Mathematical Modeling of Distributed Image Processing Algorithms

Vlad Colceriu, Danut Mihon,
Angela Minculescu
{vlad.colceriu, vasile.mihon}@cs.utcluj.ro
angela.minculescu@gmail.com
Technical University of Cluj-Napoca
Cluj-Napoca, Romania

Victor Bacu, Denisa Rodila
Dorian Gorgan
{victor.bacu,denisa.rodila}@cs.utcluj.ro
dorian.gorgan@cs.utcluj.ro
Technical University of Cluj-Napoca
Cluj-Napoca, Romania

***Abstract*—Satellite images play an important role in developing Geographical Information System software applications that prove to be useful for different Earth Science phenomena analysis. Accurate results are obtained from high resolution images, or by applying the same algorithm multiple times over a specific input data set. In both cases the data volume that needs to be processed is large, and usually involves distributed infrastructures. In order for non-technical users to use these algorithms, they should be described in a flexible manner, using workflow structure models. This paper highlights the main achievements within the GreenLand platform, regarding scheduling, executing, and monitoring the Grid processes. Its development is based on simple, but powerful, notion of mathematical directed acyclic graphs that are used in parallel and distributed executions over the Grid infrastructure.**

I. Introduction

This paper highlights the parallel and distributed satellite image processing over the Grid infrastructure, as implemented within the GreenLand platform. GreenLand is a free GIS (Geographical Information System) software used in the geospatial data management and visualization domain, which was integrated as part of the BSC-OS(Black Sea Catchment-Observation System) portal[1][2] alongside other software platforms, designed for calibration of SWAT models, such as gSwat[3] and BASHYT[4] and other general purpose GIS web applications, such as GeoServer[5] and GEOSS[6]. The following sections present some of the main goals of this system: provide a flexible description of spatial data processing, schedule, execute and monitor Grid processes, GRASS (Geographic Resources Analysis Support System) [7] library integration, and interoperability with other software platforms.

All the executable processes implement a specific functionality, related to the Earth Science domains: satellite images data extraction, thematic map creation, arithmetic operations on spatial data, raster and vector data conversion, etc. All these processes are represented within the GreenLand platform as acyclic graphs, composed from basic operators, Web services and sub-graphs [15].

The operators are identified as atomic components and represent the smallest unit of work that can be executed without further decomposition. The workflow is another GreenLand concept, used to fulfill the user needs. It could be defined as a collection of basic operators, adopting a graph-style representation. Each node implements a particular function, while the entire workflow can be used to simulate specific dataflow scenarios.

The availability of the GreenLand system for non-technical persons was the main reason for workflow based data representation. Otherwise they should have been familiar with the XML standard and with developing Linux based scripts. In order to ease the user actions, two editor tools were implemented for operator and workflow description. Another advantage of using this approach is the portability within other platforms, as described in section System related architecture.

The Grid infrastructure processing capabilities are needed due to the large volume of satellite data that could reach a few GB is size. Executing such data is a complex process and should be optimized even when executed over the Grid worker nodes. Some workflows executions are light weight, while other might take hours to complete. This way it is up to the gProcess platform [8] to apply the best scheduling techniques. Currently no solutions exist to overcome this shortcoming, but several research directions have already analyzed and put into practice[9].

The gProcess platform is used for Grid process schedule, execution and monitoring. More information about the operations performed by this platform can be found in section entitled Grid based execution.

II. Related works

The Grid processes are described using the mathematical graph concept that seems to fulfill the GreenLand requirements of extensibility and simplicity . The major disadvantage in using such a method is represented by the cyclic workflows that handle looping execution. This is a restrictive case in the GreenLand workflows editor, and the user has no possibility to define such kinds of structures. There are several applications that could be used to create workflows: Pegasus [10], Taverna [11], GridFlow [12], etc. All of these are working only with acyclic graphs, called DAG (Direct Acyclic Graph). The main difference between these tools and the OperatorEditor and WorkflowEditor, developed within the GreenLand platform, is related to the flexibility in managing the data structure, the possibility of creating hyper-graphs, depth workflow navigation, or ease in creating new basic operators by attaching a specific functionality (described throughout an executable file, script file, Web service, etc.).

Most of the GreenLand operators encapsulate GRASS functionalities that operate with raster or vector data formats. The GRASS library allows the usage of more than 300 operators, supports over 2500 different CRS (Coordinate Reference System) and handles the most common used spatial data types: Landsat, MODIS, GeoTIFF, ESRI shapefiles, etc. Due to its popularity, there are several geospatial applications that integrate this library: Sextante [13] and QGIS (Quantum GIS) [14].

The main goal of Sextante is to provide an easy method for implementing rich geo-processing algorithms, and it integrates tools like Java GIS, OpenJUMP, ArcGIS, etc. QGIS allows the user the possibility to execute geospatial data, to analyze the results, edit raster and vector data, data type conversion, etc.

One of the main goals of the GreenLand platform is to provide workflows that could be reused in other applications, such as Pegasus, Taverna, PGRADE [15], etc. This could be achieved by using the SHIWA (SHaring Interoperable Workflows for large-scale scientific simulations on Available DCIs) [16] platform that offers interoperability services in order to standardize the workflow development and portability.

Workflow interoperability enables their execution over different infrastructures, allows data sharing among scientific communities around the world, facilitates workflows migration between applications, and offers the usage of the most appropriate system or infrastructure in order to execute one specific workflow.

In order to access GRASS functions, the user has to write its own Linux bash script, in the Sextante and QGIS frameworks. On the other hand the GreenLand offers the user the possibility to do the same operations but in a more intuitive manner, by using the workflow editor. This approach allows the non-technical users to develop and process their own scenarios, without the uncertainty of introducing semantic or syntactic errors.

The GreenLand uses the gProcess platform in order to schedule, execute and monitor processes over the Grid infrastructure. Other approaches that share the same experience regard the GANGA [17] and Diane (Distributed Analysis Environment) [18] tools. Grid process configuration and monitoring is based on the GANGA tool, while the execution scheduling and task submission is related to the Diane application

III. System related architecture

GreenLand is a client-server application, available over the Web. The client-side represents the graphical user interface that fulfills user requests for a extensible, parallel running and internet accessible GIS platform. The server-side is Java based and implements functionalities for users, projects and data management. Data exchange between these two modules is based on Web services.

The only way for the user to access the backend functionality of the GreenLand application is through its graphical interface (Figure 1). A username and password authentication is required for system access.

The second architectural level consists of a set of services exposed by the GreenLand platform: users management, workflows development, execution and management, data retrieval,

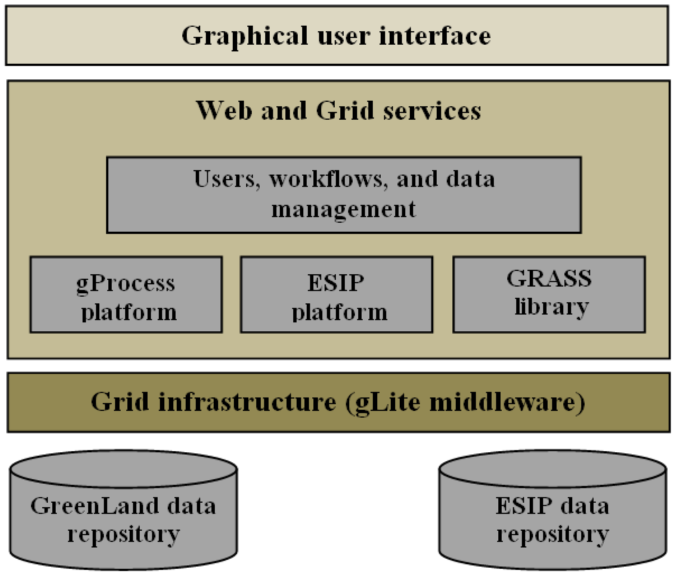


Figure 1. System related architecture

data storage, data conversion, etc. These services are available by integrating the gProcess and ESIP (Environment oriented Satellite Data Processing Platform) [19] platforms. The Web services provided by the gProcess fulfil the user requirements regarding the process scheduling, execution and monitoring.

The workflows developed by the users have two internal standard representations, both of them using the XML description. The first one is called PDG (Process Description Graph) and it is a pattern that describes only the workflow nodes types and position, and the relationship between them, but it has no knowledge about its physical inputs and outputs. This pattern is only used to store the workflow representation, and it expands during the Grid execution into a so called iPDG (instantiated PDG). This second representation shares the same XML structure as the PDG, and allows the gProcess to gather all the inputs information specified by the user (e.g. spatial data files, numerical constants, external dependencies, etc).

Based on the iPDG format, the gProcess platform performs the Grid scheduling operation. In most cases a single node in the workflow will be processed on a single CPU, but there are situations in which groups must be created in order to improve the execution efficiency. Currently this is not an automated process, because it requires a complexity analysis of the entire workflow. Several research studies were conducted in this direction, and the bases for such a module were already adopted.

The gProcess platform establishes the connection with Grid infrastructure, by implementing a subset of the gLite middleware. These services allows the data transfer (i.e. input data specified by the user) to SE (Storing Element), tasks execution over CE (Computing Element), proxy creation and delegation, Grid execution information retrieval, etc.

The ESIP platform are a set of Web services that provide the following functionalities: basic operators and workflows development, workflow representation based on DAG (Direct Acyclic Graph) patterns, spatial data management, etc. Internal representations of the basic operators are also part of the ESIP platform, exposed as: vegetation indices (e.g. NDVI, EVI), spatial data processes (e.g. mosaic, density slicing, and data extraction), statistics (e.g. histogram generation, standard deviation computation), etc.

Other services provided by the GreenLand platform are related to users management (i.e. create new account, update profile, etc.), data retrieval using the local upload mechanism, FTP data transfer and OGC services [20].

Finally it is worth mentioning that the current application stack is enrolled within the envirogrids.vo-eu.egee.org Virtual Organization, of which for testing purposes we used the sites or computing elements: RO-09-UTCN and AM-02-SEUA.

IV. Data model

This section describes the basic operator, workflow and project concepts, their development using the GreenLand editor tools as well as their internal representation within the ESIP data repository (Figure 1).

A. Project and workflow relationship

GreenLand projects are defined as virtual containers that allow workflow organization and instantiation. Each project has a unique name in the user workspace, and supports workflows attachment. A workflow can be added as multiple instances within the same project. At graphical user interface level, the project content is displayed as a forest of trees, where each tree root represents the workflow name, and leafs consists of the workflow instances. Each item inside the project, stores information about its name, description, author who developed it, inputs and outputs, etc.

From the graphical interface the user is able to specify the physical inputs for this item (workflow). For each input, only the available values are displayed to the user (e.g. if the inputs type requires a spatial data attribute, only the list of available satellite images are shown). All these information are retrieved based on ESIP services.

Executing a project consists of processing its entire list of workflows. This operation is achieved by using the gProcess services. After the Grid process begins, a monitoring mechanism gives feedback about execution progress.

B. Basic Operator Concept

Operators lie at the center of the gProcess execution environment and GreenLand management system. They represent the basic units of work, the only constructs which can get executed.

The GreenLand application allows users to create, alter and delete these structures. By doing so, it allows full customization of the Grid execution processes, from its most coarse grained constructs represented by iPDGs to its most simple, atomically executed statements.

Operators represent the most fine grained execution units; they are the only constructs that get executed on the nodes of the Grid. These units must have their respective program or executable script defined as well as any dependencies they might require, since environment in which they run is heterogeneous and offers no guarantees on shared library or version.

Table I. Operator editing conditions

	Operator is owned		Operator is used		Operator is validated	
	True	False	True	False	True	False
Insert	N/A	N/A	N/A	N/A	N/A	N/A
Update	Yes	No	Partial	Yes	Yes	Partial
Delete	Yes	No	Partial	Yes	Yes	Yes

The insertion of operators is supported via a visual editor which takes the users program and annotations and inserts it in the gProcess and GreenLand databases.

When creating an operator one has to provide besides the executable code of the program, certain additional information, which allows the GreenLand application to track the visibility, unique name, description and category of the operator.

There are two types of visibility properties defined:

- Public means that all the users may view and use the operator.
- Private means that only the owner of the operator may view or use it.

The public operators, to which a user is not owner to, but uses within its private or public workflows, can still be accessed even if the visibility of the program in question is changed. However creation of new graphs containing that element is prohibited.

The category allows the user to create its own hierarchy of operators, facilitating a quicker lookup when browsing for them.

An Application Programming Interface (API) has been created to allow the user to create operators. The problem with it is that if reuse is desired, the implementer would have to create a new program form scratch or call its desired application from within the provided ESIP (Environmental Satellite Image Processing) API.

Entering, updating and deleting Operators is not a straight forward operation, since there are some constraints involved in it as expressed in (Table I).

The first of these limitations refers to ownership of the operator, since there is a strict traceability of Grid execution which needs to be maintained. The idea is that each user should be responsible for its own distributed application. Additionally, before such an operator is made visible, it is tested locally for compliance, so that any malicious or unintended effects of the program may be detected.

The second limitation refers to whether the operator to be removed or updated is already in use. If it is used, removal and updating is done only at a formal level; else it is removed entirely from the database of operators. Deleting or altering an operator at a formal level means that any of the existing workflows which use it, can do so without becoming invalid or having their functionality changed.

Finally before moving on to workflows and hypergraphs the programming interface is discussed. It is implemented in using only the Java programming language, which brings up certain constraints regarding the generality of the platform. Of course one can call a program or script implemented in any programming or scripting language from the required Java wrapper, as long as it is supported by the operating system on the worker nodes of the Grid. Where the current worker node distributions include a CentOS version of the Linux kernel.

Also to be noted is the fact that all operators must be implemented in such a way so as to be able to parse Linux type paths, end of line characters and call executables which were compiled in Linux, preferably having all their library dependencies packaged alongside themselves.

Further constraints on the program include aspects of code structure such as [21]:

- Including the Operator class in a certain package "gPOperators"
- Extending a certain class, which includes the code for launching the operator on the Grid node "OperatorExec"
- Overriding a certain method included in the "OperatorExec" class

All these limitations exist due to the fact that these operators need to be integrated inside the gProcess platform, which was not designed to support such rich and powerful interaction as exposed by the GreenLand application.

This programming interface also includes all the dependencies and prerequisites needed for generating GRASS and GDAL based programs as described in section V-B. In order to do this a different class needs to be extended "GenericOperator" and a different method overridden "grassExecute".

C. Workflow and Hypergraphs Concept

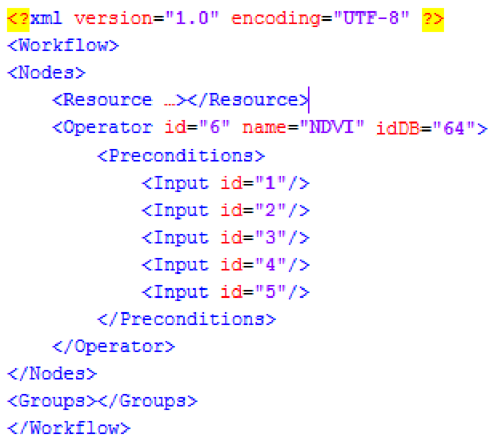
gProcess and GreenLand give users the opportunity to develop their own parallel and distributed programs. These are implemented with the help of Process Description Graphs (PDG), which plainly put are directed acyclic graphs.

Describing programs with the help of graphs is not a new concept; it has been extensively studied within [16] which presents a general solution to integrate already existing platforms together. It is also present in other well established frameworks for Grid execution such as [22] and [23].

PDG's cannot be executed on the gProcess platform since they represent only the program definition; they lack the input data necessary to perform useful actions. For execution we use another construct called Instantiated Process Description Graphs (iPDG).

iPDG's are morphologically similar with their counterparts but they give the possibility to specify user input to the defined program.

Both PDG's and iPDG's may also be referred to as workflows, since they present the flow of data, from node to node, in a Grid program.



```
<?xml version="1.0" encoding="UTF-8" ?>
<Workflow>
<Nodes>
    <Resource …></Resource>
    <Operator id="6" name="NDVI" idDB="64">
        <Preconditions>
            <Input id="1"/>
            <Input id="2"/>
            <Input id="3"/>
            <Input id="4"/>
            <Input id="5"/>
        </Preconditions>
    </Operator>
</Nodes>
<Groups></Groups>
</Workflow>
```

Figure 2. Simple PDG representing an NDVI program

The internal structure of a PDG is represented by nodes and directed edges. The nodes can be matched to operators or other PDG's. These particular types of entities, which do not make the scope of the top level structure are called sub-workflows and are similar to the idea of functions in programming languages. A structure which has multiple levels of imbrication is called a hypergraph.

Recursive structures are not supported within workflows since there is no control structures currently implemented within workflows. The reason they are not supported is due to the fact that no control structures have been implemented.

Control statements would allow the distributed program to test for termination conditions, otherwise not encountered in the current solution.

The arcs described inside a PDG and iPDG represent the flow of data. All information passed from a source node to destination passes trough gProcess file system, where it is forwarded to the corresponding execution, as specified in the workflow.

The constraints and operations presented for nodes also apply here. The major difference is that workflows are automatically created once such a request is submitted and require no additional validation of their behavior. One may assume that their behavior is implicitly safe since all their individual parts function correctly. We can make this assumption because it is only the operators that get directly executed.

gProcess and GreenLand have different representations of these two notions. gProcess uses a lightweight XML representation (Figure 2) of the directed acyclic graph.

The XML format is disadvantageous in allowing for an editable and extensible program structure mainly because of the fact that the user must specify the inputs and be able to validate the program structure manually. This means that it would need intimate knowledge about application structure. Such a solution would be impractical and furthermore unsafe since it would give the user direct access to resources, without any possibility to restrict or refute its actions.

On the other hand GreenLand allows for a database representation of the model, which gives the user the possibility to dynamically create and modify workflows, without having to know anything about internal representation. The model described was created so as to serve to the purpose of categorizing, extending and validating the workflows and their subcomponents.

The basic concepts behind the GreenLand application is the gProcess workflow, which is represented by a directed acyclic graph also called a PDG. In this graph each node represents the executable code submitted on a worker node, an operator. On the other hand an arc represents a communication path between two operators. They are not explicitly modelled since they can be inferred from the connection between two node(Figures 2 and 3).

The GreenLand data model supports ranking of operators according to categories in order make searching for a given functionality easier. Atop of this each category element offers the possibility to generate other subcategories(Figure 3), thus generating a infinitely extendible structure.

Each node of a workflow can be either a operator or another workflow, generating a multi-layered structure, inside which no cycles or self-calling elements can exist.

Additionally resources in the form of inputs and outputs are attached to a node. The amount of inputs or outputs a node may contain is unlimited, except for the case of operators, which may contain at most a single output. This constraint is imposed by gProcess functionality, which requires this in order to be able to detect operator output and communicate results between the nodes of the program graph.

Each resource supports either a string value or a file type. In order to assure that these elements are matched correctly, two types of validations need to be performed.

First a syntactic validation assuring that the file is of the required type. This validation is not done by filtering the file through a extension sieve, but by pre-emptively inspecting the file type at import time.

The second type of validation is done at the semantic level, where each file is checked so that the meta-data attached does not have conflicting values. An example of this would be the projection of the files, which according to GRASS and GDAL operators would have to be the same in order to obtain a successful execution.

Additionally it is worth mentioning that Greenland is accompanied by an interface application, which allows the user to interactively manipulate workflows, as easily as one would create, update and delete an operator [24].

V. Grid based execution

This Section presents the gProcess and GreenLand in intimate detail, highlighting their interfaces and communication protocols, which help the user to submit, create and manage distributed Grid programs.

A. GreenLand and gProcess Compatibility

GreenLand and gProcess are a pair of symbiotic applications designed to complement each other and in some cases of degraded functionality even work independently. The current implementation however requires that both applications be housed by the same machine.

GreenLand is a workflow, operator and file manager which allows the user to generate, edit and categorize Grid programs. On the other hand gProcess is a Grid execution manager , which allows the submission and cancelling of complex execution workflows.The task scheduler implemented in gProcess was also studied in [25].

Although they were thought with the idea of separability in mind, they still have to communicate with each other, to pass programs created in GreenLand to gProcess and to synchronize GreenLand data to gProcess executions.

As mentioned in Sect. IV-C, these two applications have different representations of PDG's and iPDG's. Where GreenLand has a recursive database hierarchy of operators and workflows, which contains additional information such as categories, descriptions and ownership information. Also the arcs and nodes of the graphs are represented as separate entities within the storage space. On the other hand gProcess has a lighter representation, where the entire program is contained within an executable file.

In order for things to work GreenLand must know the internal implementation of gProcess programs. This means that the GreenLand application must be able to create gProcess execution files. To do this it interrogates the gProcess database for all available operators and input types, which it uses to generate and validate its own programs.

gProcess offers services for uploading operators, workflows and required input files. These services are then called by GreenLand, so that the data edited within can become available to the Grid execution environment.

Execution and monitoring of workflows is the most important part of the GreenLand/gProcess communication and is divided in 4 distinct steps.

The first operation is the transfer of the iPDG file from the GreenLand application to gProcess. Even though both applications are housed by the same machine, they were designed to operate remotely. This is done by calling the "importXML" service of the gProcess application.

The second step requires that the file be registered as a PDG by calling "insertPDG" and then as a iPDG by calling "insertIPDG". This step is done on the same file, due to the similarities between the two file types.

After uploading the program, it is executed by calling the "execute" service, which returns information about monitoring identification number. This is then later used to single out the workflow, from within the set of monitored executions.

Monitoring is done at 2 different levels:

- Top level, which polls the execution in order to discover the state of the workflow
- Operator level, which inquires about the state of each node execution separately and extracts the output

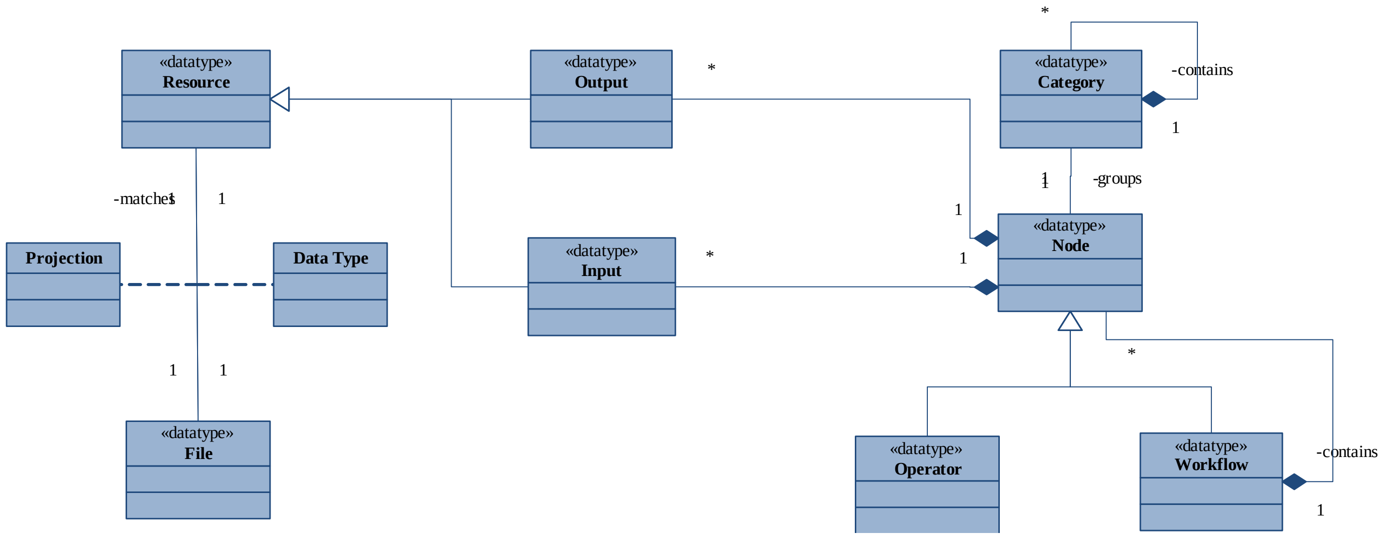


Figure 3. GreenLand Data Model

B. GRASS Integration

The operators developed for gProcess can be configured to run already existing applications. GRASS is one such case of a fully fledged desktop application, running on the Grid.

All programs which run on the Grid must not require any interactive user input. They must be applications which have non-interactive interfaces, meaning that all input must be known in advance.

The Grid platform exposes to its users a heterogeneous environment, where program versions and installed shared libraries can differ from system to system. The only constant we can count on is that the background operating system is running a Linux kernel. In such a case we cannot make any assumptions about whether an application which runs perfectly on a desktop environment will run in the same manner on all of the nodes.This means that in order to use GRASS there are certain steps which have to be performed before one can be sure of its functionality.

The primary condition that must be satisfied is that all executable and configuration files used by the operator be packaged with it as described in (Figure 4).

GRASS has a binary folder which contains all functions, which must be included in the operator dependencies. Also a configuration file specifying some of the parameters of the application, ex. *DATABASE*, *LOCATION_NAME* and *MAPSET*. More on this topic can be found in [7].

On a desktop solution the operating system will satisfy all needed shared libraries at install time. On the Grid platform an executing operator has limited privileges when writing files, accessing system state and installing programs. To compensate for this drawback all needed shared libraries were packaged with the operator.

Finally a script must be created, which generates the above mentioned configuration file and appends all executables to the *$PATH* system variable and prepends the shared libraries to the *$LD_LIBRARY_PATH* variable. The class that implements this functionality within the GreenLand programming interface is ”GrassGeneric”.

C. Grid Execution and Monitoring

Each program created by GreenLand is later executed, monitored and managed by gProcess. Once the former mentioned application is done creating and launching the workflow, the second jumps into action.

The executor service processes the iPDG description in order to accomplish workflow execution on the Grid [26], where it parses the XML file and generates the appropriate internal representation. It then tries to check the file for consistency by matching input and output types. The input data of one operator, service or resource must match the output of the node on the other end of the arc which links them.

The executor service also checks for consistency relating to the availability of the individual operators instantiated within the internal representation. If any of the operators are missing or unavailable the system tries to find an operator or service capable of substituting it, while also checking for cycles and recursive declarations. Doing so, it creates a planar structure, which is the expanded structure of the program.

When an internal representation has been created the backend application then submits each individual node of the workflow to a CE (Computing Element) of the Grid.

Once a workflow has been launched into execution, the hierarchies which existed within it are no longer visible. The user can only see the flattened, instantiated graph. This means that from the moment the workflow was launched, the monitoring can follow only the state of the entire structure and of individual operators, but not of intermediate structures.

Also canceling an entire workflow is supported, but not a singular node, since operators downstream might suffer from unsatisfied input constraints. This would require the system to cancel all dependent nodes, but since this would lead to results which would be hard to predict without having advanced knowledge of internal structure.

VI. Practical Use Case Scenarios

This section is divided into 2 subsections each detailing a different type of program generation mechanisms for gProcess corresponding to different levels of program abstraction.

The first use case will detail an operator, which was designed for merging a series of satellite images from a FTP repository into a single large image. Thus giving the user the possibility to select year, month and day of the given image and the region which required combining, without any prior knowledge of how the data had been organized on that particular repository, in order to obtain a single image of the entire Black Sea catchment area.

The reason for generating a new operator instead of a workflow was chosen due to the very particular functionalities of this use case, which could not be satisfied by other more general operators.

The second use case will detail a complex workflow generated, from a series of predefined operators. Where the requirement to be satisfied was the generation of a thematic map highlighting land use in the Istanbul metropolitan area.

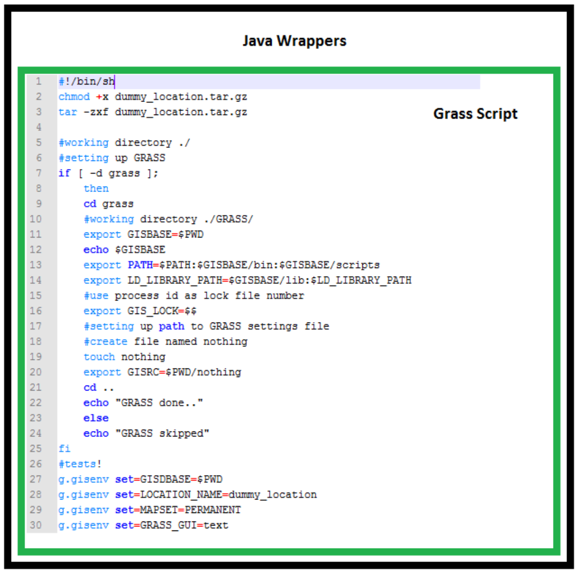
More information can be about the particularities of both these use cases can be found in document [27]

A. Mosaic Operator Use Case

This section presents the usage of a complex atomic structure within this framework. It gives an idea of how powerful and general the interface for Grid program generation really is.

The atomic operator is divided into several steps. The idea of atomicity is implemented under the paradigm of all or nothing execution. Meaning that if the operator fails, at one of the steps, no partial result will be available to the workflow.

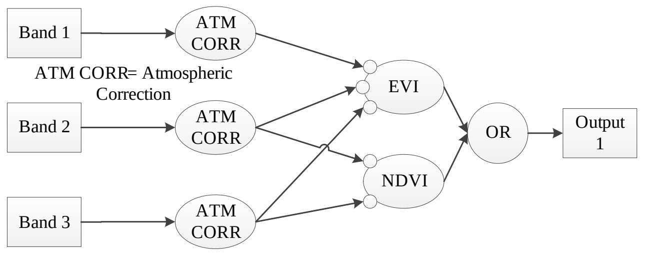
Inside the workflow there exist a list of operators allowing the user to generate a sequence of images representing a given time interval.



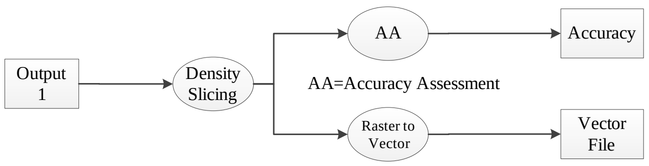
```
#!/bin/sh
chmod +x dummy_location.tar.gz
tar -zxf dummy_location.tar.gz

#working directory ./
#setting up GRASS
if [ -d grass ];
    then
    cd grass
    #working directory ./GRASS/
    export GISBASE=$PWD
    echo $GISBASE
    export PATH=$PATH:$GISBASE/bin:$GISBASE/scripts
    export LD_LIBRARY_PATH=$GISBASE/lib:$LD_LIBRARY_PATH
    #use process id as lock file number
    export GIS_LOCK=$$
    #setting up path to GRASS settings file
    #create file named nothing
    touch nothing
    export GISRC=$PWD/nothing
    cd ..
    echo "GRASS done.."
    else
    echo "GRASS skipped"
fi
#tests!
g.gisenv set=GISDBASE=$PWD
g.gisenv set=LOCATION_NAME=dummy_location
g.gisenv set=MAPSET=PERMANENT
g.gisenv set=GRASS_GUI=text
```

Figure 4. GRASS Operator Setup Script



(a) Vegetation index selection



(b) Thematic map generation

Figure 5. Workflows representing the Istanbul Thematic Map Use Case

The "Special Mosaic" operator takes multiple multiband images of various formats and glues them together according to certain metadata embedded within their corpus, which may refer to the projection of the individual bands, as well as the geographic region which they occupy. Such information provide the operator a way to combine the images.

The operator receives as its arguments the following: a link to an ftp server, a directory of that server plus username and password if necessary. The operator then decides which files to download given a specified algorithm.

The steps of the operator are divided as follows:

1) Download the images via ftp.
2) Split the images in their respective bands.
3) Combine each band from its parts.
4) Merge all results into a single image.

B. Istanbul Thematic Map Generation Use Case

In order to generate a thematic map for the Istanbul area from a given set of Landsat satellite images a series of operations needed to be performed.

Since the thematic maps are of land use in urban areas, the main operators of the workflows are those exposing vegetation indices, of which the current implementations opted for EVI and NDVI. Therefore the bands of the Landsat image being used are 1,3 and 4 corresponding to blue, red and infrared bands. Bands 3 and 4 are required for NDVI and 1,3 and 4 for EVI. Both algorithms return an image with values between -1 and 1, where values from -1 to 0 represent water bodies and 0 to 1 increasing values of vegetation.

Before the vegetation index operations can be performed, there is the need for atmospheric correction, which is based on metadata attached to the multi-band image and a series of mosaic and cropping operations, which are required due to the fact that the location of Istanbul is spread across 2 distinct Landsat images. Cropping and mosaicking are removed from figure 5 due to them not bringing any added value to the use case outside of solving a technical issue.

After applying one of the 2 vegetation indexes a density slicing algorithm is applied reducing the number of possible values of the resulting image from 256 floating point intervals to just 3 classes representing water, urban and wooded areas.

The last step of this algorithm is composed of an accuracy assessment operator and a Raster to Vector image converter, which guarantee that a sufficiently accurate thematic map represented by vector file is generated. If the accuracy is below a given threshold the workflow is executed again using different intervals for the 3 classes of the density slicing operator.

It is because of this fact that the implementation of this logical workflow has been divided into 2 parts so as to remove redundant work regarding atmospheric correction, mosaicking, cropping and vegetation index calculation (Figure 5).

VII. Conclusion

Due to the high complexity and size of input data satellite image processing requires high computing power. In order to be able to meet these requirements gProcess uses the Grid execution platform.

GreenLand extends the functionalities of gProcess by giving the user an interface with which he can customize his own programs from the coarse grained constructs represented by top level workflows to the most fine grained represented by operators.

Additionally to submission and management gProcess offers optimized execution and scheduling of multiple workflows so as to obtain the highest possible throughput.

Acknowledgment

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

References

[1] D. Gorgan, V. Bacu, D. Mihon, T. Stefanut, D. Rodila, P. Cau, K. Abbaspour, G. Giuliani, N. Ray, and A. Lehmann, "Software platform interoperability throughout envirogrids portal," *International Journal of Selected Topics in Applied Earth Observations and Remote Sensing –" JSTARS*, vol. 5, no. 6, pp. 1617–1627, 2012.

[2] D. Gorgan, V. Bacu, D. Mihon, D. Rodila, T. Stefanut, A. K., P. Cau, G. Giuliani, N. Ray, and A. Lehmann, "Spatial data processing tools and applications for black sea catchment region," *International Journal of Computing*, vol. 11, no. 4, pp. 327–335, 2012.

[3] D. Gorgan, V. Bacu, D. Mihon, D. Rodila, K. Abbaspour, and E. Rouholahnejad, "Grid based calibration of swat hydrological models," *Journal of Nat. Hazards Earth Syst. Sci.*, vol. 12, no. 7, pp. 2411–2423, 2012.

[4] P. Cau, C. Meloni, S. Manca, D. Soru, and D. Muroni, "A java based framework optimized for scientific modeling and analysis," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2011.

[5] J. Deoliveira, "Geoserver: uniting the geoweb and spatial data infrastructures," in *Proceedings of the 10th International Conference for Spatial Data Infrastructure, St. Augustine, Trinidad*, 2008.

[6] M. L. Butterfield, J. S. Pearlman, and S. C. Vickroy, "A system-of-systems engineering GEOSS: Architectural approach," *Systems Journal, IEEE*, vol. 2, no. 3, pp. 321–332, 2008.

[7] M. Neteler, M.H. Bowman, M. Landa, and M. Metz, *GRASS GIS: a multi-purpose Open Source GIS*, Environmental Modelling and Software, vol.31, pp.124-130, 2012.

[8] V. Bacu, T. Stefanut, D. Rodila, D. Mihon, and D. Gorgan, *Process Description Graph Composition by gProcess Platform*, HiPerGRID, May 28, Bucharest, vol.2, pp.423-430, 2009.

[9] V. Colceriu and D. Gorgan, "Execution time estimating framework on distributed platforms,", 2013, Unpublished.

[10] J.S. Vockler, G. Juve, E. Deelman, M. Rynge, and G.B. Berriman, *Experiences Using Cloud Computing for a Scientific Workflow Application*, ScienceCloud'11, pp.15-24, 2011.

[11] W. Tan, P. Missier, I. Foster, R. Madduri, D. De Roure, and C. Goble, *A comparison of using Taverna and BPEL in building scientific workflows: the case of caGrid*, Concurrency and Computation: Practice and Experience, vol. 22, pp.1098-1117, 2010.

[12] J. Cao, S.A. Jarvis, S. Saini, and G.R. Nudd, *GridFlow: Workflow Management for Grid Computing*, In 3rd International Symposium on Cluster Computing and the Grid (CCGrid), IEEE CS Press, May 12-15, Tokyo, Japan, pp.198-205, 2003.

[13] V. Olaya, *Sextante User's Manual*, 2011.

[14] T. Sutton, O. Dassau, and M. Sutton, *Geographical Information System User Guide*, Open Source Geospatial Foundation Project, 2011.

[15] P. Kacsuk, *P-GRADE Portal Family for Grid Infrastructures*, Concurrency and Computation: Practice and Experience, vol.23, pp.235-245, 2011.

[16] N. Cerezo, and J. Montagnat, *Scientifc Workflows Reuse through Conceptual Workflows on the Virtual Imaging Platform*, Proceedings of 6th WORKS2011, Seattle, pp.1-10, 2011.

[17] J.T. Moscicki, F. Brochu, J. Ebke, U. Egede, J. Elmsheuser, K. Harrison, R.W.L. Jones, H.C. Lee, D. Liko, A. Maier, A. Muraru, G.N. Patrick, K. Pajchel, W. Reece, B.H. Samset, M.W. Slater, A. Soroko, C.L. Tan, D.C. van der Ster, and M. Williams, *Ganga: A tool for Computational-task Management and Easy Access to Grid Resources*, Computer Physics Communications, vol. 180, pp.2303-2316, 2009.

[18] J.T. Moscicki, *DIANE - Distributed Analysis Environment for GRID-enabled Simulation and Analysis of Physics Data*, Nuclear Science Symposium, vol. 3, pp.1617-1620, 2004.

[19] V. Bacu, D. Rodila, D. Mihon, T. Stefanut, and D. Gorgan, *Error prevention and recovery mechanisms in the ESIP platform*, IEEE 6th International Conference on Intelligent Computer Communication and Processing, ICCP2010, pp.411-417, 2010.

[20] A. Padberg, and K. Greve, *Gridification of the OGC Web Processing Service: Challenges and Potential*, AGILE Workshop, pp.5-11, 2009.

[21] V. Colceriu and D. Mihon, *Operator Editor*, 2012. [Online]. Available: http://cgis.utcluj.ro/documents/OperatorEditor_user_manual.pdf

[22] P. Kacsuk, T. Fahringer, Z. Nemeth. *Distributed and Parallel Systems. Cluster and Grid Computing* , 2nd edition, 223 pages, Springer Verlag, ISBN: 0387698574 (2007)

[23] E. Deelman, G. Singh, M.H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, D. S. Katz, *Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems*, Scientific Programming Journal, Vol 13(3), 2005, Pages 219-237

[24] D. Mihon, A. Minculescu, V. Colceriu, and D. Gorgan, "Diagramatic description of distributed spatial data processing," *Romanian Journal of Human - Computer Interaction*,pp. 129-134, 2013.

[25] Pop F., *A Fault Tolerant Decentralized Scheduling in Large Scale Distributed Systems, chapter in Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies, and Applications*, N. Antonoupoulos, G. Exarchakos, M. Li, A. Liotta (Eds.), Ed. Information Science Reference (IGI Global), ISBN: 978-161-520-686-5, pp. 566-588, February 2010

[26] Gorgan D., Bacu V., Stefanut T., Rodila D., Mihon D., *Grid based Satellite Image Processing Platform for Earth Observation Applications Development.* IDAACS'2009 - IEEE Fifth International Workshop on "Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications", 21-23 September, Cosenza, Italy, IEEE, Computer Press, ISBN: 978-1-4244-4901-9, 247-252 (2009).

[27] F. B. Balcik, C. Goksel, K. Allenbach, M. Gvilava, K. Rahman, D. Gorgan, and V. Mihon, *Building Capacity for a Black Sea Catchment Observation and Assessment supporting Sustainable Development*, 2012.