Intermediate	Low

consequence, a solution is needed in order to grid-enabled OWS and especially WPS, while hiding the complexity of the Grid (and other distributed computing infrastructures) to SDI users.

3. Gridification approaches

Currently, two types of gridification processes have been recognized: encapsulation and integration (Open Geospatial Consortium, 2009; Shaon and Woolf, 2009). *Encapsulation* is recognized as a "low-level gridification" meaning that applications or services remain unchanged and can interact with distributed computing resources in the backend. For the *Integration* process, applications and services are resources fully embedded into the Grid middleware. Table 1 gives a comparison of the two approaches.

The encapsulation approach allows distributing calculation tasks and accessing data in a Grid environment but applications or services are not intended to become a Grid service as a whole (e.g., no security mechanism). To improve the encapsulated services scalability, the WPS interface must be extended as exemplified by Woolf and Shaon (2009) who encapsulate Job Submission Description Language (JSDL) within a WPS *Execute* operation. Although this approach is interesting and promising, it requires users to additionally understand and deal with JSDL language. Nevertheless, this kind of gridification is relatively easy to perform and the implementation can be considered as independent of the underlying Grid middleware (Baranski, 2008). In comparison, the integrative approach is a more complex task requiring implementing the WPS specification directly into Grid middleware. A good example of such an implementation is the work done by the gLite-OWS (G-OWS) Working Group (Nativi et al., 2009) that is implementing OGC Web Services (OWS) as Grid services within the gLite middleware used by the Enabling Grid for E-sciencE (EGEE) project.[6] The main advantage of this gridification approach is that OWS are implemented as native Grid services and may offer all qualities of any services offered by a Grid environment (e.g., security, information and monitoring, job management, data management). Moreover, this facilitates and allows easier compliance with standards like the Open Grid Forum (OGF), Open Grid Services Architecture (OGSA), the Organization for the Advancement of Structured Information Standards (OASIS), and the Web Service Resource Framework (WSRF) (Yanfeng et al., 2006; Nativi et al., 2009).

Despite the fact that integrative gridification is the most promising type for SDI support, this approach remains rather complex in its implementation because it requires important developments to transform OWS into Grid services (Yanfeng et al., 2006; Di et al., 2008; Muresan et al., 2008; Mazzetti et al., 2009). Moreover, the dependency of this approach on various middleware results in specific OWS middleware implementations. This situation might be potentially solved in the future with the adoption of interoperability standards like OGSA and WRSF. The objective of OGSA is to enable interoperability among various Grid middleware by introducing a service-oriented approach into the Grid (Yanfeng et al., 2006). OGSA defines a Grid service as "a web service that provides a set of well-defined interfaces and that follows specific conventions" (Ghimire et al., 2005). Moreover, a grid service is typically described by the Grid Web Service Description Language (GWSDL), allows service discovery through Universal Description Discovery and Integration (UDDI) registries, and uses SOAP to exchange information. The Open Grid Services Infrastructure (OGSI), which provides recommendations on the infrastructure layer to support OGSA, is evolving towards web services standards through WSRF (Yanfeng et al., 2006). One particularity of Grid services is that they interact continually (e.g., Job Management Service consulting the Resource Discovery Service to find a suitable computing element matching job requirements). Hence, information about their state is required. In other words Grid services are stateful while web services are usually stateless. Consequently, WSRF specifies how to make a web service stateful as required by OGSA. However, the encapsulation approach does not provide Grid services functionalities, WSRF support is lacking, and possibilities to use other computing backends (like Clouds) is difficult, as these might for instance require the implementation of other job submission languages.

3.1. Mediation: a new approach to gridification

The integration of heterogeneous resources in distributed systems is not a recent problem (Wiederhold, 1992) and it is gaining importance with the emergence of large-scale distributed (web-based) applications (Herault et al., 2007). As information systems evolve and extend their scope, they increasingly depend on different heterogeneous resources (e.g., databases, web technologies, computing infrastructures) (Wiederhold and Genesereth, 1997). All these resources are in general developed and maintained separately. Dealing with the diversity and heterogeneity of different resources may therefore be an obstacle to develop high-level applications such as decision-making tools. Hence, integrating disparate resources requires a layer of mediation between clients and different pieces of software or services.

According to Herault et al. (2007) the concept of mediation has emerged as an answer to the lack of interoperability between clients and services in Information Systems (IS). The mediation approach requires identifying situations of particular heterogeneity, and implementing adaptation logic (Nativi and Bigagli, 2009). A specialized and lightweight component, called a mediator, does the execution. A mediation layer can be composed of one or several mediators to possibly form chains managing requests and answers between a client and a service (Fig. 2).

Mediators can accomplish tasks dealing with control (e.g., routing, filtering, aggregation), transformation (e.g., format translation, ontology matching, semantic enrichment), quality of service (e.g., security, transaction), service level agreement (e.g., contract negotiation/management) (Herault et al., 2007). In other words, mediators are used to bind clients and resources implementing some of the binding tasks required to SOA clients (i.e., service consumers).

In the present study, using a mediation concept appears interesting not only to integrate different pieces of software, but mainly in terms of scalability (e.g., adding new computing backends and/or functionalities) and (potential) independency to middleware.

4. Implementation and architecture

To enable the concepts of the mediation approach, a WPS interface implementation was developed. The proposed WPS implementation through the mediation approach was built on an intermediate gridification level and offers possibilities to

[6] http://www.eu-egee.org/.

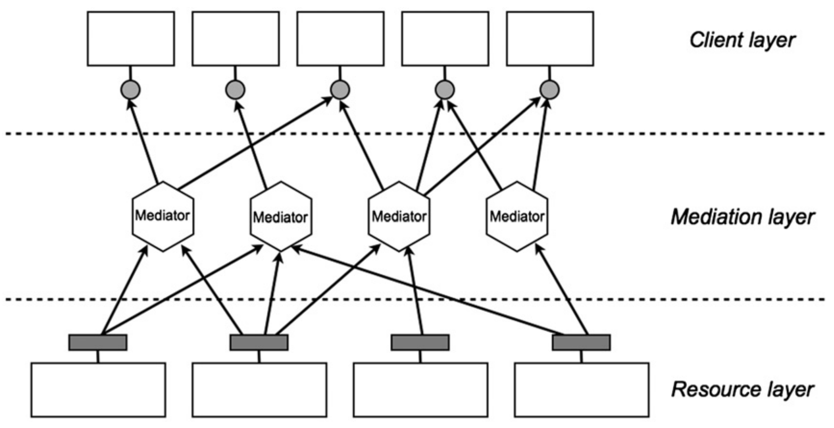


Fig. 2. Mediation layer (adapted from Herault et al., 2007).

overcome some of the previously highlighted shortcomings by integrating and benefiting from the following functionalities offered by various pieces of software:

(1) Simple implementation of WPS specification.
(2) Use of WPS Application Profiles to accommodate the Grid environment.
(3) No dependency on Grid (or other) middleware.
(4) Service providers to write scalable WPS process and users to select backends following their computing power requirements.
(5) The management of data tiling and merging when using a geospatial data set.
(6) Potential access and use of other distributed computing infrastructures such as Clouds (e.g., Amazon Elastic Compute Cloud,[7] Google App Engine[8]), Clusters (e.g., Oracle Grid Engine[9]), and Desktop Grids (e.g., XtremWeb[10] that exploits an array of desktop computers spread over a network to execute distributed tasks).

Traditionally, a WPS instance runs on a local web GIS server using third-party geospatial libraries and/or software (e.g., GRASS, ArcGIS). However, in order to ensure the portability and scalability of WPS on various distributed computing infrastructures, it is necessary to:

(1) use a suitable implementation of WPS, ideally open source, to adapt the code and run it on distributed and heterogeneous Linux platforms;
(2) use a suitable software package allowing the development of tiling and merging modules for raster and vector data sets;
(3) use a suitable software package that deals with security/ access (e.g., authentication, authorization), job submission, and job management in a flexible way on different computing infrastructures.

After reviewing different WPS implementations, tools and software, the following packages were selected:

(1) PyWPS[11] for the WPS implementation.
(2) GDAL/OGR[12] for data handling (i.e., tiling and merging).
(3) Ganga[13] for computational task management.

Therefore, the presented WPS-mediated implementation depends on the selected software packages and on Linux Operating System. However, this specific implementation aims to show the possibilities to overcome, through the general concept of mediation, some issues related to encapsulation or integration approaches.

PyWPS (Cepicky, 2007) is an OGC WPS 1.0.0 implementation written in Python. PyWPS does not process the data by itself but uses GRASS to access geospatial functionalities. This is interesting because it offers the possibility to access the full list of GIS algorithms provided by GRASS, while avoiding the need to rewrite the code of each basic core GIS capabilities (e.g., clip, intersect, buffer) like in other WPS implementations (e.g., Deegree[14]). PyWPS is a Free and Open Source Software (FOSS) distributed under GNU General Public License (GPL). In addition, PyWPS was recommended by the GEOSS, INSPIRE and GMES an Action In Support (GIGAS) project[15] because it is an up-to-date WPS implementation, easy to install on most Linux platforms, and because Python is recognized as a good choice to easily write processes (GIGAS Consortium, 2010). Furthermore, PyWPS offers features that other WPS implementations do not provide. It does not involve complex installation procedures (e.g., compilation), and can be adapted to work as a standalone package. The Geospatial Data Abstraction Library (GDAL) and OGR Simple Feature Library are open source libraries for handling raster and vector geospatial data formats, respectively. Theses libraries are widely used in the GI community, offering various capabilities to manipulate data, and offer Python bindings. They are used to implement data tiling and the corresponding data merging. Finally, Ganga is a Python frontend for jobs specification, submission, management, and monitoring on various distributed resources (Harrison et al., 2006; Moscicki et al., 2009). Ganga provides a simple and consistent environment for processing data on heterogeneous resources; a wide community of users already exists. Ganga allows transparent switching between running test jobs on a local server up to large-scale processing on the Grid (Elmsheuser et al., 2008; Maier, 2008). The aim of Ganga is to hide Grid (and other distributed computing infrastructures) technicalities by offering the possibility to switch computing backend by changing a single parameter of a job (Moscicki et al., 2009). The job is the central concept within Ganga as it contains the complete description of a processing task (e.g., code to execute, input data, specification of computing environment). Another noteworthy capability offered by Ganga is that it can be easily extended and customized through pluggable modules to access different Grid middleware such as Globus, Condor, and Unicore, as

[7] http://aws.amazon.com/ec2/.
[8] http://code.google.com/appengine/.
[9] http://www.oracle.com/us/products/tools/oracle-grid-engine-075549.html.
[10] http://www.xtremwebch.net/.
[11] http://pywps.wald.intevation.org/.
[12] http://www.gdal.org/.
[13] http://ganga.web.cern.ch/ganga/.
[14] http://www.deegree.org/.
[15] http://www.thegigasforum.eu/.

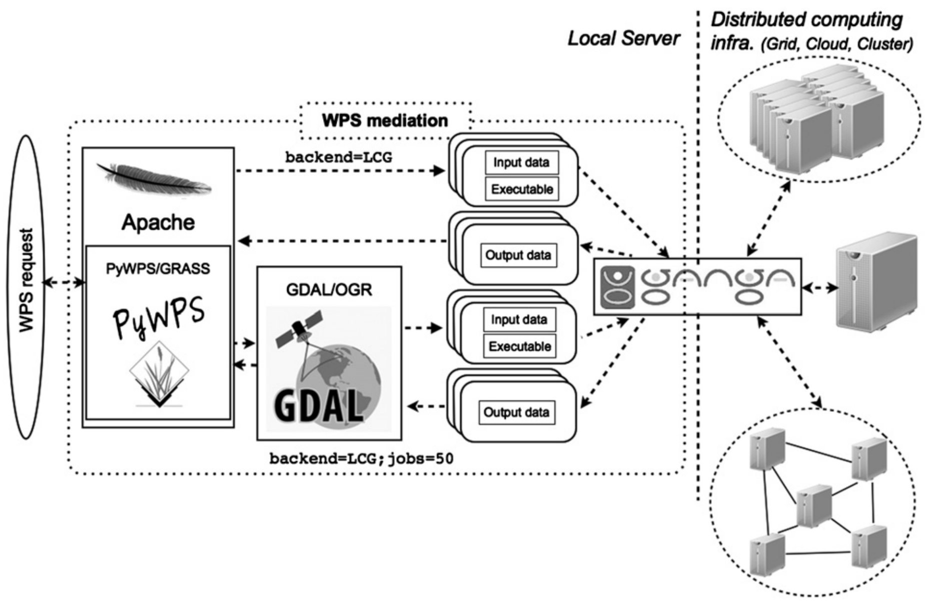


Fig. 3. Technical architecture of the proposed WPS mediation. The WPS mediation layer receives and handles the Execute query, evaluates parameters, and according to them is running the geoprocessing task on the local server or on a selected distributed computing infrastructure. The first option "backend=LCG" will send a list of files to be processed in parallel, while the second option "backend=LCG;jobs=50" allows tiling a large data set according to the number of desired calculation jobs and then submit jobs to the selected backend.

well as different dedicated backends such as PanDA (Vanderster et al., 2010) or ARC (Read et al., 2008). A first attempt to provide a Cloud backend to Ganga (Diaz et al., 2011) is currently under development as part of the CLOBI[16] project. In consequence, this tool allows the use of different computing backends with a common interface acting as an interoperable layer (Moscicki et al., 2009).

It is important to mention that the implemented solution is restricted to a Linux environment because large Grid infrastructures rely on middleware developed for this Operating System (OS).

The proposed WPS implementation based on the mediation approach relying on PyWPS/GRASS, GDAL/OGR and Ganga is presented in Fig. 3. This implementation supports optional HTTP GET and mandatory HTTP POST methods.

Through the use of WPS Application Profiles, the current WPS specification provides a mechanism to choose where to execute a process and how to tile/merge data. Consequently this allows users to choose a computing backend and select the number of jobs (corresponding to the number of tiles) to be sent on working nodes. The term tile refers either to vector or raster data that are divided into smaller chunks. When executing distributed geoprocesses, each job corresponds to a "regular" WPS request sent through Ganga by the mediation layer along with input data (tiled by GDAL/OGR) and a modified version of PyWPS/GRASS binaries. During processing phase, the WPS mediation layer waits until all calculations from individual nodes are finished before merging the results.

The proposed parameters are:

(1) *backend*: for choosing the computing backend (using the codification used in Ganga). For instance, local server (Local), Sun Grid Engine (SGE), and LHC Computing Grid (LCG) are implemented.
(2) *jobs*: for selecting the desired number of tiles and corresponding jobs to be submitted.

[16] http://code.google.com/p/clobi/.

Fig. 4 gives a more detailed insight into the chain of processes involved within the mediation layer. When submitting a WPS query, *GetCapabilities* and *DescribeProcess* are processed regularly: WPS queries the instance and PyWPS sends corresponding XML documents to the client. In the case of an *Execute* request that does not contain the optional *backend* and *jobs* parameters, the query is handled by the mediation layer as a "regular" WPS and is run locally on the web GIS server. However, if the request contains the proposed parameters (in DataInput), then two alternatives are offered to users:

(1) add *backend* argument:`http://localhost/cgi-bin/wps?service=WPS&version=1.0.0&identifier=buffer&request=execute&datainputs=[data=http://foo.bar/cities.gml,http.//foo.bar/rivers.gml;width=5.3;backend=LCG].`
(2) add *backend* and *jobs* arguments:`http://localhost/cgi-bin/wps?service=WPS&version=1.0.0&identifier=buffer&request=execute&datainputs=[data=http://foo.bar/cities.gml;width=5;backend=LCG;jobs=5].`

In the former case, users can select a computing backend and provide a list of data inputs (separated by a comma) that need to be processed in parallel (i.e., task parallelism). In the latter case, data are first tiled according to the number of required jobs and possible dependencies at tile borders (i.e., data parallelism), and then the mediator writes corresponding Python and Shell scripts to submit jobs on different nodes. Once all sub-tasks are finished, Ganga sends back tiled results to a GDAL/OGR Python script for merging into the final result sent back to the client. Different tiling/merging strategies can be applied and may strongly affect performances (Werder and Krüger, 2009). The adopted solution will be further explained under Section 4.1.4

4.1. Development

The three main components that need to communicate in order to handle mediated and "traditional" WPS requests are either

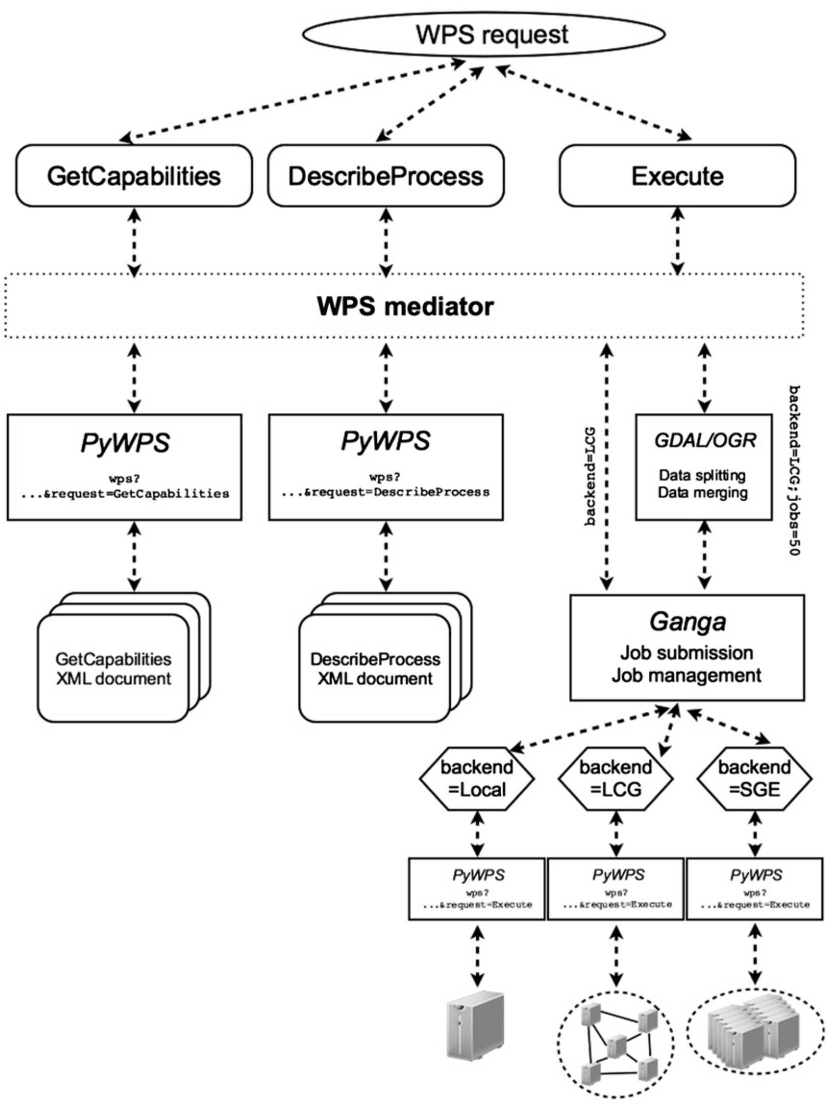


Fig. 4. WPS mediation layer.

written in Python or support Python bindings. Hence, developing a dedicated WPS mediator with this scripting language appeared as a reasonable choice to bind the different software involved. Moreover, Python is generally installed by default on Linux distributions and ensures that processes can be executed on remote computing nodes.

The following mediation layer components were developed/adapted, thus enabling a distributed environment.

4.1.1. *PyWPS/GRASS package adaptation*

The most important step in the implementation process was to adapt the PyWPS/GRASS in order to make it executable on different Linux platforms as a standalone package. Indeed, in distributed computing environments users typically do not know which distribution of Linux is available and do not have administrative rights to install software on remote resources. In consequence, a software package needs to be executable without the need to install any third-party components.

For that purpose, GRASS binaries were downloaded directly from GRASS website. Unwanted or useless components (e.g., documentation files, widgets) were removed and some static libraries were added. Finally, PyWPS source was adapted (e.g., configuration files, functions). The resulting modified PyWPS/GRASS package (available as a tarball) can be run anywhere and independently on the following tested Linux systems (with Python version 2.5 or above): Scientific Linux, Ubuntu, Debian, SUSE, and Fedora.

4.1.2. *Process implementation*

To implement the process logic, PyWPS uses stand-alone Python scripts that have to be written in the PYWPS_PROCESSES directory. A PyWPS process has one class (Process) and two methods: __init__() and execute(). Within the __init__() method, users have to provide some general information (e.g., title of the process, identifier, data input and outputs) while in the execute() method the process itself is implemented calling GRASS commands.

To execute a process on a remote computing node, it is necessary (1) to access locally stored data and binaries and (2) to be able to reference them with path and/or filename. Once a job is submitted data and binaries will be sent together to remote computing nodes of which users do not know the location. Therefore in the process logic itself it is important to make path to data and binaries independent of their location by creating suitable variables.

4.1.3. *Supported data formats and sources*

GeoTiff, Geography Markup Language (GML) and shape files are currently implemented. However, the framework can be easily extended to all formats supported by GDAL[17]/OGR.[18]

4.1.4. *Data tiling and merging*

One of the main advantages of distributed computing is the possibility to divide a given task into several subtasks, which can be executed in parallel. In the case of processing a large vector or raster data set, this means dividing it into smaller sub-sets. For that purpose, the parameter *jobs* has been proposed. Indeed, the input data set will be subdivided according to the number of calculation jobs specified by a user in the mediated WPS request. Werder and Krüger (2009) give detailed descriptions about various issues that can arise when tiling/merging data. These authors highlighted that to efficiently execute parallel geospatial calculations on the Grid it is required to know which geospatial algorithms can potentially benefits from parallelization, which type of parallelism is suitable (i.e., data or task), how parallelization can affect the results and how data are distributed. Hence all these considerations will affect the tiling/merging strategies. In the mediation layer, users choosing between the two types of query (presented under Section 4.1) make the choice between task and data parallelism. If users submit data that need to be tiled (i.e., case 2), it is important to take into account possible correlations at tile borders to obtain continuous results (Werder and Krüger, 2009) (e.g., NDVI process is not influenced by border effects while slope calculation is). The envisioned solution within the mediation layer is to develop WPS Application Profiles describing required parameters to take into account specificities introduced by distributed computing infrastructures and to support efficient tiling/merging strategies. From our point of view, the following parameters are a minimum set that must be considered: (1) whether the algorithm can be parallelized or not, (2) the type of correlations at tile border, (3) the extent of the boundaries, and (4) the smoothing functions to be applied. Depending on the data type (raster or vector, defined in process metadata), the potential inter-dependencies between the created tiles, and the values of the previously mentioned parameters, the mediation layer first calls for the required module (Python script using GDAL or OGR) to subdivide the data accordingly. Second, the layer writes the needed Ganga scripts with the correct references to data. When all processing subtasks are finished, the mediation layer calls the corresponding module to merge data and send it back to the requestor. It should be noted that input and output must be geospatial data.

To achieve these tiling/merging tasks before entering a Grid and submitting jobs, gaining access to the data is essential. Hence, a specific module was developed supporting different data sources: OWS (WFS and WCS) servers, remote data repositories (web accessible folder), and locally stored data (uploaded by File Transfer Protocol (FTP)).

4.1.5. *Job submission*

The mediation layer writes corresponding Ganga scripts to submit jobs on distributed computing infrastructures with the following command:

```
ganga <name_of_script>.py.
```

Each job consists of (1) a PyWPS/GRASS binaries package, (2) data input, and (3) a shell script to execute the WPS request on the remote computing node.

[17] http://www.gdal.org/formats_list.html.

[18] http://www.gdal.org/ogr/ogr_formats.html.

The following example shows a typical Ganga script (Example 1):

```
j=Job()
j.inputsandbox=
['egMediator.tar.gz','TM3.tif','TM4.tif']
j.application=Executable()
j.application.exe=File('testWPS.sh')
j.backend=LCG(middleware='GLITE')
j.backend.requirements=LCGRequirements()
j.outputsandbox=['*']
j.submit()
```

Example 1. Ganga script

The corresponding shell script (Example 2) is used to execute the calculation through a WPS request directly on the remote computing node:

```
#!/bin/bash
abspath=$(cd ${0%/*} && echo $PWD/${0##*/})
path_only=`dirname ''$abspath''`
tar -xzf egMediator.tar.gz
mv TM3.tif $path_only''/''egMediator/datainput/
TM3.tif
mv TM4.tif $path_only''/''egMediator/datainput/
TM4.tif
chcon -t texrel_shlib_t $path_only''/''egMedia-
tor/WPS/grass/lib/*.so
chcon -t texrel_shlib_t $path_only''/''egMedia-
tor/WPS/grass/lib/*.so.1
python $path_only''/''egMediator/egmediator.py
'service=WPS&version=1.0.0&request=execute&i-
dentifier=ndvi&datainputs=[red=TM3.tif;nir=
TM4.tif]'
```

Example 2. Shell script

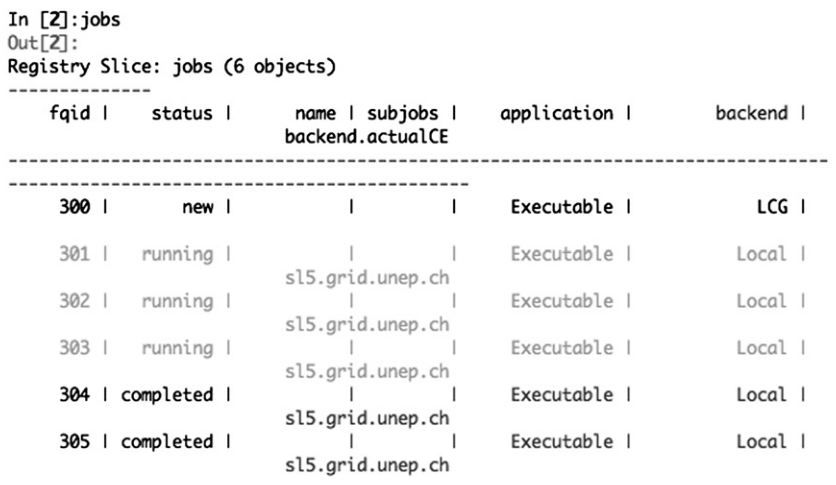
Once jobs are sent to the Grid, users can monitor their status using Ganga (Fig. 5).

Once a job is finished, Ganga retrieves the result by moving it outside the Grid and writing it on a dedicated output folder on the GIS server.

4.1.6. *Security*

To access resources and submit jobs in a Grid environment, users must have a valid long-lived certificate (for authentication and authorization) issued by a certificate authority (CA) trusted by the resource provider. When submitting a job, grid users must first generate a short-lived proxy certificate (i.e., valid in general 12 h) derived from their personal credentials that will be used to authenticate them on the remote resource (Robinson et al., 2005). From a user's point of view, acquiring a certificate is time-consuming, and management of certificates may be difficult especially when accessing various types of computing resources (e.g., Grids, Clouds) that require different authentication processes (Crampton et al., 2007, 2011). Therefore, offering (web-based) single sign-on capabilities may be very useful. This is currently a "hot topic" within the computing infrastructures community, with new venues to manage credentials through federation of CAs (Di Stefano et al., 2009; Jie et al., 2011), Shibboleth (Wang et al., 2009) as well as completely new authentication/authorization mechanisms (Cornwall et al., 2004) being explored.

However, the current WPS specification does not provide any security mechanisms (Shaon and Woolf, 2009). To solve this issue and allow a mediated WPS request to submit jobs on a Grid infrastructure, a common solution is to store the proxy certificate on a central repository protected by a username/password. This



```
In [2]:jobs
Out[2]:
Registry Slice: jobs (6 objects)
--------------
    fqid |    status |      name | subjobs |    application |       backend |
                          backend.actualCE
--------------------------------------------------------------------------------
--------------------------------------------
     300 |        new |             |         |     Executable |           LCG |

     301 |    running |             |         |     Executable |         Local |
                          sl5.grid.unep.ch
     302 |    running |             |         |     Executable |         Local |
                          sl5.grid.unep.ch
     303 |    running |             |         |     Executable |         Local |
                          sl5.grid.unep.ch
     304 |  completed |             |         |     Executable |         Local |
                          sl5.grid.unep.ch
     305 |  completed |             |         |     Executable |         Local |
                          sl5.grid.unep.ch
```

Fig. 5. Jobs monitoring with Ganga (in command line).

consents any Grid client that has access to these credentials to retrieve the proxy certificate (Robinson et al., 2005; Menglong et al., 2009). A widely adopted solution to handle proxy certificates is the online credential repository called MyProxy[19] (Novotny et al., 2001). Users can create a proxy certificate on the MyProxy repository using their long-lived Grid certificate and delegate a username/password combination to the created proxy certificate (Padberg and Greve, 2009). The WPS Application Profile enables users to embed MyProxy parameters (e.g., host, port, username, password) within the WPS request as a special DataInput parameter. Then, the mediation layer manages username/password combination to retrieve short-lived credentials generated by the MyProxy Server and allows the submission of jobs on a Grid. Adding such a security component will introduce an overhead that should be negligible if the request takes a long time to process (Di et al., 2008). However, this approach brings major flexibility and offers the possibility to access Grid resources through a simple authentication mechanism (e.g., username, password) and consequently enables users to access to their credentials (e.g., web browser, mobile devices) seamlessly to submit jobs on a Grid through a WPS request.

4.1.7. *Service metadata*

The extension of the WPS interface with the possibility to execute geoprocessing calculations on different backends requires modifying accordingly the metadata of the service available through the *GetCapabilities* operation. In fact, the objective is to describe an extended capability of the WPS instance. This implies that such extended capabilities must be valid for every process that users will run using the mediated WPS. Hence, the XML document sent to the client must provide a list of supported backends as suggested:

```
<wps:SupportedBackends>
   <wps:Default>
      <ows:Backend>local</ows:Backend>
   </wps:Default>
   <wps:Supported>
      <ows:Backend>local</ows:Backend>
      <ows:Backend>sge</ows:Backend>
      <ows:Backend>lcg</ows:Backend>
   </wps:Supported>
</wps:SupportedBackends>
```

[19] http://grid.ncsa.illinois.edu/myproxy/.

Example 3. Proposed extension for the GetCapabilities XML document providing a list of supported computing backends.

Users can declare different computational backends directly in the PyWPS configuration file that is used by the *GetCapabilities* operation to create the corresponding XML response. Obviously, clients that may query capabilities offered by the mediated WPS instance must support such an extension.

4.1.8. *Process metadata*

The possibility of executing geoprocessing algorithms on distributed computing infrastructures not only requires modifying accordingly the *GetCapabilities* operation but also that these additional capabilities and dependencies be reflected in the description (including inputs and outputs) of each WPS process obtained through a *DescribeProcess* operation. WPS Application Profiles allow enhancing the description of each process by defining inputs/outputs parameters and their values. This mechanism offers the possibility of specifying the dedicated parameters required to accommodate WPS request to distributed computing environments. Such "grid-enabled WPS Profiles" allow, in the current state of the presented implementation, the selection of a computing backend and the execution of the number of jobs following the parameters definition in the profile (see Section 4). Consequently, WPS Application Profiles enable users to embed backend and jobs parameters within a mediated WPS request as special values of the DataInput parameter. Finally, the mediation layer will handle the request to efficiently negotiate with the different pieces of software involved.

Furthermore, using grid-enabled WPS Application Profiles will allow the user to define required parameters in order to manage a secure access to Grid resources (see Section 4.1.6), as well as to develop efficient strategies for data tiling/merging (see Section 4.1.4).

5. Use case: NDVI computation

The proposed WPS mediation layer has been developed and will be tested in the context of the enviroGRIDS project, funded under the European Commission (EC) Seventh Framework Program. This project focuses on the ecologically unsustainable development and the inadequate resource management that is often observed in the Black Sea hydrological catchment area. A large catalog of environmental data sets (e.g., land use, hydrology, and climate) has been gathered and is used to perform distributed spatially explicit simulations to build scenarios of key environmental changes. The project is also developing a Spatial Data Infrastructure (SDI) to feed its regional data into the Global Earth Observation System of

Systems (GEOSS), while being linked to the Enabling Grid for E-sciencE infrastructure (EGEE). During its 4 years timeframe, the enviroGRIDS project aims at reaching the following objectives for which a Grid infrastructure will be important:

(1) A high-resolution (sub-catchment spatial and daily temporal resolution) water balance model will be applied to the entire Black Sea (BS) catchment using the Soil Water Assessment Tool (SWAT) (Arnold et al., 1998). This tool will be gridified and used on the EGEE infrastructure.
(2) Adequate sensitivity and uncertainty analyses will be performed on the BS SWAT model. A gridified version of the SWAT-CUP (Abbaspour et al., 2007) tool will be used for that purpose.
(3) Access to real time data from sensors and satellites will provide early warning and decision support tools to policy-makers and citizens. These data may be streamlined into the grid-enabled enviroGRIDS SDI to ensure fast computation and dissemination of results.

The strong Grid component of the project will foster data interoperability, and is triggering new directions of research or alternative ways of analyzing high-resolution data sets.

One of these directions is the proposed WPS mediation that will be tested to compute Normalized Difference Vegetation Index (NDVI) at the catchment scale. Indeed, vegetation indices time series might be useful data sets to refine SWAT models calibration (Arnold et al., 1998). Vegetation monitoring using remotely sensed data is carried out by means of vegetation indices, which are mathematical transformations designed to assess the spectral contribution of green plants to multispectral observations (Maselli et al., 1998). The NDVI is calculated from Near InfraRed (NIR) and Red (R) bands as follows (Rouse et al., 1974):

$$\mathrm{NDVI} = (\mathrm{NIR}-R)/(\mathrm{NIR}+R)$$

The corresponding WPS geoprocessing task has been developed according to the different requirements mentioned and discussed previously. To study the benefits of the NDVI WPS, we show here its application on two large data sets (Table 2). The first one is a medium resolution remote sensing image (250 m resolution) of 1.2 GB provided by the MODerate resolution Imaging Spectroradiometer[20] (MODIS) and covers the entire Black Sea catchment (2 millions square km). This first data set will be used to test the NDVI process on a local GIS server and, if needed, will be executed on the Grid. The second one is a set of nine high-resolution orthophotos (10 cm resolution) of the city of Geneva (Switzerland), covering an area of one square kilometer each, for a total memory size of 3.6 GB. These images are provided by the "Système d'Information du Territoire Genevois (SITG)[21]". This second data set will be used to make a comparison between different executions of the NDVI process:

– on a merged version of the nine orthophotos on a local GIS server,
– on a merged version of the nine orthophotos on the Grid with 50 jobs (i.e., use of *backend* and *jobs* parameters to tile and merge data),
– on a local GIS server with already tiled orthophotos and submitted as a file list in the *datainputs* parameter using only the *backend* parameter,
– on the Grid with already tiled orthophotos and submitted as a file list in the *datainputs* parameter using only the *backend* parameter.

[20] http://modis.gsfc.nasa.gov/.
[21] http://www.sitg.ch/.

Table 2
Indicative comparison of mediated WPS processes executed locally or on the Grid (times are expressed in minutes/seconds).

	Tiling	Submission	Execution	Merging	**Total**
MODIS, local	–	–	11′51″	–	**11:51**
MODIS, grid (j=10)	2′20″	5′10″	8′40″	3′00″	**19′10″**
SITG—merged, local	N/A	N/A	N/A	N/A	**N/A**
SITG—merged, grid (j=50)	72′20″	40′25″	51′10″	66′40″	**230′35″**
SITG—9 tiles, local	–	–	43′20″	11′20″	**54′40″**
SITG—9 tiles, grid	–	5′20″	30′50″	11′20″	**47′30″**

From the obtained indicatives results several observations can be made:

- Despite the fact that the used MODIS data set might be considered as voluminous (1.2 GB), the developed NDVI WPS process executed on a single GIS server (with 3 GHz CPU/4 GB RAM) can easily handle this data set avoiding the need to use the Grid.
- However, a high-resolution merged image ($30{,}000 \times 30{,}000$ pixels at 10 cm resolution) cannot be processed locally on a single WPS instance due to lack of memory.
- Processing the same data set on the Grid is feasible but important overheads are introduced. In particular, subdividing and merging such a data set in 50 tiles is time consuming. This probably comes from GDAL internal process implementation (e.g., multiple load of the file in memory). Submission and execution on the different nodes can also be quite expensive and dependent of the workload of each computing elements.
- Despite the small overhead introduced by jobs submission, executing calculations on the Grid on already tiled data sets is the most efficient way. In this way, and in contrary to a local execution that processes jobs sequentially, the nine submitted jobs are executed simultaneously. In theory, such parallel execution can improve the overall processing phase by a factor corresponding to the number of submitted jobs (nine in our example). Nevertheless it is not the case here due principally to varying workload conditions on different nodes. The improvement is around 15% and should increase with growing number of jobs (i.e., processing 10,000 tiles will probably show a much larger improvement).

6. Discussion and perspectives

This tool was developed as a proof-of-concept of the mediation approach to grid-enable OGC WPS specifications. The implementation was successful and first results show both benefits and limitations. In particular, this approach enables the possibility of further developing develop WPS implementation (with the help of Application Profiles), offering some of the advantages of a Grid service (e.g. secure access to resources, SOAP-based messaging, statefullness, process scalability).

6.1. WPS mediation benefits

The major benefit of this approach is the ability to flexibly and seamlessly execute WPS on different computing backends, ranging from a local GIS server up to large scale computing infrastructures like a Grid. This scalability is important because it enables users to develop their geoprocesses locally and run them later on various distributed computing infrastructures if more processing power is required. It should be noted that the developed WPS mediator was tested only with the EGEE infrastructure and a local GIS server. Nevertheless, experiments in High Energy Physics successfully used Ganga to submit jobs on various backends (Elmsheuser et al., 2008; Moscicki et al., 2009), and even added new computing backends (Read et al., 2008; Vanderster et al., 2010; Diaz et al., 2011). The presented mediated WPS instance should therefore run without problem on any other infrastructure for which a backend plugin exists in Ganga. In addition, the modular and pluggable framework of Ganga allows one to easily develop new plugins for accessing infrastructures that are not yet implemented such as Clouds or Desktop Grids. Hence, Ganga appears as a suitable solution to hide the complexity of various computing infrastructures by acting as an interoperability layers between different backends and users. Finally, the mediation approach is sufficiently general to be applicable to other WPS implementations and job submission software.

From the results on NDVI processing, it shows that accessing Grid resources allows processing larger data sets than it is possible on a single computer. However, submitting very large data sets (i.e., several GB) is not very efficient as it introduces several important overheads (in particular for tiling and merging data). Nevertheless, this can be seen as additional/optional functionality offered to users that do not have sufficient memory on local resources. A more realistic usage and common practice when serving large data sets (such as those from the SITG) is to make them available already tiled in order to reduce download time and to improve data access to specific areas (i.e., avoiding the need to download a complete data set if not needed). Therefore, submitting these tiles in file lists as data inputs that can be processed in parallel on the Grid appears to be an efficient and promising possibility.

Finally, WPS Application Profile is an interesting feature offered by the current WPS specification. Our specific implementation has shown that it is an efficient way to accommodate WPS request to distributed computing environments. This allows benefiting from the capabilities of the Grid with the simplicity of WPS.

6.2. WPS shortcomings

Actual WPS specification is not sufficient to fully benefit from distributed computing infrastructures capabilities. We showed that service metadata available through a *GetCapabilities* operation must introduce this extension by providing a list of supported backends and process metadata must be extended through WPS profiles to take into account specificities of distributed computing environments.

Additionally, some aspects of the current WPS specification are currently not sufficient to handle different types of references to ensure sufficient scalability within the process logic. Additionally, this issue highlighted the fact that currently available data types in WPS specification are too generic and need to be enhanced (Nash, 2008). Nevertheless, the *ComplexData* type, used to handle vector and raster data in the current WPS specification, does not provide a possibility to reference locally stored data as inputs (i.e., it only addresses remote data sources). Although not foreseen for this usage, the *LiteralData* type helps to overcome this issue. This data type accepts any character strings and then allows introducing path and filename to local resources. If data inputs are declared as such then it is possible to reference locally stored data sets.

6.3. Technical limitations

The current implementation of WPS mediator suffers from different technical limitations that may be overcome with further developments.

At this stage, the mediator only accepts three widely available formats as input: GeoTiff, GML and shape files. However, further development can easily extend the input formats to other vector and raster formats supported by the GDAL/OGR libraries.

When processing data, real scenarios (e.g., generating a flood risk map, analyzing habitat distribution of birds) generally involve various complex tasks/calculations. The process of turning data into information requires the organization of these tasks into sequences or chains of processes. In a web services environment, building flexible and efficient workflows coordinated by an orchestration engine can do this. Orchestration engines are generally based on SOAP and WSDL to exchange structured information while OGC standards rely on XML-RPC (Fleuren and Muller, 2008). These two different ways of representing data may limit the chaining capabilities of OWS. However, WPS has a very limited support of SOAP/WSDL in its current specification. Hence, at this stage of development, it was impossible to effectively test the integration of mediated WPS processes within a chain of services. The release of PyWPS 3.2 will introduce an extended support of SOAP and should therefore facilitate service chaining in the mediation layer.

Finally, OGC standards are in general stateless (i.e., lacking capabilities to give information about their state). However, monitoring jobs status (e.g., submitted, running, complete) is straightforward with Ganga, and some Python bindings should make it possible to develop suitable interfaces to retrieve information about the state of submitted jobs. Additionally, the WPS interface has an optional status parameter that partly enables reporting on the status of execution (Open Geospatial Consortium, 2007).

6.4. Performances

Granell et al. (2009) stated that performance is one of the main problems when implementing geoprocessing tasks in a distributed environment. These authors showed that network bandwidth, data transportation and data validation are potential bottlenecks when dealing with large geospatial data sets. Except in a few cases (e.g., dedicated high performance network), the Internet network bandwidth will always be a limiting factor, as data will be transported over the Internet. Thus, minimizing data transportation (i.e., having data stored physically close to the processing elements) and data validation (i.e., parsing data used within a process) is of high importance.

In order to decrease the overall overhead, the mediation approach might be potentially useful as it allows moving some of the components directly into the Grid. Indeed, a job using the Mediation layer is constituted by (1) process logic, (2) data to be processed, and (3) a PyWPS/GRASS binary package (engine) to execute the process (currently weighting around 27 MB). If the latter package is directly accessible on the nodes of a Grid infrastructure and data is available in Storage Elements (SE), the mediator will be used only to forward WPS requests and send process logic through Ganga. This may considerably decrease the overhead caused by the size of PyWPS/GRASS binaries and data. Such a “moving-code” approach (Müller et al., 2010) can be advantageous as it allows moving the algorithm code, rather than data, to a processing instance. Data is then processed as close as possible to its source, which reduces bandwidth and data transportation time.

Currently Ganga is only used within the mediation layer to submit a geoprocessing task on different backends. Ganga does not consider the workload on different computing nodes. In other words, if a node is available, then a job is submitted. When submitting multiple jobs at the same time, the overall processing time may be negatively affected. In such a situation, Ganga will wait until sufficient nodes are available to submit these jobs. However, in view of integrating Grid capabilities into SDIs, responsiveness appears to be an important condition. In order to overcome this issue, one possible solution it to upgrade the mediation layer using a tool called DIANE (for DIstributed ANalysis Environment)[22] that can be used in conjunction with Ganga. DIANE aims to improve the quality of services using automatic control, workload management and jobs scheduling on a set of distributed worker nodes (Moscicki, 2003). This may (1) reduce the application execution time by using the resources more efficiently, (2) provide fully automatic execution and failure management, (3) efficiently integrate local and Grid resources, and (4) provide mechanisms to efficiently send jobs on nodes that are less loaded (Moscicki, 2004).

Furthermore, it can be interesting to avoid the necessity to let users determine the computing backend and/or the number of jobs that need to be submitted. In the currently implemented solution, users need to know whether the computational requirements of the geoprocessing task require to be distributed. However for an optimum use of computing resources, it would be good to delegate this choice to the mediator. A possible solution to implement this functionality necessitates developing an algorithm that optimizes the job submission and that can be used in conjunction with DIANE. Such an algorithm must consider various parameters such as file size, data complexity (e.g., resolution, geometry) and number of available worker nodes per backend. Based on these parameters (and potentially others), the mediator can then evaluate the resources required to efficiently process the data, select the suitable backend (e.g., local, cluster, Grid), and calculate the number of jobs/tiles to be used. Several attempts (Teo et al., 2003; Hu et al., 2005; Yang et al., 2011) have been made to optimize jobs calculations and processing of remote sensing images. All these studies highlight that achieving the objective of high performance computing of geospatial data is not an easy task, but that investigated solutions of optimization offer good potential.

Other factors that may influence performances of the mediated WPS service are the ones linked to download and tiling modules. If data are not stored locally they need to be downloaded and this may require some time depending on file size and bandwidth. Once download is finished, a dataset is tiled according to the number of jobs requested by the user. In such a case, these tiles can still have a size of several megabytes and must be uploaded to the different computing nodes introducing a data transportation overhead. If tiled data are directly available in SE this allows bypassing the need to download and tile data, thus bringing the induced overhead close to zero. Nevertheless, the extension of the mediation layer to take into account Grid specificities such as data accessible through Logical File Name (LFN) remains to be investigated further. Other possibilities of achieving the objective of fast access to large and widely distributed amounts of data are (1) Peer-to-Peer (P2P) technologies (Yanfeng et al., 2006; Sanchez-Artigas and Garcia-Lopez, 2010) or (2) development of new protocols like GridJet that accelerate data transport (Wang et al., 2009).

Finally, parallelization is an important factor to take into account because it has the potential of strongly affecting the quality of proposed services. For Padberg and Greve (2009) there is no generic parallelization method that fits every geospatial process. Parallelization can indeed be done at the task (i.e., task parallelism) or at the data level (i.e., data parallelism) (Werder and Krüger, 2009). In the former case, a task can be split into various independent subtasks working on the same or different data set, while in the latter case each subtask operates on a subset of the whole data set. Data parallelism introduces the necessity to efficiently tile and merge data (to lower induced overhead) and to consider effects (e.g., interdependencies) at tile borders to obtain adequate result (Werder, 2010). Moreover, not all geoprocessing tasks can be subdivided into independent calculations (e.g., climate models requiring communication among jobs). Consequently, suitable strategies are required. Werder and Krüger (2009) propose possible alternatives to tile/merge data: by object, by geometry, by operation, by spatial criteria, and by level of detail. In the mediation approach, further investigations are required to find an effective and efficient solution using grid-enabled WPS Application Profiles to manage and offer such capabilities to users.

[22] http://it-proj-diane.web.cern.ch/it-proj-diane/.

7. Conclusions

WPS is a promising specification to handle data and a key element to enable SDIs as web-based geoinformation environment. Nevertheless, various issues emerge when trying to use WPS in data and computing-intensive domains like environmental sciences. To overcome these problems a distributed computing paradigm and especially Grid computing appear to be interesting candidates to empower SDIs. However, SDIs and Grids are technologically different, and matching these two types of infrastructures is challenging. For that purpose, OGC WPS specifications need to be improved and extended in order to benefit from superior storage and computing capacities offered by distributed computing.

The presented conceptual approach of mediating different GIS and Grid software components has been exemplified by a specific implementation (i.e., using PyWPS/GRASS, GDAL/OGR and Ganga), offering the possibility to:

(1) potentially execute WPS request on any computing backend implemented within Ganga (e.g., local servers, Clusters, Grids, Clouds) while hiding the technicalities of these infrastructures.
(2) add two optional parameters (through WPS Application Profile): *backend* and *jobs*, allowing users to specify where to execute a selected process and how many sub-tasks are required.

A NDVI WPS process highlighting benefits both in term of performance and scalability has successfully exemplified this implementation. Nevertheless, several issues and shortcomings have been raised and require further investigations in order to build responsive, efficient and reliable grid-enabled SDIs. Some of these shortcomings (e.g., statefullness, asynchronousity, data types, process management) will be tackled in the future WPS 2.0.0 specification that is expected later in 2011. The hope it that the new version improves the grid-enablement process and facilitates the use of distributed computing resources.

The proposed mediation concept appears a promising alternative to encapsulation and integration approaches, opening new perspectives for the grid-enablement process. The mediation concept also enriches the discussion concerning future improvements of the WPS specification along with supporting the Grid community in finding new and promising directions of research and development (Schwiegelshohn et al., 2010).

Acknowledgments

The authors would like to acknowledge the European Commission "Seventh Framework Program" that funded the enviroGRIDS Project (Grant Agreement no. 227640) and UNEP for its support. We thank Dorian Gorgan and his team for their support on Grid technologies. A special thank to Denisa Rodila who helped to install and configure gLite and Ganga. A warm thank to enviroGRIDS task 2.4 (remote sensing) members for providing data and guidance in developing a use case to test the developed tool. We are grateful to the "Système d'Information du Territoire Genevois" for providing orthophotos used to test the NDVI process. We thank Irene Garonna and two anonymous reviewers for their helpful comments on an earlier draft.

References

Abbaspour, K.C., Vedjani, M., et al., 2007. SWAT-CUP Calibration and Uncertainty Programs for SWAT. MODSIM07: Land, Water and Environmental Management—Integrated Systems for Sustainability. Christchurch, New Zealand, pp. 1–7.

Arnold, J., Srinivasan, R., et al., 1998. Large area hydrologic modeling and assessment—part 1: model development. Water Resources Bulletin 34 (1), 73–89.

Baranski, B., 2008. 52° North WPS-G, A Grid-Enabled OGC Web Processing Service (WPS). OGC–OGC Collaboration Workshop. Boston, p. 4.

Baranski, B., 2008. Grid Computing Enabled Web Processing Service. GI-Days 2008. Münster, p. 12.

Baranski, B., Schäffer, B.et al., 2009. Geoprocessing in the Clouds, p. 13.

Brauner, J., Foerster, T., et al., 2009. Towards a Research Agenda for Geoprocessing Services. In: Proceedings of 12th AGILE International Conference on Geographic Information Science. Hannover, Germany, p. 12.

Brauner, J., Schaeffer, B., 2008. Integration of GRASS Functionality in Web Based SDI Service Chains. FOSS4G. Cape Town, South Africa.

Cepicky, J., 2007. PyWPS 2.0.0: The Presence and the Future. Geoinformatics FCE CTU 2007. Prague, Czech Republic.

Cornwall, L., Jensen, J., et al., 2004. Authentication and authorization mechanisms for multi-domain grid environments. Journal of Grid Computing 2 (4), 301–311.

Crampton, J., Lim, H.W., et al., 2007. A Certificate-Free Grid Security Infrastructure Supporting Password-Based User Authentication. In: Proceedings of Sixth Annual PKI R&D Workshop 2007. Gaithersburg, MD, USA.

Crampton, J., Lim, H.W., et al., 2011. User-friendly and certificate-free grid security infrastructure. International Journal of Information Security 10 (3), 137–153.

Di, L., Chen, A., et al., 2003. The Integration of Grid Technology with OGC Web Services (OWS) in NWGISS for NASA EOS Data. GGF8 and HPFC12. Seattle, USA.

Di, L.P., Chen, A.J., et al., 2008. The development of a geospatial data Grid by integrating OGC Web services with globus-based grid technology. Concurrency and Computation—Practice and Experience 20 (14), 1617–1635.

Di Stefano, A., Morana, G., et al., 2009. A credentials management system for secure trade in a grid services marketplace. In: Proceedings of 2009 18th International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises. IEEE, New York, pp. 189–194.

Diaz, L., Granell, C., et al., 2008. Case study: geospatial processing services for web-based hydrological application. Geospatial Services and Applications for the Internet, 31–47.

Diaz, R.G., Ramo, A.C., et al., 2011. Belle-DIRAC setup for using Amazon elastic compute cloud. Journal of Grid Computing 9 (1), 65–79.

Elmsheuser, J., Brochu, F., et al., 2008. Distributed analysis using GANGA on the EGEE/LCG infrastructure. Journal of Physics: Conference Series 072014 (072018)–072014 (072018).

Fleuren, T., Muller, P., 2008. BPEL workflows combining standard OGC web services and grid-enabled OGC web services. In: Proceedings of 2008 34th Euromicro Conference Software Engineering and Advanced Applications (SEAA). IEEE, Parma, Italy.

Foerster, T., Schaffer, B., 2007. A client for distributed geo-processing on the web. Web and wireless geographical information systems. In: Proceedings Seventh International Symposium, W2GIS 2007, Lecture Notes in Computer Science, vol. 4857, pp. 252–263.

Folino, G., Forestiero, A., et al., 2010. A grid portal for solving geoscience problems using distributed knowledge discovery services. Future Generation Computer Systems—The International Journal of Grid Computing—Theory Methods and Applications 26 (1), 87–96.

Foster, I., Yong, Z., et al., 2008. Cloud computing and grid computing 360-degree compared. In: Proceedings of 2008 Grid Computing Environments Workshop. IEEE, Austin, TX.

Friis-Christensen, A., Ostländer, N., et al., 2007. Designing service architectures for distributed geoprocessing: challenges and future directions. Transactions in GIS 11 (6), 799–818.

GIGAS Consortium, 2010. GEOSS, INSPIRE and GMES an Action in Support (GIGAS): D2.2b Data Access and Processing TN-FINAL, p. 81.

Ghimire, D.R., Simonis, I., et al., 2005. Integration of Grid Approaches into the Geographic Web Service Domain. GSDI-8. Cairo, p. 9.

Gorgan, D., Bacu, V., et al., 2009. Grid based satellite image processing platform for Earth observation application development. In: Proceedings of IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. IDAACS 2009.

Granell, C., Diaz, L., et al., 2009. Distributed geospatial processing services. Encyclopedia of Information Science and Technology, Information Science Reference, 1186–1193.

Harrison, K., Tan, C.L., et al., 2006. GANGA: a grid user interface for distributed data analysis. In: Proceedings of the UK e-Science All Hands Meeting, p. 518–525.

Hayakawa, Y.S., Oguchi, T., et al., 2008. Comparison of new and existing global digital elevation models: ASTER G-DEM and SRTM-3. Geophysical Research Letters 35 (17).

Herault, C., Thomas, G., et al., 2007. A distributed service-oriented mediation tool. In: Proceedings of 2007 IEEE International Conference on Services Computing, pp. 403–409.

Hu, Y.C., Xue, Y., et al., 2005. Data-parallel method for georeferencing of MODIS level 1B data using grid computing. In: Sunderam, V.S., VanAlbada, G.D., Sloot, P.M.A., Dongarra, J.J. (Eds.), Computational Science—ICCS 2005, vol. 3516, Pt. 3. Springer-Verlag, Berlin, pp. 883–886.

Jie, W., Arshad, J., et al., 2011. A review of grid authentication and authorization technologies and support for federated access control. ACM Computing Surveys 43, 2.

Kiehle, C., Greve, K., et al., 2006. Standardized geoprocessing—taking spatial data infrastructures one step further. In: Proceedings of Ninth AGILE Conference on Geographic Information Science. Visegrad, pp. 273–282.

Kiehle, C., Greve, K., et al., 2007. Requirements for next generation spatial data infrastructures—standardized web based geoprocessing and web service orchestration. Transactions in GIS 11 (6), 819–834.

Lanig, S., Zipf, A., 2010. Proposal for a web processing services (WPS) application profile for 3D processing analysis. In: 2010 Second International Conference on Advanced Geographic Information Systems, Applications, and Services (GEOPROCESSING).

Lee, C.A., Percivall, G., 2009. The evolution of geospatial e-infrastructures. GIS Science 3, 68–70.

Li, X., Di, L., et al., 2010. Sharing geoscience algorithms in a web service-oriented environment (GRASS GIS example). Computers and Geosciences 36 (8), 1060–1068.

Maier, A., 2008. Ganga—a job management and optimising tool. Journal of Physics: Conference Series 072021 (072018)–072021 (072018).

Maselli, F., Gilabert, M.A., et al., 1998. Integration of high and low resolution NDVI data for monitoring vegetation in Mediterranean environments. Remote Sensing of Environment 63 (3), 208–218.

Mazzetti, P., Nativi, S., et al., 2009. A grid platform for the European civil protection e-infrastructure: the forest fires use scenario. Earth Science Informatics 2 (1), 53–62.

Menglong, Y., Yong, G., et al., 2009. Spatial data access control in grid environment. In: Proceedings of 2009 17th International Conference on Geoinformatics. IEEE, Fairfax, VA.

Moscicki, J.T., 2003. Distributed analysis environment for HEP and interdisciplinary applications. Nuclear Instruments and Methods in Physics Research Section A—Accelerators Spectrometers Detectors and Associated Equipment 502 (2–3), 426–429.

Moscicki, J.T., 2004. DIANE—distributed analysis environment for GRID-enabled simulation and analysis of physics data. 2003 IEEE Nuclear Science Symposium, Conference Record 1–5, 1617–1620.

Moscicki, J.T., Brochu, F., et al., 2009. GANGA: a tool for computational-task management and easy access to Grid resources. Computer Physics Communications 180 (11), 2303–2316.

Müller, M., Bernard, L., et al., 2010. Moving code in spatial data infrastructures—web service based deployment of geoprocessing algorithms. Transactions in GIS 14 (S1), 101–118.

Muresan, O., Pop, F., et al., 2008. Satellite image processing applications in MedioGRID. In: Proceedings of Fifth International Symposium on Parallel and Distributed Computing.

Nash, E., 2008. WPS application profiles for generic and specialised processes. In: GI-Days 2008: Proceedings of the Sixth Geographic Information Days. IFGI, University of Munster, Muenster, Germany.

Nativi, S., Bigagli, L., 2009. Discovery, mediation, and access services for earth observation data. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 2 (4), 233–240.

Nativi, S., Verlato, M., et al., 2009. G-OWS (gLite-OWS) Working Group Charter. In: Nativi, S., Verlato, M., Pacini, F. (Eds.), p. 17.

Novotny, J., Tuecke, S., et al., 2001. An online credential repository for the grid: MyProxy. In: Proceedings of the Tenth IEEE International Symposium on High Performance Distributed Computing. IEEE Computer Society, p. 104.

Open Geospatial Consortium, 2007. OpenGIS Web Processing Service, p. 87.

Open Geospatial Consortium, 2009. OGC OWS-6 WPS Grid Processing Profile Engineering Report. In: Baranski, B. (Ed)., p. 93.

Padberg, A., Greve, K., 2009. Gridification of OGC web services: challenges and potential. GIS Science 3, 77–81.

Padberg, A., 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 16.

Read, A., et al., 2008. Complete distributed computing environment for a HEP experiment: experience with ARC-connected infrastructure for ATLAS. Journal of Physics: Conference Series 119 (6), 062041.

Robinson, J.P., Gemmill, J., et al., 2005. Web-enabled grid authentication in a non-Kerberos environment. In: Proceedings of Sixth IEEE/ACM International Workshop on Grid Computing. IEEE, Seattle, WA.

Rouse, J.W., Schell, J.A., et al., 1974. Monitoring the Vernal Advancement of Retrogradation of Natural Vegetation. NASA/GSFC, Type III. Final Report, pp. 1–371.

Sanchez-Artigas, M., Garcia-Lopez, P., 2010. eSciGrid: a P2P-based e-science Grid for scalable and efficient data sharing. Future Generation Computer Systems—The International Journal of Grid Computing—Theory Methods and Applications 26 (5), 704–719.

Schaeffer, B., 2008. Towards a Transactional Web Processing Service (WPS-T). GI Days. Münster, Germany, p. 27.

Schaffer, B., Foerster, T., 2008. A client for distributed geo-processing and workflow design. Journal of Location Based Services, 194–210.

Schwiegelshohn, U., Badia, R.M., et al., 2010. Perspectives on grid computing. Future Generation Computer Systems—The International Journal of Grid Computing—Theory Methods and Applications 26 (8), 1104–1115.

Shaon, A., Woolf, A., 2009. Grid-enabling OGC Web Services. In: Proceedings of UK e-Science All Hands Meeting 2009. Oxford, UK, pp. 1–3.

Stollberg, B., Zipf, A., 2007. OGC Web processing service interface for web service orchestration: aggregating geo-processing services in a bomb threat scenario. Web and wireless geographical information systems. In: Proceedings of Seventh International Symposium, W2GIS 2007 Lecture Notes in Computer Science, vol. 4857, pp. 239–251.

Teo, Y.M., Low, S.C., et al., 2003. Distributed geo-rectification of satellite images using Grid computing. In: Proceedings of International Parallel and Distributed Processing Symposium (IPDPS 2003). IEEE Computer Society, Nice, France.

Vanderster, D.C., et al., 2010. A PanDA backend for the ganga analysis interface. Journal of Physics: Conference Series 219 (7), 072056.

Wang, F.Z., Helian, N., et al., 2009. Eight times acceleration of geospatial data archiving and distribution on the grids. IEEE Transactions on Geoscience and Remote Sensing 47 (5), 1444–1453.

Wang, X.D., Jones, M., et al., 2009. Shibboleth Access for Resources on the National Grid Service (SARoNGS). IEEE Computer Society, Los Alamitos.

Werder, S., 2010. Data Integration in a modular and parallel grid-computing workflow. In: Proceedings of WebMGS 2010—First International Workshop on Pervasive Web Mapping, Geoprocessing and Services. Como, Italy.

Werder, S., Krüger, A., 2009. Parallelizing geospatial tasks in grid computing. GIS Science 3, 71–76.

Wiederhold, G., 1992. Mediators in the architecture of future information-systems. Computer 25 (3), 38–49.

Wiederhold, G., Genesereth, M., 1997. The conceptual basis for mediation services. IEEE Intelligent Systems and Their Applications 12 (5), 38–47.

Woolf, A., Shaon, A., 2009. An approach to encapsulation of grid processing within an OGC web processing service. GIS Science 3, 82–88.

Yanfeng, S., Fan, Z. Jack, et al., 2006. A grid-enabled architecture for geospatial data sharing. In: Proceedings of 2006 IEEE Asia-Pacific Conference on Services Computing. CD-ROM, 7 pp.

Yang, C.W., Wu, H.Y., et al., 2011. Using spatial principles to optimize distributed computing for enabling the physical science discoveries. Proceedings of the National Academy of Sciences of the United States of America 108 (14), 5498–5503.