OGC Compliant Services for Remote Sensing Processing over the Grid Infrastructure

Danut Mihon, Vlad Colceriu, Victor Bacu,
Denisa Rodila, Dorian Gorgan
Computer Science Department
Technical University of Cluj-Napoca
Cluj-Napoca, Romania
{vasile.mihon, vlad.colceriu, victor.bacu,
denisa.rodila, dorian.gorgan}@cs.utcluj.ro

Karin Allenbach[1, 2], Gregory Giuliani[1, 2]
[1]Institute for Environmental Sciences, enviroSPACE
University of Geneva 1227 Carouge, Switzerland
[2]United Nations Environment Programme,
Global Resource Information Database
1211 Châtelaine, Switzerland
allenbach@unepgrid.ch, gregory.giuliani@unige.ch

***Abstract*—The latest issues in simulating and analyzing different Earth Science phenomena require the development of complex algorithms, based on satellite images in different formats. The main goal of this paper is to process these data in a standard manner and to automatically publish the execution results by using the latest Web Processing Services (WPS). The development of these services needs to be slightly different when involving large volume of data processed over the Grid infrastructure opposed to standalone machines. This paper provides an implementation solution of the WPS standard within the GreenLand platform, and exemplifies it on the Black Sea catchment hydrologic modeling use case.**

I. Introduction

This paper concerns with the integration of the Web Processing Services (WPS) into the GreenLand application [1] that is built over the Grid infrastructure. The research is part of the FP7 enviroGRIDS project [2], funded by the European Commission through contract no. 226740.

GreenLand is a Grid based software application that operates in the Geographic Information System (GIS) domain, and it is used for geo-spatial data management and analysis, satellite image processing, graphics/maps generation, spatial modeling and visualization. In particular this application offers support for three major case studies: Black Sea catchment hydrologic modeling, land cover/land use analysis of the Istanbul geographic area, and the Rioni river hydrologic analysis [3].

By geo-spatial data we mean raster inputs such as satellite images in different formats (e.g. Moderate Resolution Imaging Spectro-Radiometer (MODIS), Landsat, Aster, etc.) and vector inputs (e.g. ESRI shapefiles). These data are retrieved from remote repositories or from the user's local machine, and converted into a GeoTIFF internal representation.

There are several objectives that are highlighted throughout this paper: OGC WPS integration within the GreenLand platform, the access mechanisms to the GreenLand WPS services, and the interactive implementation of new geospatial algorithms based on the WPS guidelines.

The first one proposes the integration of the Open Geospatial Consortium (OGC) standards - WPS in particular [4] - with the Grid infrastructure, where the GreenLand application is the intermediate communication layer. The advantages of performing this integration are: standardized data access and processing, interoperability with external platforms, flexible data models for storing and exposing the spatial information.

The WPS standard was successfully used in small scale processing, where the computation complexity was not that high. Different issues were encountered in cases of large scale scenarios, where the standard does not provide enough details about the geo-spatial data execution, monitoring, and results presentation.

The three GreenLand case studies, described earlier, require an intensive processing of a large volume of spatial data. All these aspects motivate the usage of the Grid infrastructure and the necessity of extending the WPS standard towards the Grid platform.

The WPS package consists of a set of three services (GetCapabilities, DescribeProcess, and Execute) that are accessible over the Web. Another objective of this paper is related with the usage of these services: directly from the GreenLand application, API access from external platforms, and browser based access.

Usually the Execute service requires a list of mandatory parameters, such as the path to the inputs data set that are going to be used at runtime. In some cases this attribute is hard to be written manually (e.g. in case of accessing the WPS as a link given in the Web browser address bar). Instead an automated generation of the path could be achieved from the GreenLand application.

This is the main reason why the GreenLand-WPS integration is the most easy to use solution (against the three available ones, described earlier), regardless of the users' experience in computer science domain. More details about these concepts are going to be presented in the following sections.

The last objective of this paper is related to the possibility of adding new WPS processes or updating the existing ones. This is possible due to the PyWPS [6] tool that allows the specification of each process as a Python script. The flexible model of the extended WPS standard has an important role in achieving this objective that aims to improve the platforms interoperability, by re-using these algorithms (implemented on external platforms) instead of developing them from scratch.

II. Related work

Different implementations of the WPS standard demonstrated the benefits of this approach [5]. Small scale processing is the common characteristic for all these solutions. When using this standard for complex use cases, extra computation resources need to be added. The most appropriate methods are related to the distributed infrastructures, Grid and Cloud in particular that offer high power computation and storage resources.

There are several research studies of large scale applications development over the Grid [7], [8], [9] and Cloud infrastructures [10]. None of them experiments the WPS integration, and use instead non-standardized executions.

The GreenLand platform also uses the Grid infrastructure for parallel and distributed computations. What differentiates it from the previous mentioned applications is the implementation of OGC-based mechanisms that allow a standardized execution of spatial data. This means that the GreenLand platform is able to interoperate with external systems for sharing, processing, and visualizing of spatial data.

The majority of Earth Observation researches recommend standard access to geo-spatial data. Article [11] provides a proof of concept solution for executing the spatial data on certain backend infrastructures, by using a mediation approach. The paper [12] proposes an integration of the WPS standard with the Cloud infrastructure by using the Hadoop Map-Reduce model.

This paper describes the integration of the WPS standard directly within the GreenLand platform that allows the possibility of developing complex use cases in an interactive manner, closely related to the human natural language. This approach differentiates from the previous mentioned solutions, and allows user access to these features, without the need to have background knowledge related to computer science domains.

There are multiple frameworks that implement the WPS standard. The 52n WPS version could be used for deploying services on standalone machines. It provides a standardized access to data, and allows the creation of new processes through the Development Kit that was integrated in the last release. It uses the R language for implementing the geo-spatial algorithms [13].

Degree WPS [14] is another framework that implements this standard. Currently it is integrated with some of the most known spatial data processing applications, such as Sextante [15]. The Geographic Resources Analysis Support System (GRASS) library [16] is used for implementing different geo-spatial algorithms, and it is used especially on standalone machines.

The PyWPS [6] is a Python based framework that uses the GRASS library for describing the geo-spatial features. On the other hand it offers the possibility to access remote services that provide the same types of algorithms. Because the GreenLand platform uses the GRASS library for the operators' development, the PyWPS implementation was adopted as support for accessing and executing these operators by following the WPS standards.

III. OGC standard general overview

The rules and guidelines of data sharing represent the basis for several standards organizations that put them into practice in the GIS domain fields. According to these issues, they manage to increase the interoperability between systems and geo-spatial data.

There are several standards that are related with the GIS system (e.g. Open Geospatial Consortium OGC [4], International Organization for Standardization ISO [17], Spatial Data Transfer Standard SDTS [18], Organization for the Advancement of Structured Information Standards OASIS [19]), but for spatial data management the most important ones are OGC and ISO.

The Open Geospatial Consortium (OGC) is a non-profit organization that provides guidelines for service oriented spatial data processing and visualization. Based on these standards, the developers are able to create interoperable services for information access and information exchange, but also complex data structures that could be accessible to a large number of applications. The goal of the OGC organization is to provide standards for developers and users in order to produce services for accessing spatial data, and to assure that geo-spatial interoperability:

- Allows the creation of standards (integrated into daily processes) regarding spatial data computing;
- Facilitates interoperability between GIS applications;
- Facilitates the implementation of open architectures.

Without interoperability and standardization, data access and data exchange is really difficult among organizations. In general, any Web service must have the ability to describe its own capabilities, enabling this way other services and products to interoperate based on its standard functionalities. The OGC usage allows this process flow between different GIS software applications without the need to develop new translation mechanisms to assure their interconnection.

There are several OGC standards that are commonly used for accessing, visualizing, analyzing, and processing data throughout Web services:

- Web Map Service (WMS): defines a Web interface that allows geo-referenced data retrieval as map layers. For security reasons the WMS service does not give access to the real data, but instead it creates different layers representation (e.g. JPEG, PNG, TIFF) of this data.
  The WMS interface is a three-steps process that consists in the following operations: GetCapabilities (exposes the service functionalities and lists all its available layers), GetMap (produces a map based on the selected layers set), and GetFeatureInfo (returns information about the generated map content).
  For each WMS request there are lists of mandatory and optional parameters. The response is an XML file that contains information about the service and the data availability;
- Web Feature Service (WFS): it allows features retrieval and management using the GML format. It is intended to be used only for vector datasets, while the

WCS works mainly on raster images. The WFS interface is a thee-steps process: GetCapabilities (similar to the one described for WMS), DescribeFeatureType (defines the structure of the feature), and GetFeature (returns the response encoded with GML schema).
There are two implementations of this standard: basic WFS and transactional WFS. The first one is used to query and to retrieve features, while the WFS-T provides services for features creation, deletion, and update;

- Web Coverage Service (WCS): provides standardized access to raster datasets. Like the rest of the OGC services, the WCS interface consists in three types of requests: GetCapabilities, DescribeCoverage, and GetCoverage. Based on the XML result, generated by the GetCapabilities, the user is able to select and download data for a specific area of interest. The area's geographic coordinates, image format, width, height, and projection are a few of the mandatory fields required by this service;
- Web Processing Service (WPS): besides data accessibility, the OGC standard also provides geo-spatial data processing services through the WPS interface. It allows users to: know what processes are available, to select the proper input data, to create and run different models, to perform management operations to the output results, etc.
  As any OGC standard implementation, it includes three types of operations: GetCapabilities (returns a list of the service capabilities together with all its available processes), DescribeProcess (for each process it provides a general description of the parameters and their types), and Execute (performs the process execution. It does not offer tools for monitoring the execution progress, and once the result is available it is send to the user);
- Simple Features SQL (SFS): It is an open standard that offers rules and guidelines for storing, querying, updating, and retrieving geo-spatial features from SQL databases. SFS establishes an architectural framework for features representation, provides syntaxes for defining geometric attributes attached to those features, and describes a set of geometry types in order to ease the data exchange processes.

The GreenLand platform implements the majority of the previous mentioned services (WMS, WCS, and WPS), but this paper offers details only about the Web Processing Service.

IV. System related architecture

This section describes the concepts, the solutions, and the technologies involved throughout the experiments of integrating the WPS standard within the GreenLand platform. The high power computing resources are crucial in optimizing the processing of large volume of geo-spatial data [20].

In order to fulfil the three objectives proposed in this paper, a combination of multiple tools and technologies (Figure 1) was performed, such as: the Grid infrastructure for geo-spatial data processing, the OGC services for standardized data access and specification, PyWPS that acts like a middleware between the WPS Execute operation and the hardware platform, the GreenLand together with the gProcess and GRASS libraries that provides a framework for developing and using the WPS related services.

The Grid infrastructure could be defined as a worldwide computer network that offers parallel and distributed support for storing and processing large volume of data [21].

A. *WPS general overview*

The Open Geospatial Consortium (OGC) is a non-profit organization that provides guidelines, rules, and software API, recognizable throughout the entire geo-spatial community. Based on these standards, the developers are able to create interoperable services for information access and information exchange, but also complex data structures that could be processed by a large number of applications [4].

The OGC Web Processing Service allows standardized access to data and geo-spatial algorithms, by using the Web technologies. There is a list of mandatory parameters that must be attached to each such service. The inputs and outputs lists, type of request, version of the WPS, and the unique identifier for the algorithm are the most frequently used.

The WPS standard includes three types of operations, accessible as URLs: GetCapabilities, DescribeProcess, and Execute. The GetCapabilities is used for obtaining certain information about all the available services. This information is packed within an XML metadata document that specifies their identifiers, name and description, the provider, the type of projection, etc. The URL http://<server_domain>/wps/wps.py?service =wps&version=1.0.0&request=GetCapabilities can be used to exemplify this operation.

The <server_domain> represent the URL location of the PyWPS server. The service, version and the request parameters are mandatory for all WPS operations and are used to identify the OGC standard (WPS in this case) and its implementation version.

For detailing a particular service, the DescribeProcess operation should be used. Based on the unique identifier, the inputs and outputs lists for the algorithm could be retrieved. The URL associated with this operation has the following structure: http://<server_domain>/wps/wps.py?service =wps&version=1.0.0&identifier=NDVI&request=Describe Process.

The identifier parameter is needed only for the DescribeProcess and the Execute operations that are closely related with a specific process.

The last operation allows the execution of a certain process, based on a well-defined list of inputs data set. One such example is given bellow: http://<server_domain>/wps/wps.py?service=wps&version=1.0.0&identifier=NDVI&dataInputs=[input1=value1; input2= value2; …;inputn=valuen]&request=Execute. For this operation, the URL structure is more complex, because it involves the specification of a valid path to the input data that is stored locally or on remote repositories.

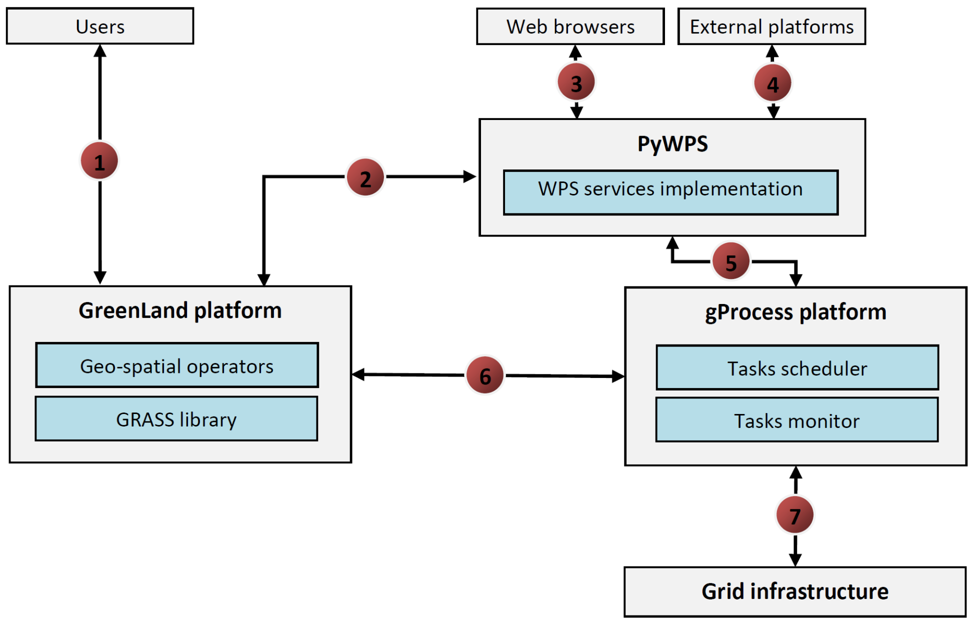

Fig. 1. System related architecture

B. *Modules interconnection and characteristics*

The PyWPS is a software application, written in Python programming language that implements the WPS standard. It offers the possibility to access HTTP services that contain different geo-processing algorithms. On the other hand, the PyWPS also allows the direct access of GRASS functions.

In cases of executing large scale scenarios, the direct usage of the GRASS features is not enough. The GreenLand platform addresses such complex use cases that need tens of minutes for completing their execution. This is the main reason why a new algorithm development method was implemented that uses the PyWPS for accessing them over the HTTP protocol.

The gProcess tool [22] could be defined as an intermediate layer between the GreenLand platform and the Grid infrastructure. All the geo-spatial algorithms within this platform are transcoded at runtime as nodes of a workflow (graph). Using this data model, it is easier to execute and monitor the processes launched over the Grid infrastructure.

The gProcess platform is used for scheduling the tasks over the worker nodes of the Grid. In this case, each node of the workflow is considered to be a task. This process involves the placement of all tasks within a queue and dynamically deploys them on the Grid CPUs. When the number of tasks is greater than the number of available hardware resources, the gProcess scheduler waits until one of the worker node finishes, and assigns it with another task. The execution of the entire workflow is completed when all its inner nodes (tasks) are successfully processed.

The gProcess also offers a monitor mechanism that provides to the users information about the status of the execution. Based on this feature the users are able to remotely control the Grid executions.

The geo-spatial algorithms within the GreenLand platform were implemented based on the functionalities provided by the GRASS library. Currently, this library consists of more than 400 data processing modules that are organized in several categories: vector, raster, image processing, database management, etc.

The GRASS product uses the GDAL and OGR libraries [23] for data conversion between multiple types. GRASS modules could be accessed based on command line instructions. This way it is easier to integrate them with other GIS applications [16].

Figure 1 highlights how the previous mentioned components are related to each other. The WPS standard is implemented through the PyWPS server that could be accessed in three ways: directly from browsers by using the URL structure, from external platforms based on the API provided by the server, and from the GreenLand application.

Integrating the WPS with the Grid infrastructure generates issues that are not encountered in the traditional usage of the standard:

- Create a flexible data model that allows the development and the maintenance of the WPS processes. In order to deal with this issue, the PyWPS tool was chosen. It provides the possibility to access (via HTTP) remote algorithms that describe complex use case scenarios. The Python script is simple, and includes definition of the inputs and the output of the process.
  The remote algorithm (resident on the GreenLand platform) is the core of this flexible data model. The Python script is not affected when this algorithm changes its inner functionality, but only in cases in which the inputs or the output parameters are updated;
- In traditional usage of the WPS services, the data security is not that important. On the other hand, the Grid infrastructure filters its users, and offers

the execution rights only to the ones that have valid certificates, emitted by a Certification Authority (CA). This security issue is solved at the GreenLand platform level, where only users with valid credentials are allowed to use its functionality;

- Usually the time required to execute a WPS process is not that high. This gives the possibility of presenting the output result to the user almost instantaneously (in the same request-response loop).
  Processes that use the Grid infrastructure, demand powerful computing resources, and their execution take a long time to complete (e.g. tens of minutes). As seen, there is no possibility to provide an immediate final result to the user. Instead the GreenLand application periodically interrogates the execution status, and when completed, it offers the user the possibility to download the result, based on an URL address.

V. Implementation

This section describes the relationships between the components from Figure 1, and tries to provide the best solutions to different issues that arose on the way. The implementation of the WPS standard, within the GreenLand platform, is a 7-steps process that symbolizes each phase with a filled circle, like in the previous figure.

The first step represents the user authentication within the GreenLand platform, by using the username and password credentials. Once the user is authenticated, it is assigned with a valid Grid certificate that could be used for later executions.

The GreenLand platform, among other functionalities, provides a list of geo-spatial operators that were developed by using the GRASS library. These algorithms are from different Earth Science domains, and prove to be useful for: land cover/land use analysis (e.g. NDVI, Accuracy assessment, Density slicing, etc.), hydrologic modeling (e.g. Black Sea catchment use case study), atmospheric pollution, etc.

By default, the execution of these operators does not involve the WPS standard. Instead it is performed on a regular basis, by submitting Grid jobs through the gProcess platform in a non-standardized manner. This type of execution corresponds to the following sequence of steps: 1, 6, and 7 (see Figure 1).

One of the objectives of this paper is to prove that the WPS standard could be successfully used within the Grid parallel and distributed executions. Steps sequence: 1, 2, 5, and 7 represent an alternative WPS scenario for non-standard execution of the GreenLand processes. The main difference from the previous execution method is the involvement of the PyWPS server that allows the implementation of the WPS standard.

A. Creation of a new WPS process

The PyWPS is used for describing a standardized access method to the GreenLand operators. It uses Python scripts that allow HTTP invocation of remote services, and it consists of four sections (Algorithm 1):

Algorithm 1 PyWPS exemplification for the NDVI algorithm

```
 1: WPSProcess.__init__(self,
 2:       identifier = ”NDVI”,
 3:       title=”NDVI process”,
 4:       abstract=”Computes the NDVI index.”,
 5:       version = ”1.0”,
 6:       storeSupported = True,
 7:       statusSupported = True,
 8:       grassLocation = False)
 9: self.NIR = self.addLiteralInput(
10:       identifier=”nirBand”,
11:       type=type(””),
12:       title=”NIR band input;image/tif”)
13: self.Red = self.addLiteralInput(
14:       identifier=”redBand”,
15:       type=type(””),
16:       title=”Red band input;image/tif”)
17: self.Result = self.addLiteralOutput(
18:       identifier = ”result”,
19:       title=”Output result”,
20:       type=type(””))
21: data=urllib.urlopen(’http://cgis2ui.mediogrid.utcluj.ro/GreenL
22:       andv2/executeChain&process=”NDVI”&inputs=[{
23:       ”type”:”tif”, ”value”:”’+self.NIR.getValue()+’”},
24:       {”type”:”tif”,”value”:+ ”’+self.Red.getValue()+’”}]).
25:       read()
26:
27: self.Result.setValue(data)
```

- Header (lines 1-8): contains the information about the process, such as the unique identifier, the title and description, an attribute that indicates the usage of GRASS code within the script, etc.;
- Specification of inputs (lines 10-17): the name, description, and the type represent the minimal set of parameters that need to be specified for each input. There are two available types: *literalData* and *complexData*. The first one is used for sending numerical values, strings, or Boolean operators. The second type is used when the input requires the byte array of a local or a remote data source;
- Specification of the output (lines 18-21): has the same data structure as the input section;
- The body section (lines 23-29): depending on the value of the grassLocation parameter, this section contains GRASS functions, or a method that performs a remote service call to the GreenLand operators.

The Normalized Difference Vegetation Index (NDVI) is a common algorithm used in the classification of the land use/-land cover from different geographic areas. Its implementation is given in Algorithm 1, and it uses the following formula in order to perform the classification process:

$$NDVI = \frac{NIR - Red}{NIR + Red} \quad (1)$$

As can be seen there is a perfect match between this formula and the Python script defined in previous algorithm. The two inputs of the algorithm represent different types of satellite images, converted to GeoTIFF [24] by the GreenLand platform.

Usually, a satellite image contains several spectral bands. The NDVI algorithm uses the Red and NIR bands in order to create the classes of the land cover. The result of this operator generates a single-band satellite image that is available for the user to download and to perform further analysis.

After specifying the inputs and output, the Python script invokes a remote service (*executeChain*) that is hosted on the GreenLand platform. This service contains the actual implementation of the NDVI formula, while Algorithm 1 provides only a standardized access to this service. The invocation is done by using the HTTP protocol, and attaches the two parameters as additional arguments. It is worth mentioning that the *nirBand* and *redBand* identifiers store the full path to the physical resources that are going to be used at execution time.

B. WPS access types

The processes provided through the PyWPS server have three access modes: directly from the GreenLand application, API access from external platforms, and browser based access. The first mode assumes that the user is authenticated at the GreenLand level, and has access to all its geo-spatial operators. Within this platform, the user has the possibility to select one of these algorithms and to specify its inputs. After that it can establish a bidirectional communication channel with the PyWPS server (step 2 of Figure 1):

- PyWPS $\rightarrow$ GreenLand platform: the user is able to interrogate the list of existing processes (GetCapabilities) and to visualize the detailed description for each of them (DescribeProcess);
- GreenLand platform $\rightarrow$ PyWPS: in order to start a new processing over the Grid infrastructure, the user is able to invoke the Execute WPS operation. The inputs selected by the user have to be sent as additional parameters to the Python script that further passes them to the *executeChain* service that starts the Grid processing.

In some cases the inputs path to the physical resources involves a complex combination of folders and files. In such situations there is a high probability of introducing syntactical errors while specifying these paths manually.

The main advantage of accessing the WPS processes from the GreenLand platform is that it leaves no space for such errors. The user is provided with a list of aliases, instead of the real names of the satellite images. After selecting one input, the system generates in the background the entire path to that resource, and sends it to the PyWPS server.

The WPS processes could also be accessed directly from the browser (step 3 of Figure 1), by using its URL address. The main disadvantage is that the user has the full control in building the URL, including the paths to the input resources. Errors may occur in these cases, and the system is not able to provide specific support. This is the main reason why this paper recommends the usage of the first access method of the WPS services.

Using external platforms for invoking the PyWPS processes (step 4 of Figure 1) represent the third accessibility mode. It is similar with the GreenLand-WPS mechanism, but it lacks of some important features: the customized version of the metadata retrieved by using the GetCapabilities and DescribeProcess operations, the access to the GreenLand geospatial data repository, and the automatic generation of the input paths.

All accessing modes from Figure 1, are bidirectional. The connection to the PyWPS is used when invoking the Execute WPS operation, while the connection from the PyWPS is required to expose information about the available services (as metadata) by using the GetCapabilities and DescribeProcess operations.

C. WPS execution over the Grid infrastructure

After the PyWPS receives an Execute request with the proper inputs data set, it identifies the process and invokes the *executeChain* service that in his turn calls one of the algorithms (e.g. NDVI) available in the GreenLand repository. At this stage (step 5 from Figure 1) the next action is to use the gProcess platform in order to partition the geo-spatial algorithm into atomic tasks, and to schedule them to different Grid worker nodes.

At runtime, the process enters several stages: submitted, running, completed, and cancelled. The first stage allocates a specific number of Grid computing nodes, and sends the tasks together with their additional dependencies to these physical machines.

The running stage is identified as the actual Grid execution, where each node processes a task or a group of tasks. If the tasks are related between them, when an intermediate result is generated, it is used as input for other tasks.

When all intermediate results are available, the entire execution of the algorithm is completed. At this point, the user is able to access the results in a standardized manner, by using the WPS operations.

In cases in which errors occurred in the Grid based processing of the geo-spatial algorithms, the cancelled status is displayed. For such situations, the recommendation is to restart the execution.

In traditional WPS usage most of the algorithms need a relatively small amount of time to finish their executions (e.g. a few seconds). This allows the system to provide the WPS execution result almost instantaneously. On the other hand, the GreenLand platform addresses large scale use cases that require tens of minutes of Grid processing. Because of this aspect, a monitoring module was implemented at the gProcess level that periodically notifies the PyWPS server about the status of the execution.

The monitor module displays a ”Not executed” message or a valid URL from where the user is able to download the results. If the WPS in integrated within the GreenLand platform, this link is masked under the shape of a download button. In case of direct browser access the URL in display as plain text. When accessing the PyWPS services from external platforms, the metaphor of downloading the result should be customized by certain criteria.

VI. Experiments

This section describes the standardized execution of the Mosaic algorithm, exemplifying at each step the implementation details, from the GreenLand perspective. This proof of concept experiment is related to the validation of integrating the WPS standard within the Grid based execution processes, by using the GreenLand platform as an intermediate level. In conclusion, the experiment does not aim to measure the performance of the Grid processing, but only to validate the proposed solution.

A. Experiment description

The Mosaic is the core algorithm of the Black Sea catchment hydrologic modeling use case, which generates the shape of the entire geographic area of this catchment, by merging multiple tiles one to another. These tiles represent single-band satellite images of MODIS products (MOD15 and MOD16 in particular) that are extracted at runtime, based on the users' requests [3].

These products are multi-layers stacks of 1 km resolution provided on 8-days basis, where the maximum file size is 5.8 Mb for each. The spatial data is stored in large repositories, such as Numerical Terradynamic Simulation Group (NTSG) that offers remote access to the geo-spatial information, via File Transfer Protocol (FTP) protocol. New measurements are added periodically by means of satellite sensors that provide raw data that is pre-processed and then inserted into these repositories.

For an easier identification of these data, the NTSG consortium partitioned the Earth surface into horizontal and vertical tiles. For this experiment only the tiles that cover the Black Sea catchment are needed. The MODIS data is organized in years, starting from 2000 until the current time. Because it is an 8-days temporal resolution product, each year contains day-folders (e.g. D001, D009, D0017) indexed from 1 and increasing up to 361.

Taking into account all these aspects, the Mosaic implementation requires an automatic data retrieval script that transfers at runtime the relevant information from remote repositories to the Grid worker nodes.

B. Selecting the inputs data set

Before executing a WPS process, the user has to query the list of available services. This is possible by performing the GetCapabilities request that will return a XML metadata file. Due to the lack of space, Figure 2 highlights only a fraction of this document that contains the description of the processes. Among other details, the metadata contains information about the provider of these services and the Web addresses to the WPS operations.

In order to perform the experiment, the Mosaic process is going to be selected. The next step is to get a detailed description of its inputs and output, by using the DescribeProcess request. The ows:Identifier attribute is used as a unique identifier of the process.

```
<wps:ProcessOfferings>
    <wps:Process wps:processVersion="1.0">
        <ows:Identifier>NDVI</ows:Identifier>
        <ows:Title>NDVI process</ows:Title>
        <ows:Abstract>Computes the NDVI </ows:Abstract>
    </wps:Process>
    <wps:Process wps:processVersion="1.0">
        <ows:Identifier>Mosaic</ows:Identifier>
        <ows:Title>Mosaic process</ows:Title>
        <ows:Abstract>Computes the Mosaic</ows:Abstract>
    </wps:Process>

                          . . .
</wps:ProcessOfferings>
```

Fig. 2. GetCapabilities metadata file

```
<DataInputs>
    <Input minOccurs="1" maxOccurs="1">
        <ows:Identifier>year</ows:Identifier>
        <ows:Title>Year</ows:Title>
        <LiteralData>
            <ows:DataType ows:reference= "http://www.w3.
org/TR/xmlschema-2/#string"> string </ows:DataType>
            <ows:AnyValue/>
        </LiteralData>
    </Input>

                          . . .
</DataInputs>
```

Fig. 3. DescribeProcess metadata file for the Mosaic workflow

The product type (MOD15 or MOD16), a list of bands for each product (e.g. Evapo-transpiration - ET, Leaf Area Index - LAI), and the processing year (e.g. 2000, 2001, ..., until current year) are the input parameters that are requested by the WPS Mosaic process. For a better understanding of this action, Figure 3 highlights the structure only for the year input of the process.

The GreenLand platform is able to parse the DescribeProcess metadata, and to present it to the user in a more user friendly manner, by means of combo boxes, check boxes, buttons, dynamic text, etc.

The next action is to initiate the Grid execution, through the PyWPS server. This is possible by using the WPS Execute operation. The URL of this action is automatically generated in the background, based on the values selected by the user at the graphical level of the GreenLand application.

Let's assume that we want to process the LAI band from the MOD15 product, ET band of the MOD16 product, and set the processing year to 2010. The URL for the Execute operation has the following structure: http://<server_domain>/wps/wps.py?service=wps&version=1.0.0&identifier=Mosaic&request=Execute&datainputs=[year=2010;mod15=LAI;mod16=ET]. The <server_domain> represents the URL location of the PyWPS server, while the datainputs field contains all the parameters specified by the user.

C. Grid based execution

The Execute request is received by the PyWPS sever that interprets it, and assigns each parameter from the *datainputs* to local variables in the Python script. Such an example was described in Algorithm 1 for the NDVI process.

The Mosaic is a GRASS based algorithm that is resident on the GreenLand platform. Once the WPS execution request reaches the PyWPS server, it is redirected to this geo-spatial algorithm. Among the inputs list, additional information needs to be sent, such as: a unique identifier, a short description, the type required for each input together with its entry from the GreenLand database.

Until this stage, the Mosaic experiment was performed based on the rules and guidelines provided by the WPS standard. The next steps represent the actual Grid execution that is outside of the standard scope. This execution starts at the gProcess level that is responsible for partitioning the Mosaic algorithm into atomic tasks.

The one year processing contains multiple executions of data with a time delay of 8 days (the NTSG data repository updates every 8 days with new satellite images). So, in one year we have 365 days that divided by 8, results 45 executions. Because the user selected two MODIS products (MOD15 and MOD16) there will be 90 independent executions.

Allocating one Grid worker node for each task is not efficient, because its execution is not that time consuming. In order to optimize the entire processing flow, the gProcess platform creates groups of nine tasks, where each group is going to be executed on a single worker node.

After scheduling the entire Mosaic process, transferring the data to each Grid machine is the next step. Initially, data is resident on remote repositories and needs to be copied at runtime to the worker nodes. These repositories contain horizontal and vertical data tiles that cover the entire Earth surface. For this experiment only 12 tiles (the ones from the Black Sea catchment area) are needed.

Each of the 90 executions will process all 12 tiles, where their results will provide a good indicator for hydrologic prediction in the Black Sea catchment region.

Usually, the Mosaic process requires two hours to complete its execution. During this time, the user needs to know the status of the Grid execution. The gProcess platform provides a monitor mechanism that periodically interrogates the execution state, and sends the feedback to the GreenLand platform that in its turn displays it to the user in an easy to understand manner.

When the Grid execution completes, the user is able to download the results or to perform further operations. Figure 4 presents one of the results generated by processing the Mosaic algorithm that is partitioned into 12 tiles that cover the Black Sea catchment geographic area.

VII. Conclusions

This paper aims to integrate the WPS standard for executing the GreenLand algorithms over the Grid infrastructure. The Mosaic experiment proofs that this approach is valid and could be extended for other types of algorithms.

The other two objectives (different WPS access modes and the flexible implementation of WPS processes) were also described, by highlighting the possible implementation solutions.

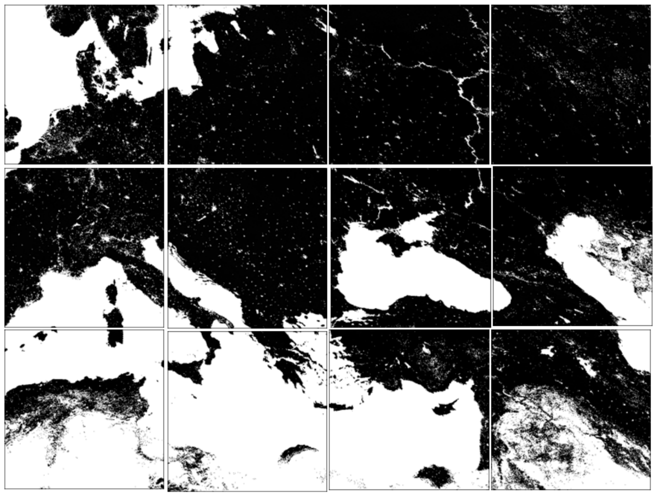

Fig. 4. The result generated by the Mosaic operator

The OGC consortium does not offer enough implementation details for all situations. For example, applying the WPS standard in Grid computing is difficult due to the fact that the Grid infrastructure allows only authenticated users to perform different operations. One recommendation is to extend the current standard to also enhance these use cases.

Acknowledgment

This research is supported by the enviroGRIDS Project funded by the European Commission, through the Contract 226740.

References

[1] D. Mihon, V. Colceriu, F.B. Balcik, K. Allenbach, M. Gvilava, D. Gorgan, "Spatial Data Exploring by Satellite Image Distributed Processing", Geophysical Research Abstracts, EGU General Assembly 2012, vol.14, pp.13278, 2012.

[2] enviroGRIDS project, http://envirogrids.net/

[3] B.F. Bektas, C. Goksel, S. Sozen, K. Allenbach, M. Gvilava, K. Rahman, D. Gorgan, D. Mihon, "Remote Sensing Services - ESIP Platform and Hot Spot Inventory Case Studies", enviroGRIDS Deliverable D2.11, 2012. Available at: http://envirogrids.net/index.php?option=com_jdownloads&Itemid=13 &view=finish&cid=139&catid=11

[4] Open Geospatial Consortium, (2007), "OpenGIS Web Service Common Implementation Specification", pp.1-153.

[5] L. Diaz, C. Granell, M. Gould, "Case study: Geospatial Processing Services for Web- based Hydrological Application", Book chapter: Geospatial Services and Applications for the Internet, pp.31-47, 2008.

[6] J. Cepicky, "PyWPS 2.0.0: The Presence and the Future", Geoinformatics, 2007, http://geoinformatics.fsv.cvut.cz/gwiki/PyWPS_2.0.0:_The_presence_and _the_future

[7] G. Aloisio, M. Cafaro, "A Dynamic Earth Observation System", Parallel Computing, vol.29, pp.1357-1362, 2003.

[8] D. Gorgan, V. Bacu,T. Stefanut, D. Rodila, D. Mihon, "Grid Based Satellite Image Processing Platform for Earth Observation Application Development", Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, pp.247-252, 2009.

[9] D. Gorgan, V. Bacu, D. Mihon, D. Rodila, T. Stefanut, K. Abbaspour, P. Cau, G. Giuliani, N. Ray, A. Lehmann, "Spatial Data Processing Tools and Applications for Black Sea Catchment Region", International Journal of Computing, vol.11 (4), pp. 327-335, 2012.

[10] Y. Wang, S. Wang, D. Zhou, "Retrieving and Indexing Spatial Data in the Cloud Computing Environment", 1st International Conference on Cloud Computing, pp.322-331, 2009.

[11] G. Giuliani, S. Nativi, A. Lehmann, N. Ray, "WPS Mediation: an Approach to Process Geospatial Data on Different Computing Backends", Computers and Geosciences, vol.47, pp.20-33, 2011.

[12] Z. Chen, N. Chen, C. Yang, L. Di, "Cloud Computing Enabled Web Processing Service for Earth Observation Data Processing", IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing, vol.5, pp.1637-1649, 2012.

[13] A. Wytzisk, M. Gould, F. Holsmuller, "Research and Instruction Mixing Commercial and Open Source Tools", 14th Workshop AGILE International Conference on Geographic Information Science, 2011.

[14] Degree Webservices documentation, 2013, http://download.deegree.org/documentation/3.2-pre13/deegreeWebservices.pdf

[15] Sextante documentation, 2011, http://gvsigce.sourceforge.net/sextante_web/pdf/SextantePaper.pdf

[16] M. Neteler, H. Mitasova, "Open Source GIS: A GRASS GIS Approach", Kluwer Academic Publishers/Springer, Boston, Dordrecht, and London, 2004.

[17] D. Wesloh, J. Sturley, "Implementing ISO Data Quality Standards Using ESRI's GIS Data ReViewer", National Geospatial-Intelligence Agency, 2004.

[18] FGDC Facilities Working Group, "Spatial Data Transfer Standard (SDTS). Part 7: CADD Profile", Federal Geographic Data Committee, 2000.

[19] OASIS standard, http://www.oasisopen.org

[20] J. Brauner, T. Foerster, S. Bastian, B. Bastian, "Towards a Research Agenda for Geoprocessing Services", 12th AGILE International Conference on Geographic Information Science, Hannover, Germany, pp.2-12, 2009.

[21] Z. Young, I. Raicu, S. Lu, "Cloud Computing and Grid Computing 360-Degree Compared", Grid Computing Environments Workshop, pp.1-10, 2008.

[22] V. Bacu, T. Stefanut, D. Rodila, D. Gorgan, "Process Description Graph Composition by gProcess Platform", HiPerGRID - 3rd International Workshop on High Performance Grid Middleware, Proceedings of CSCS-17 Conference, vol.2., pp. 423-430, 2009.

[23] T. Mitchell, "Web Mapping Illustrated", O'Reilly, First edition, ISBN 978-0-596-00865-9, pp.349, 2005.

[24] J. Rhodes, C. Bailey, P. Brown, "FalconView GeoTIFF Profile", Georgia Teach Research Institute, pp.1-25, 2006.