Earth Observation Data Processing in Distributed Systems

Petcu Dana, Panica Silviu, Neagul Marian, Frîncu Marc and Zaharie Daniela,
Ciorba Radu, Diniş Adrian
Computer Science Department, West University of Timişoara, Romania
E-mail: gisheo-uvt@lists.info.uvt.ro

Keywords: distributed computing, earth observation, data processing

Received: February 15, 2010

Earth observation systems have a continuous growth in the user and internal requirements that can be handled nowadays only using distributed systems. These requirements are shortly reviewed in this paper. Huge data-sets management and processing are of special interest, as well as the particularities of the Earth observation data. On the technological side, the focus is put on service-oriented architectures that are facilitating the linkage of data or resources and processing. As proof of concept of current distributed system capabilities, the technological solutions used to build a training platform for Earth observation data processing are exposed and discussed in details.

Povzetek: S pomoèjo distribuiranega sistema je realiziran opazovalni sistem Zemlje.

1 Introduction

The paradigm known as Data Intensive Science [7] is currently changing the way research and innovation is being conducted. This paradigm is based on access and analysis of large amounts of existing or new data that were or are created not only by scientific instruments and computers, but also by processing and collating existing archived data. Earth observation systems, in particular, are gathering daily large amounts of information about the planet and are nowadays intensively used to monitor and assess the status of the natural and built environments.

Earth observation (EO) is most often referring to satellite imagery or satellite remote sensing, the result of sensing process being an image or a map. Remote sensing refers to receiving and measuring reflected or emitted radiation from different parts of the electromagnetic spectrum. Remote sensing systems involve not only the collection of the data, but also their processing and distribution. The rate of increase in the remote sensing data volume is continuously growing. Moreover, the number of users and applications is also increasing and the data and resource sharing became a key issue in remote sensing systems. Furthermore, EO scientists are often hindered by difficulties locating and accessing the data and services. These needs lead to a shift in the design of remote sensing systems from centralized environments towards wide-area distributed environments that allow a scale-out in a wide range of issues from real-time access to enormous quantities of data to experimental repeatability through the use of workflows. The underlying technologies used in service-oriented architectures, either Web, Grid or Cloud based, are facilitating this transition as well the linkage of data, resources, and processing.

In this context, the paper starts with a survey of the current requests imposed on distributed systems and coming from remote sensing application field. This survey is based on several recent research reports of EO and distributed systems communities and it is an extended version of [19]. A deeper look is dedicated to the Grid usage benefits for EO through techniques like bringing computations to the data, rather than data to the computations. Furthermore, a snapshot of the requests that can be satisfied by the current technologies is provided through a case study on a newly proposed service-based system for training in EO. A short list of conclusions is provided in the last section.

2 EO requests on distributed environments

Satellite image processing is usually a computational and data consuming task and special techniques are required for both data storage and processing in distributed environments. In what follows we point some main topics. This section is a survey of the ideas exposed in the recent scientific reports [3, 6, 7, 8].

2.1 Data management

The management of the distribution of data, from storing to long-term archiving, is currently an important topic in EO systems. The first issue is the data format that is varying from image files, databases, or structured file. Usually an EO data contain metadata describing the data, such as the dimensionality or reference coordinates. Another issue is related to the user need to access remotely the EO data. Due to the size of the EO data, a distributed file system is needed. For more than three decades there are several distributed file systems enabling multiple, distributed servers to be federated under the same file namespace. Another issue is the data discovery and this is currently done usually exploiting metadata catalogs. Replica management services are essential for EO systems, allowing to determine an optimal physical location for data access based on data destination aiming to reducing the network traffic and the response time. Data transfers secure protocols were developed to extends the traditional file transfer protocol.

In what concerns file catalogs there are no current standards, but several implementations are available in Grid environments that are using special file catalogs allowing data replications. The same situation is valid also for metadata catalogs; fortunately, in the particular case of EO this issue is pursued by the Open Geospatial Consortium (http://www.opengeospatial.org).

While for the basic needs mentioned above there are several stable and standardized solutions, the current key issue in EO data management is to make the data reachable and useful for any application through interoperability.

Interoperability is achieved through the usage of standard interfaces and protocols. Interoperable interfaces are attractive to users allowing the fast design of distributed application based on multiple components. Achieving interoperability includes also building adapted interfaces providing different front ends to basic services and bridging protocols. There are at least two layers for interoperability: for resource format and domain encoding, and semantic interoperability.

Interoperability solutions for resources structures and content are often application-field dependent. The solutions are related to different levels, like device, communication, middleware and deployment ones. At device level, the solutions are mostly standardized and are referring to the interfaces to the storage devices. At communication level, there are standardized data transfer protocols (as HTTP, HTTPS, or GridFTP), standardized protocols for Web services, and less standardized data movers for heterogeneous computing environments. At middleware level there are fewer standard solutions. For example, for data storage it is necessary a single consistent interface to different storage systems – a solution is coming from Grid community through the open standard storage resource manager, a control protocol for accessing mass storage.

In what concerns the interoperability of federated databases, a standard again proposed by the Grid community is the Open Grid Services Architecture Data Movement Interface (OGSA-DMI, http://forge.gridforum.org/sf/projects/ogsa-dmi-wg).

At deployment level, interoperability degradation is related to the event of new deployments and currently there are no automated tools or standard interfaces allowing the propagation of updates.

While resource-level interoperability is ensuring the compatibility of implementations at hardware and software levels, the semantic interoperability is enabling data and information flows to be understood at a conceptual level. Research efforts are currently devoted to the definition of generic data models for specific structured linguistic data types with the intention to represent a wide class of documents without loosing the essential characteristics of the linguistic data type.

Data provision services in EO are not satisfying the nowadays' user needs due to current application and infrastructure limitations. The process of identifying and accessing data takes up a lot of time, according [6], due to: physical discontinuity of data, diversity of metadata formats, large volume of data, unavailability of historic data, and many different actors involved.

In this context, there is a clear need for an efficient data infrastructure able to provide reliable long-term access to EO data via the Internet, and to allow the users to easily and quickly derive information and share knowledge. Recognizing these needs, the European INSPIRE Directive (http://inspire.jrc.ec.europa.eu) requires all public authorities holding spatial data to provide access to that data through common metadata, data and network service standards. OPeNDAP (http://opendap.org/) is a data transport architecture and protocol widely used in EO; it is based on HTTP and includes standards for encapsulating structured data, annotating the data with attributes, and adding semantics that describe the data. Moreover, it is widely used by governmental agencies to EO data [6].

The Committee on EO Satellites (www. ceos.org) maintains a Working Group on Information Systems and Services with the responsibility to promote the development of interoperable systems for the management of EO data internationally. This group plans to build in the next decade the Global EO System of Systems (GEOSS) targeting the development of a global, interoperable geospatial services architecture [11].

2.2 Data processing

To address the computational requirements introduced by time-critical satellite image applications, several research efforts have been oriented towards parallel processing strategies. According to the Top500 list of supercomputer sites, NASA, for example, is maintaining two massively parallel clusters for remote sensing applications. The recent book [23] presents the latest achievements in the field of high performance computing (HPC).

Currently ongoing research efforts are aiming also the efficient distributed processing of remote sensing data. Recent reports are related to the use of new versions of data processing algorithms developed for heterogeneous clusters as [22]. Moreover, distributed application framework specifically have been developed for remote sensed data processing, like JDAF [30]. EO applications are also good candidates for building architectures based on components encapsulating complex data processing algorithms and being exposed through standard interfaces like in [10].

As datasets grow larger, the most efficient way to perform data processing is to move the analysis functions as close to the data as possible [28]. Data processing can be easily expressed today by a set-oriented, declarative language whose execution can benefit enormously from cost-based query optimization, automatic parallelism, and indexes. Moreover, complex class libraries written in procedural languages were developed as extension of the underlying database engine. MapReduce, for example, is nowadays a popular distributed data analysis and computing paradigm; its principle resembles the distributed grouping and aggregation capabilities existing in parallel relational database systems. However, partitioned huge datasets are making distributed queries and distributed joins difficult. While simple data-crawling strategy over massively scaled-out data partitions is adequate with MapReduce, this strategy is suboptimal: a good index can provide better performance by orders of magnitude [28]. Moreover, joins between tables of very different cardinalities are still difficult to use.

Web services technology emerged as standard for integrating applications using open standards. In EO, the Web services play a key role. A concrete example is the Web mapping implementation specification proposed by OpenGIS (http://www.opengis. org). Web technologies are allowing also the distribution of scientific data in a decentralized approach and are exposing catalogue services of dataset metadata.

Grid computing services and more recent Cloud computing services are going beyond what Web services are offering, making a step forward towards an interactive pool of processes, datasets, hardware and software resources.

3 Grid-based environments for Earth observation

The promise of a Grid for EO community is to be a shared environment that provides access to a wide range of resources: instrumentation, data, HPC resources, and software tools. There are at least three reasons for using Grids for EO:

1. the required computing performance is not available locally, the solution being the remote computing;
2. the required computing performance is not available in one location, the solution being cooperative computing;
3. the required services are only available in specialized centres, the solution being application specific computing.

Realizing the potential of the Grid computing for EO, several research projects were launched to make the Grid usage idea a reality. We review here the most important ones.

3.1 Grid-based EO initiatives

Within the DataGrid project funded by the European Commission, an experiment aiming to demonstrate the use of Grid technology for remote sensing applications has been carried out; the results can be found for example in the paper [15]. Several other international Grid projects were focused on EO, like SpaceGrid (http://www.spacegrid.org), Earth Observation Grid (http://www.e-science.clrc.ac.uk/web/projects/earthobservation), or Genesis [35]. The MediGrid project (http://www.eu-medigrid.org) aimed to integrate and homogenize data and techniques for managing multiple natural hazards. The authors of paper [1] present an overview of SARA Digital Puglia, a remote sensing environment that shows how Grid and HPC technologies can be efficiently used to build dynamic EO systems for the management of space mission data and for their on-demand processing and delivering to final users.

A frequent approach is to use the Grid as a HPC facility for processor-intensive operations. The paper [29], for example, focuses on the Grid-enabled parallelization of the computation-intensive satellite image geo-rectification problem. The aim of the proposed classification middleware on Grid from [31] is to divide jobs into several assignments and submit them to a computing pool. The parallel remote-sensing image processing software PRIPS was encapsulated into a Grid service and this achievement was reported in [33]. In the paper [34] is discussed the architecture of a spatial information Grid computing environment, based on Globus Toolkit, OpenPBS, and Condor-G; a model of the image division is proposed, which can compute the most appropriate image pieces and make the processing time shorter.

CrossGrid (http://www.crossgrid.org) aimed at developing techniques for real-time, large-scale grid-enabled simulations and visualizations, and the issues addressed included distribution of source data and the usefulness of Grid in crisis scenarios.

DEGREE (http://www.eu-degree.eu) delivered a study on the challenges that the Earth Sciences are imposing on Grid infrastructure. D4Science (http://www.d4science.org) studied the data management of satellite images on Grid infrastructures. G-POD (http://eogrid.esrin.esa.int/) aims to offer a Grid-based platform for remote processing the satellite images provided by European Space Agency. The GlobAEROSOL service of BEinGRID [24] is processing data gathered from satellite sensors and generates an multi-year global aerosol information in near real time.

The GEOGrid project [26] provides an e-Science infrastructure for Earth sciences community and integrates a wide varieties of existing data sets including satellite imagery, geological data, and ground sensed data, through Grid technology, and is accessible as a set of services.

LEAD (https://portal.leadproject.org/) is creating an integrated, scalable infrastructure for meteorology research; its applications use large amounts of streaming data from sensors.

The Landsat Data Continuity Mission Grid Prototype (LGP) offers a specific example of distributed processing of remotely sensed data generating single, cloud and shadow scenes from the composite of multiple input scenes [8].

GENESI-DR (http://genesi-dr.eu) intends to prove reliable long-term access to Earth Science data allowing scientists to locate, access, combine and integrate data from space, airborne and in-situ sensors archived in large distributed repositories; its discovery service allows to query information about data existing in heterogeneous catalogues, and can be accessed by users via a Web portal, or by external applications via open standardized interfaces (OpenSearch-based) exposed by the system [6].

Several other smaller projects, like MedioGrid [20], were also initiated to provide Grid-based services at national levels.

3.2 Remote sensing Grid

A Remote Sensing Grid (RSG) is defined in [8] as a highly distributed system that includes resources that support the collection, processing, and utilization of the remote sensing data. The resources are not under a single central control. Nowadays it is possible to construct a RSG using standard, open protocols and interfaces. In the vision of [8] a RSG is made up of resources from a variety of organizations which provide specific capabilities, like observing elements, data management elements, data processing and utilization elements, communications, command, and control elements, and core infrastructure.

If a service oriented architecture is used, modular services can be discovered and used to build complex applications by clients. The services should have the following characteristics [8]: composition, communication, workflow, interaction, and advertise. These requirements are mapped into the definition of specific services for workflow management, data management and processing, resource management, infrastructure core functions, policy specification, and performance monitoring.

The services proposed in [8] are distributed in four categories: workflow management services, data management services, applications in the form of services, and core Grid services.

In the next section we describe a case study of a recent Grid-based satellite imagery system that follows the RSG concepts.

4 Case study: GiSHEO

The rapid evolution of the remote sensing technology is not followed at the same developing rate by the training and high education resources in this field. Currently there are only a few number of resources involved in educational activities in EO. The CEOS Working Group of Education, Training and Capacity Building is one of the few facilities that is collecting an index of free EO educational materials (http://oislab.eumetsat.org/CEOS/webapps/).

Recognizing the gap between research activities and the educational ones, we have developed recently a platform, namely GiSHEO (On Demand Grid Services for Training and High Education in EO, http://gisheo.info.uvt.ro) addressing the issue of specialized services for training in EO. Contrary to the existing platforms providing tutorials and training materials, GiSHEO intends to be a living platform where experimentation and extensibility are the key words. Moreover, special solutions were proposed for data management, image processing service deployment, workflow-based service composition, and user interaction. A particular attention is given to the basic services for image processing that are reusing free image processing tools. A special feature of the platform is the connection with the GENESI-DR catalog mentioned in the previous section.

While the Grid is usually employed to respond to the researcher requirements to consume resources for computational-intensive or data-intensive tasks, GiSHEO aims to use it for near-real time applications for short-time data-intensive tasks. The data sets that are used for each application are rather big (at least of several tens of GBs), and the tasks are specific for image processing (most of them very simple). In this particular case a scheme of instantiating a service where the data are located is required in order to obtain a response in near-real time. Grid services are a quite convenient solution in this case: a fabric service is available at the server of the platform that serves the user interface and this service instantiates the processing service where the pointed data reside. In our platform the Web services serve as interfaces for the processing algorithms. These interfaces allow the remote access and application execution on a Grid using different strategies for fast response.

The platform design concepts were shortly presented in [5, 17] and the details about the e-learning component can be found in [9]. The EO services were described in [21] and the data management is detailed in [13]. In this section we describe the technological approaches that were undertaken to solve the key issues mentioned in the previous sections.

4.1 System architecture

The GiSHEO Architecture is a Grid-enabled platform for satellite image processing using a service oriented architecture structured on several levels including user, security, service, processing and a data level (Figure 1).

The user level is in charge with the access to the Web user interface (built by using DHTML technologies).

The security level provides security context for both users and services. The security context defines the mechanisms used for authentication, authorization and delegation. Each user must be identified by either using a username/password pair or a canonical name provided by a digital certificate. The services are using a digital certificate for authentication, authorization, and trust delegation. The authorization is based on VOMS (Virtual Organization Management Service, http://vdt.cs.wisc.edu/VOMS-

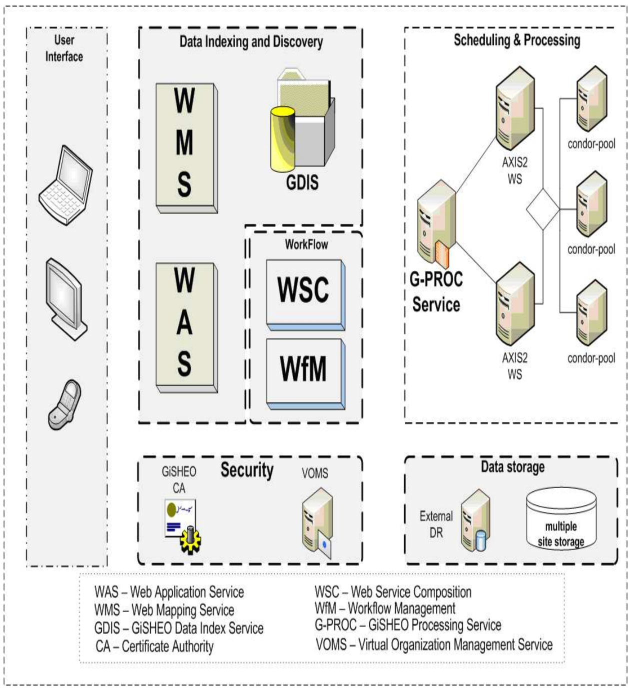

User Interface
Data Indexing and Discovery
WMS
GDIS
WAS
WorkFlow
WSC
WfM
Scheduling & Processing
AXIS2 WS
condor-pool
condor-pool
condor-pool
G-PROC Service
AXIS2 WS
GiSHEO CA
Security
VOMS
Data storage
External DR
multiple site storage
WAS – Web Application Service
WMS – Web Mapping Service
GDIS – GiSHEO Data Index Service
CA – Certificate Authority
WSC – Web Service Composition
WfM – Workflow Management
G-PROC – GiSHEO Processing Service
VOMS – Virtual Organization Management Service

Figure 1: GiSHEO's architecture.

documentation.html) service which extends the PKI (Public Key Infrastructure) mechanism by adding support for certificate attributes. The service level exposes internal mechanisms part of the GiSHEO platform by using various Grid services technologies including:

– The Grid processing service - internal processing platform exposed as a specialized Web service and capable of connecting with an external resource management system.

– The workflow service - internal workflow engine which can be accessed by using a specialized Web service.

– The data indexing and discovery - access to the GiSHEO proposed data management mechanisms.

At processing level the GiSHEO platform uses Condor HTC (Condor High Throughput Computing, http://www.cs.wisc.edu/condor/ description.html) as processing model, task registration, scheduling and execution environment. It uses also a direct job submission interface using Condor's specific Web service interface.

At data level two different types of data are involved: database datasets which contain the satellite imagery repository and processing application datasets used by applications to manipulate satellite images.

Each of these components will be presented in further details in the following subsections.

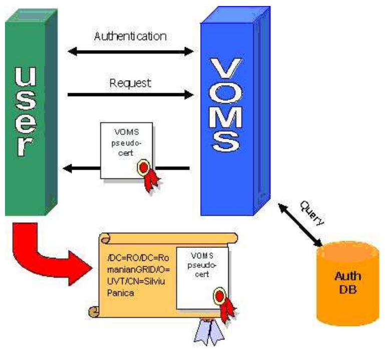


Figure 2: VOMS workflow.

4.2 Security mechanisms

The security level of the GiSHEO platform is divided into three categories: authentication, authorization and delegation.

The authentication is accomplished by using X.509 digitally signed certificates. Each entity either a user or a service will have a digital certificate attached. In case users choose to use only a username/password pair, a digital certificate (also a user private/public key) will be also generated and stored on the Web portal side (using a private key vault). In order to be valid, each certificate must be signed by a trusted CA (Certificate Authority), GiSHEO CA or a EuGRIDPMA (European Policy Management Authority for Grid Authentication, www.eugridpma.org) affiliated CA.

In conclusion each entity will be uniquely identified by using a CN (Canonical Name) which has the following form:

```
/DC=RO/DC=GiSHEO CA/O=UVT/CN=User1
/DC=RO/DC=GiSHEO CA/O=UVT/CN=service/GTD
```

For authorization, a VOMS service approach has been chosen. VOMS is a service providing VO membership for both users and services by using a set of attributes that are included inside the user's digital certificate. VOMS can be viewed as an extension to the simple digital certificate authorization (in which case only CA signing validation is made). As the following example will show in VOMS each entity is mapped to a set of attributes as configured by the VO administrator:

```
"/DC=RO/DC=GiSHEO CA/O=UVT/CN=User1" .gisheo
"/DC=RO/DC=GiSHEO CA/O=UVT/CN=U2" .sgmgisheo
"/gisheo/Role=ops" .gisheo
"/gisheo/Role=ops/Capability=NULL" .gisheo
"/gisheo/Role=VO-Admin" .sgmgisheo
```

In the above example `User1` is mapped to `.gisheo` group while `U2` is mapped to `.sgmgisheo` group. Each group has attached one or more attributes. For example group `.sgmgisheo` has attribute `/gisheo/Role=VO-Admin` attached to it which means that any service user belonging to group `.sgmgisheo` is a VO Administrator.

The VOMS authorization process is described by the following steps (Figure 2). First a request for validation arrives from a VOMS service. Then the VOMS service checks the user's CN membership and creates a proxy certificate (with a limited life time) with user's attributes available for GiSHEO VO. After the creation of the proxy certificate it can be used by the user to securely access any services belonging to the GiSHEO VO.

The GiSHEO Infrastructure uses a single sign-on authentication system; therefore a delegation of credentials mechanism is needed. For this to happen a MyProxy Credentials Management Service (http://grid.ncsa.uiuc.edu/myproxy/) is used. MyProxy also provides VOMS authentication mechanisms therefore can be easily integrated with the VOMS service. The goal of the MyProxy service is to provide delegation mechanism for both entities, users and services.

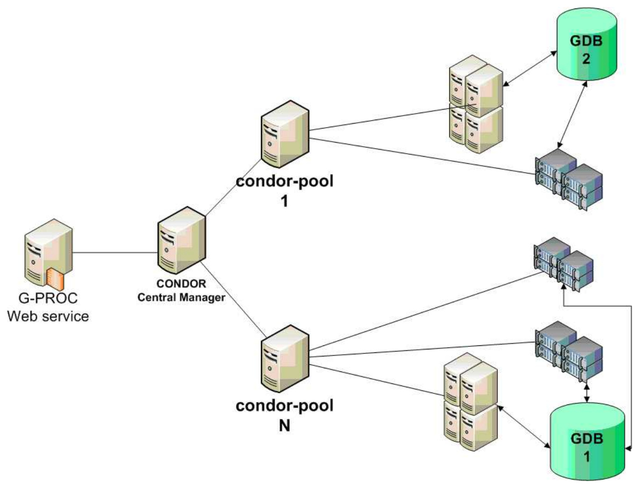


Figure 3: G-PROC with CondorWrappers.

4.3 Processing platform

The GiSHEO processing platform consists of three parts, the Web service interface for the processing platform, the EO processing tools connectors and a set of connectivity wrappers that describe the mechanism of connecting GiSHEO's platform to Condor HTC workload management system (WMS).

The Grid Processing Web service (G-PROC) is built using Web service technologies from Apache Axis2 (http://ws.apache.org/axis2/). It is responsible for the interaction with other internal services including the Workflow Composition Engine. Its main responsibilities are at this point to receive tasks from the workflow engine or directly from the user interface, to use a task description language (the ClassAd meta language for example in case of Condor HTC) in order to describe a job unit, to submit and check the status of jobs inside the workload management system, and to retrieve job logs for debugging purposes.

In order to access the available computational resources, G-PROC provides a set of wrappers as interface with the Condor WMS (Figure 3). This interface can be expanded to support other types of grid resources (e.g. Globus Toolkit 4, EGEE gLite Middleware).

The proposed wrapper interface is able to support the entire Grid specific life cycle: job registration (each task request must be translated into the WMS specific language - i.e the ClassAd language for Condor), job submission (each translated task becomes a Condor specific job ready to be executed), job status (at any time the status of submitted jobs can be retrieved), job logging (when requested, a logging file can be created for debugging reasons) and job retrieval (at the end the output of the job execution can be retrieved).

G-PROC's last component is represented by the processing tools' connectors. The connectors create a bridge between the processing tools and the processing infrastructure. They are required when the arguments of the processing application need to be translated so that they match arguments defined by the WMS description language. They are also linked directly with the WMS and its specific job description language.

4.4 Storage architecture

One of the fundamental components of the GiSHEO project is the system responsible for storing and querying the geographical data.

The GiSHEO Data Indexing and Storage Service (GDIS) provides features for data storage, indexing data using a specialized RDBMS, finding data by various conditions, querying external services and for keeping track of temporary data generated by other components. GDIS is available to other components or external parties using a specialized Grid service. This service is also responsible for enforcing data access rules based on specific Grid credentials (VO attributes, etc.).

4.4.1 Data storage

The Data Storage component part of GDIS is responsible for storing the data by using available storage backends such as local disk file systems (eg. ext3), local cluster storage (eg. GFS [27], GPFS [25]) or distributed file systems (eg. HDFS, KosmosFS, GlusterFS). One important requirement for the storage component is that data distributed across various storage domains (local or remote) should be exposed through a unique interface.

This is achieved by implementing a front-end GridFTP service capable of interacting with the storage domains on behalf of the clients and in a uniform way (Figure 4). The GridFTP service also enforces the security restrictions provided by other specialized services and related with data access.

The GridFTP service has native access to the Hadoop Distributed File System (http:// lucene.apache.org/hadoop) offering access to data stored inside the internal HDFS file systems and providing the required access control facilities.

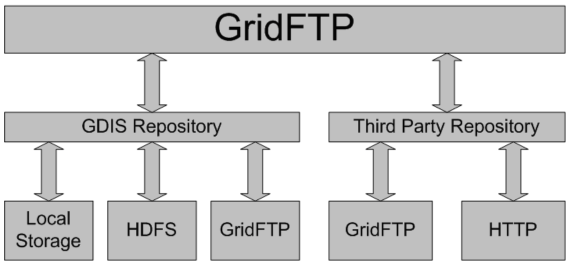


Figure 4: GridFTP interface.

Together with the data indexing components the GridFTP service provides features for manipulating the data repository by providing a basic method for managing data (upload, deletion, retrieval, etc.).

4.4.2 Data indexing

The data indexing components enable GDIS to provide fast access to the data offered by the storage component. Data indexing is performed by PostGIS, an OGC (Open Geospatial Consortium) compliant Spatial Database extension for the PostgreSQL RDBMS engine. The PostGIS layer indexes the metadata and location of the geographical data available in the storage layer. The metadata usually represents both the extent or bounding box and the geographical projection of the data (representing the exact geo-location).

The PostGIS Layer provides advanced geographical operations (backed by a GiST index) which allows searching the data by using various criteria including interaction with raw shapes, interaction with shapes representing geo-political data (country, city, road, etc.) or any other type of geographical data which can be represented in PostGIS. The geo-political data is typically provided by data imported from public sources and extended using the Open Street Map (OSM) data.

The catalogue also maintains data about the type of the datasets. This data is useful not only for retrieving data from the catalogue but also for the workflow engine and execution components. It is used by these components to enforce the data types that can be used with various image processing operations or workflows.

4.4.3 Data query

Given the advanced data indexing capabilities of the PostGIS component, GiSHEO provides an advanced and highly flexible interface for searching the project's geographical repository. The search interface is built around a custom query language (LLQL - Lisp Like Query Language) designed to provide fine grained access to the data in the repository and to query external services (TerraServer, GENESI-DR, etc). The syntax of the query language is inspired from the syntax of the LISP language and partially by LDAP filters. The language allows querying the repository both for raster images (Figure 5) and also for various aggregated data or object properties (Figure 6).

```
(select '(url, owner)
 (and
  (or
   (ogc:interacts
    (osm:country "Romania"))
   (ogc:interacts
    (osm:country "Hungary"))
    )
  (gdis:type "RASTER/AERIAL")
 )
)
```

Figure 5: Raster query.

```
; Find cities in Romania
; filter by bbox
(select-place '(name)
 (and
  (ogc:interacts
   (ogc:bbox 16.69 43.97 24.8 48.48))
  (osm:country "Romania")
  (osm:type "city")
 )
)
```

Figure 6: Feature query.

Figures 5 and 6 show in detail how the LLQL syntax looks like. The PostGIS related queries are translated directly to PostGreSQL queries, while the external lookups are resolved prior to submitting the query's to PostGIS.

The GDIS layer also provides a simpler query language called GDIS Query Language (GQL). GQL is suitable for search engines or direct user query. The queries are translated automatically into corresponding SQL queries (through an extensible custom made SQL generation engine).

A simple example of an GQL query is:

```
"place:Timisoara,Timis,Romania
 type:DEM vendor:NASA"
```

When invoked using this query the catalogue returns all datasets that match the criteria (when an explicit operand was not specified the default is `and`).

4.4.4 External services

Another set of tasks handled by GDIS are represented by the interaction with external services. In this case GDIS represents a thin middleware layer interacting with external repositories and exposes only one unique interface (similar and possibly integrated with the internal repositories). One example of external back-ends supported by GDIS is represented by the GENESI-DR catalog.

4.5 Image processing workflows

Processing satellite images for usage in geographical and historical analysis could require large amount of processing steps involving different image processing transformations. This scenario implies linking the processing algorithms to form a workflow either defined by the user or selected from an already existing list. These algorithms could be located on the same or on different machines spread inside a Grid. In the former case each of them could be exposed as a Web or Grid service. Distributing them across a Grid where each resource exposes several algorithms could help in balancing the resource workload.

Starting from some practical applications involving historical and geographical analysis a few usage scenarios which require more than one image transformation on the source image can be detailed.

In archaeology, for example, assuming that there is a need to identify artificial ancient sites containing human settlements and fortifications from satellite images, the following image transformations could be applied in sequence: gray level conversion, histogram equalization, quantization and thresholding. Applying the resulted image over the initial one allows users to better observe the previously described artificial structures.

Another example could involve identifying linear shapes such as roads or linear fortifications (wave like structures). In this case a workflow made up of the following sequence of operations could provide useful information as output: grayscale conversion, edge detection (e.g. Canny filter), lines detection (e.g. Hough transform).

Related with geography a scenario in which the vegetation index for some particular area needs to be detailed could result from the following operations which also need to be applied in sequence: extract red band, extract infrared band, compute by using the previously obtained images the Normalized Difference Vegetation Index (NDVI).

In the same way detecting changes in river beds or vegetation can be computed by first applying some shape recognition techniques on images taken at different moments in the past and then by overlaying them.

4.6 Workflow composition engine

In general workflow image processing transformations are sequential or parallel involving at some point a join or a split. There are few cases which require the use of loops.

As each of the transformation is exposed as a Web or Grid service belonging to a certain resource and due to the dynamic nature of Grids a self adaptive scenario in which tasks would be reassigned (when their corresponding resources might become unable to solve them needs) to be taken into consideration. The natural way to achieve this is by using an Event-Condition-Action (ECA) approach. ECA usually implies a rule governed scenario in which an action takes place on as a result of an event and in case one or more conditions are met. The reason for choosing this paradigm is that it allows the separation of logic represented by rules and data which is represented by objects, declarative programming which is useful for applications focused on what to do instead on how to do it, scalability, centralization of knowledge, etc.

The proposed workflow engine, namely OSyRIS (Orchestration System using a Rule based Inference Solution), detailed in this section is based on DROOLS (http://drools.org) which uses an object oriented version of the RETE [4] algorithm. A simplified workflow language has been built on top of it with the aim of offering a simple yet general rule based language. The following subsection will present in greater detail the language with emphasis on its syntax and its capability of expressing general workflows.

4.6.1 Rule-based language

The rule based language, namely SiLK (Simple Language for worKflow), envisioned as part of the GiSHEO project aims at providing a simple yet general language which could be used without modifications in general purpose workflows. The basis of the language are the following: tasks and relations (rules) between them. It is similar with the SCUFL [16] language, but does not rely on XML: it allows the introduction of more workflow specific issues and the ECA approach allows a greater flexibility when expressing data and task dependencies. The following paragraphs will detail these characteristics.

Each task is made up of several mandatory and optional attributes. The mandatory attributes consist of at least one input and one output port. Each task can have several such ports as it could receive input from more than one task and could produce more than one output. The optional attributes are also called meta-attributes. They are not used by the workflow engine and are simply passed over to the service handling the task under the assumption that it can decode and understand them. Meta-attributes are declared by using quotation marks both for the attribute name as well as for the attribute value.

Each task needs to be declared before actually being used. It can be noticed the lack of any mandatory attributes concerning the input or output data type and content. This is due to the fact that the compatibility issues between tasks are resolved at the moment the workflow is created by using methods which are specific to each workflow. These methods should be implemented by the platform running the workflow engine and should not be incorporated inside the workflow description. Because of the nature of the rule engine there is a need for a fictive start task which has the role of a trigger causing the first actual task in the workflow to be executed.

Rules are defined by simply mentioning the events and conditions which should take place in order to trigger the execution of right hand side tasks. Each event is being seen as a completed task and is placed on the left hand side of the rule. Linking the output of left hand tasks with the input of right hand side tasks is accomplished by using variables.

For example the rule:

```
A[a=o1] -> B[i1=a]
```

links the output of task A with the input of task B through variable a. Conditions are placed at the end of the rule and can involve numerical or string variables:

```
A[d=o1] -> B[i1=d] | d<1.
```

In the same way splits and joins made of, but not restricted to, two tasks could be expressed in the same way as the following rules:

```
# synchronized join
A[b=o1],B[c=o1] -> C[i1=b#i2=c]
# parallel split
A[a=o1] -> B[i1=a],C[i1=a],D[i1=a]
```

Loops can be also modeled as in the following example:

```
A[d=o1],B[e=o1] -> A[i1=d#i2=e] | d<1
and
A[d=o1],B[e=o1] -> C[i1=d#i2=e] | d>=1.
```

The former rule expresses the condition to reiterate the loop while the latter expresses the condition to exit the loop. As a remark it should be noticed that when several right hand side tasks need to be activated their execution will take place in parallel. Synchronization between several tasks can also be achieved by adding them into the left hand side of the rule:

```
A[b=o1],B -> C[i1=b].
```

The previous example shows how task A is synchronized with task B and cannot execute until the latter one is completed. Tasks can also have multiple instances. For instance, a rule could produce 5 instances of a task:

```
B[a=o1] -> C[i1=a#instances=5]
```

with the default number of instances being one. Instances of left hand tasks are usually consumed when the rule fires

```
B[a=o1#consume=true] -> C[i1=a].
```

However this feature is optional with the default behaviour being consume. It is useful for example when there are n rules with the same left hand task which can fire but only k rules are needed. In this case having k instances of that task could prove useful. Another utility for this feature could be the case when a rule is needed to be fired several times. Multiple task instances allow users to express workflows related to natural processes such as those involved in chemical reactions [14] and cellular membranes [18].

Several meta-attributes including datagroup, dataset, processing and argument-list need to be introduced in order to fully define a GiSHEO workflow. The meta-attributes are used to identify the image to be processed, the operation and the arguments to be used. For example the datagroup and dataset attributes identify the group and the set inside the group to which the image belongs. The processing attribute identifies the operation to be applied to the image. Its value follows a C-like prototype format with return type, operation name and argument list. The attribute-list specifies the optional attributes used by the operation. It is a list where the values are pairs in the form <name=value>. Each value is separated by a # sign. The name inside the pair must match the name given to the attribute in the processing description.

```
# Define a fictive start task.
# Needed for initial activation
A0:= [o1:output="FTP address",
     "instances"="1"];
# The following two tasks belong to the
# processing workflow
A:= [i1:input,o1:output,"datagroup"="ID",
 "dataset"="ID",
 "processing"="outIMG_1 grayscale(inIMG_1)",
 "argument-list"=""];
B:= [i1:input,o1:output,"datagroup"="ID",
 "dataset"="ID", "processing"=
 "outIMG_1 canny(inIMG_1#canny1#
  aperture_size#canny2)",
 "argument-list"=
 "<canny1=80>#<aperture_size=3>#
  <canny2=120>"];
C:=[i1:input,o1:output,"datagroup"="ID",
 "dataset"="ID","processing"=
 "outIMG_1 hough_lines(inIMG_1#min_line#
  min_gap#hough_treshold)",
 "argument-list"=
 "<min_line=20>#<min_gap=10>#
 <hough_treshold=100>",
 "isLast"="true"];

# Start rule: compute grayscale
# from the initial image
A0[a=o1] -> A[i1=a];
# Compute Canny from the grayscale image
# A[a=o1] -> B[i1=a];
# Compute a Hough transform from
# the Canny image
B[a=o1] -> C[i1=a];
```

Figure 7: Workflow example using SiLK

Figure 7 shows a complex example, for detecting linear structures by combining several basic operations as: grayscale conversion, edge detection with Canny filter and lines detection with Hough transform. It also presents the required attribute values.

As previously mentioned using an ECA approach allows for creating adaptive workflows which can react to changes either in the configuration of the Grid or inside the workflow itself. Changes inside the Grid are handled by creating specific rules which allow resource selection based on various task scheduling criteria. Modifications of the workflow are usually accomplished either by inserting or retracting at runtime rules belonging to it or by modifying the executor of the task in case a better one is found. It is very hard or almost impossible to express adaptivity by using classic workflow languages such as WS-

BPEL (http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html), SCUFL [16], JSDL (Job Submission Description Language (JSDL), http://www.gridforum.org/documents/GFD.56.pdf), DAGs (Directed Acyclic Graphs) or Petri Nets. Most of the previously listed languages are either too verbose and strict due to the use of XML or their structure is predetermined by the corresponding DAG or Petri Net which cannot be changed at runtime.

Different from the previous classic approaches are the ones based on ECA. The nature inspired approach based on chemical reactions and described in paper [14] falls into this category. Moreover it allows self adaptation to changes by using the High Order Chemical Language also described in the paper. Other approaches include the AGENTWork [12], a workflow engine based on a language supporting rule based adaptation and which is used in medical practices. The paper [32] presents a Condition-Action (CA) language and platform called FARAO, based on an Adaptive Service Oriented Architecture (ASOA), which has the aim of supporting the development of adaptable service orchestrations.

4.6.2 Particularities of the workflow engine

Each of the non ECA workflow languages listed in the previous subsection has a workflow enactment engine built on top of it. WS-BPEL relies for example on Active-BPEL (http://www.active-endpoints.com/active-bpel-engine-overview.htm), SCUFL uses Taverna [16] while Condor and Pegasus [2] use DAGman to execute workflows expressed as DAGs.

Not all of the previously mentioned workflow systems provide however sufficient functionalities to cover system and logical failures that could occur when running the workflow. To cope with this issue a workflow management platform has been built on top of the workflow language described in the previous subsection. Its role is to execute the workflow and to handle system and logical failures by inserting additional rules inside the initial rule database. These new rules will be expressed by using the same language and will be interpreted by using DROOLS.

The workflow platform is also responsible for checking the integrity of the workflow syntax. However it does not check the compatibility between different nodes. This problem is left to the specialized system which will use the platform.

Given existing ECA based workflow engines such as AGENTWork [12] which already deal with adaptiveness issues, the aim of this platform is not only to provide a simpler yet general language but to offer solutions in form of workflows to different problems such as archaeological and geographical image related transformations. In this direction the goal is to build a workflow backwards from the desired result to the input requirements given by the user, and to provide the user with one or more solutions from which he/she can use the most appropriate one.

4.7 Example for visualising linear shapes using the GiSHEO interface

GiSHEO offers a web portal from which the user can select images, apply various operations on them and visualize the results. In what follows we present an example that shows all the required steps starting with the selection of the image and until the final image is obtained. The processing we selected follows the workflow example described in Figure 7 from previous subsection.

First the desired geographical area needs to be selected by introducing a GQL expression (see subsection 4.4.3 for details) as shown in Figure 8. After the area is selected the user can choose the area of interest (see Figure 9). This is done by simply using the mouse to select a rectangular region that will be used when image processing operations are invoked.

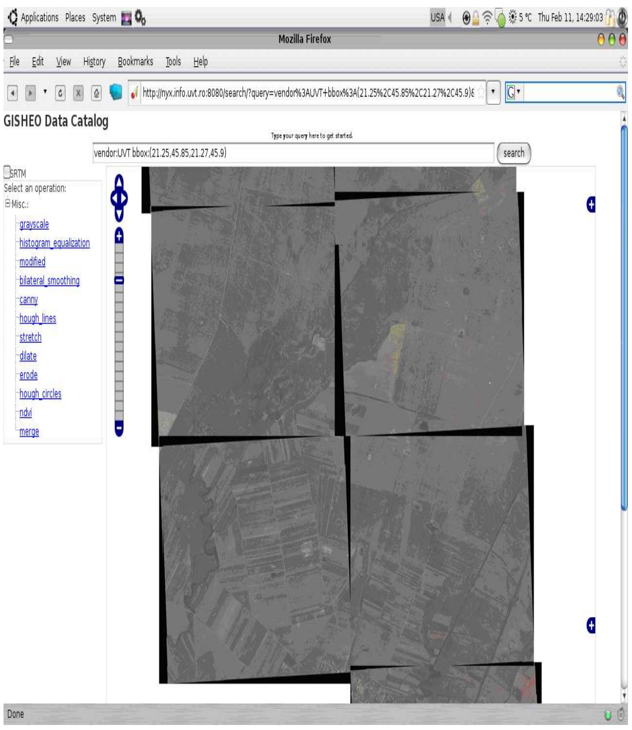
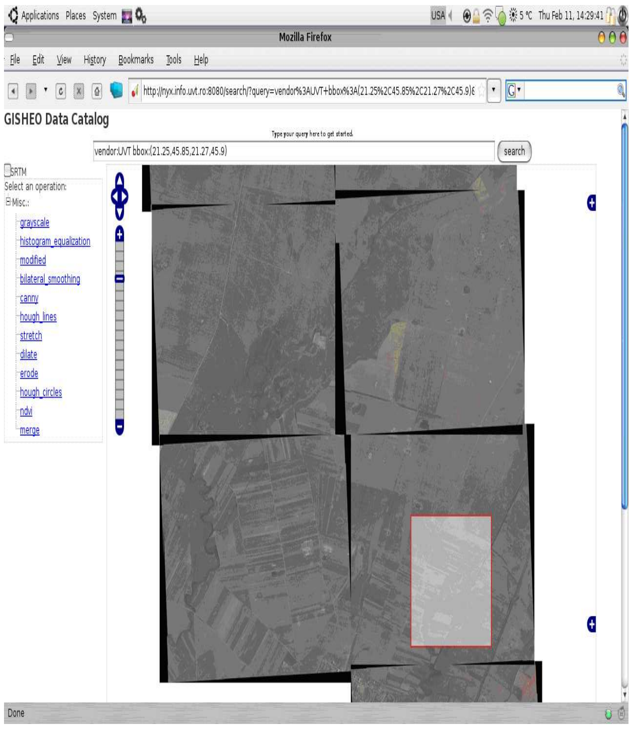


Figure 8: Selecting the images by geographical position.

Once the desired area has been selected the user can proceed by choosing the operations he/she would like to perform on the images. As our example follows the workflow specified in Figure 7, the first operation the user will select is the grayscale conversion (see Figure 10). Once a name for the result image is set the request is sent to the server which triggers the execution. The result can be visualized as soon as it is available (as seen in Figure 11).

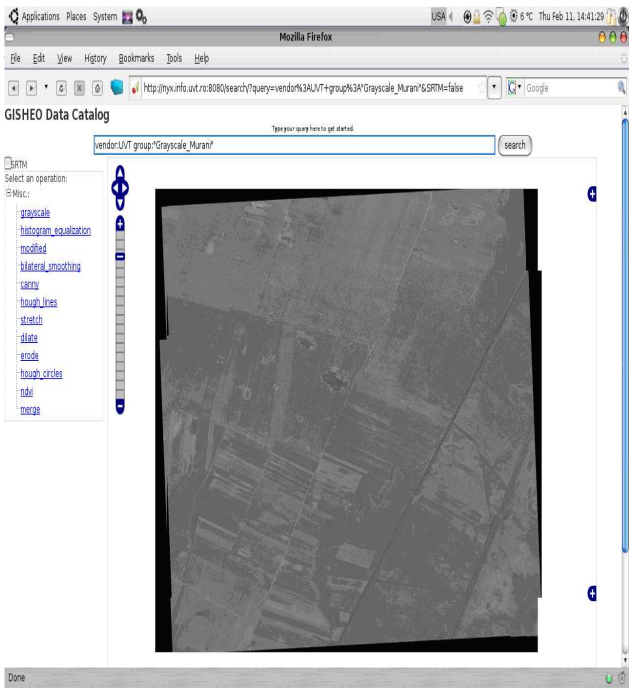
The other requests are for the Canny edge detector (see Figures 12 and 13) and for the Hough transform for lines detection (see Figures 14 and 15).

Once the final operation is completed the user can clearly see the linear shapes indicated by red lines (see Figure 15). By comparing this final image with the initial one (see Fig-


GISHEO Data Catalog

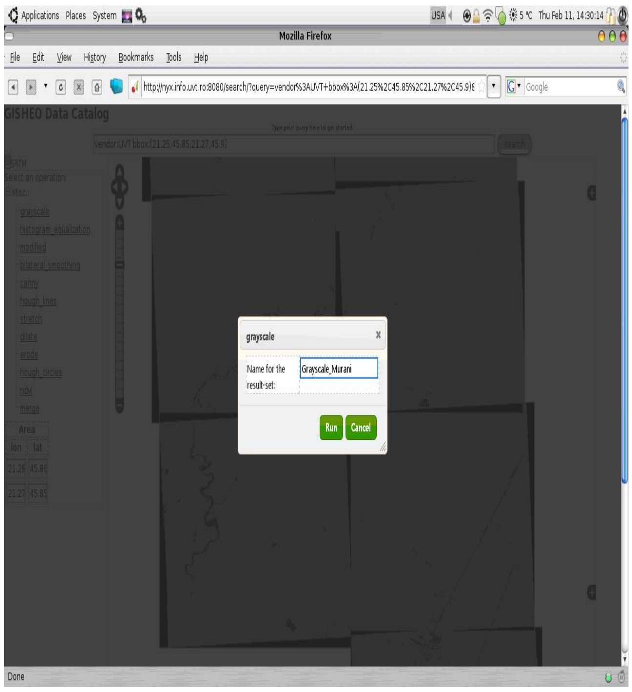
vendor:UVT bbox:(21.25,45.85,21.27,45.9)
Applications Places System
Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://nyx.info.uvt.ro:8080/search/?query=vendor%3AUVT+bbox%3A(21.25%2C45.85%2C21.27%2C45.9)
Type your query here to get started.
search
SRTM
Select an operation:
Misc.:
grayscale
histogram_equalization
modified
bilateral_smoothing
canny
hough_lines
stretch
dilate
erode
hough_circles
ndvi
merge
Done

Figure 9: Selecting the area of interest inside the selected images.


GISHEO Data Catalog

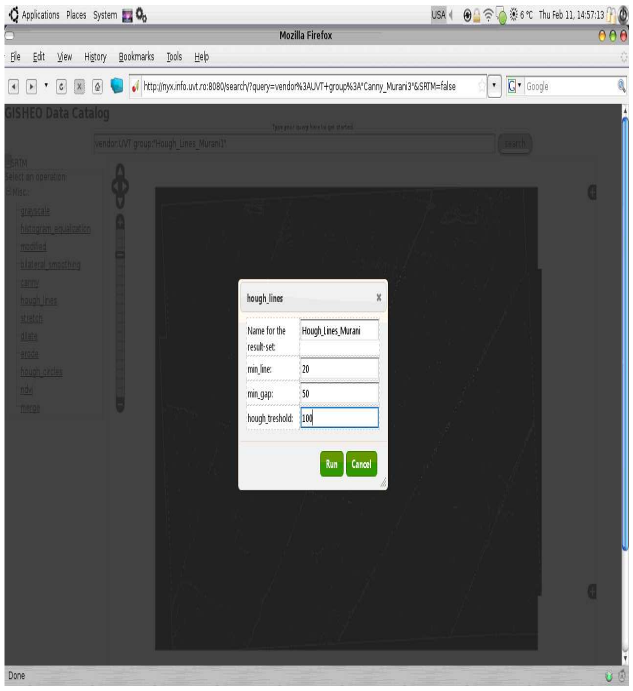
vendor:UVT group:"Grayscale_Murani"
Applications Places System
Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://nyx.info.uvt.ro:8080/search/?query=vendor%3AUVT+group%3A"Grayscale_Murani"&SRTM=false
Type your query here to get started.
search
SRTM
Select an operation:
Misc.:
grayscale
histogram_equalization
modified
bilateral_smoothing
canny
hough_lines
stretch
dilate
erode
hough_circles
ndvi
merge
Done

Figure 11: Visualizing the result of the grayscale conversion.


grayscale

Name for the result-set: Grayscale_Murani

Figure 10: Applying the grayscale conversion.


hough_lines

Name for the result-set: Hough_Lines_Murani

min_line: 20

min_gap: 50

hough_treshold: 100
Mozilla Firefox
http://nyx.info.uvt.ro:8080/search/?query=vendor%3AUVT+group%3A"Canny_Murani3"&SRTM=false
GISHEO Data Catalog
Run
Cancel
Done

Figure 12: Applying the Canny edge detector on the grayscale image.

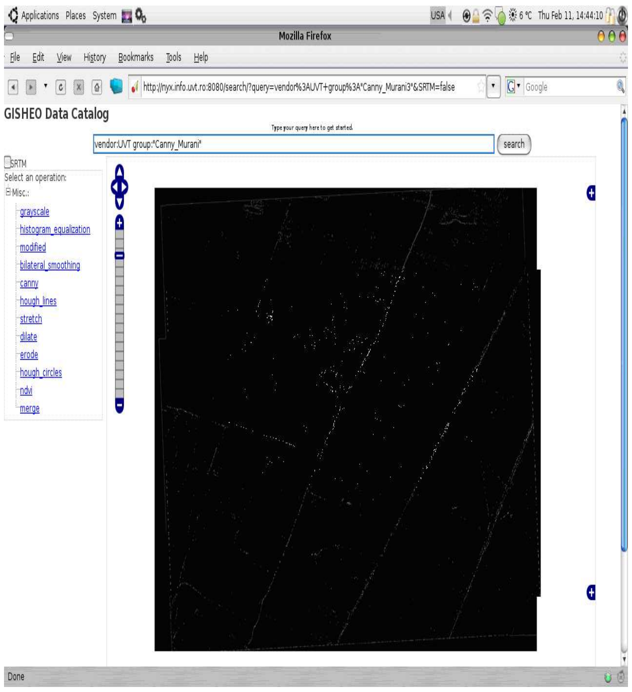
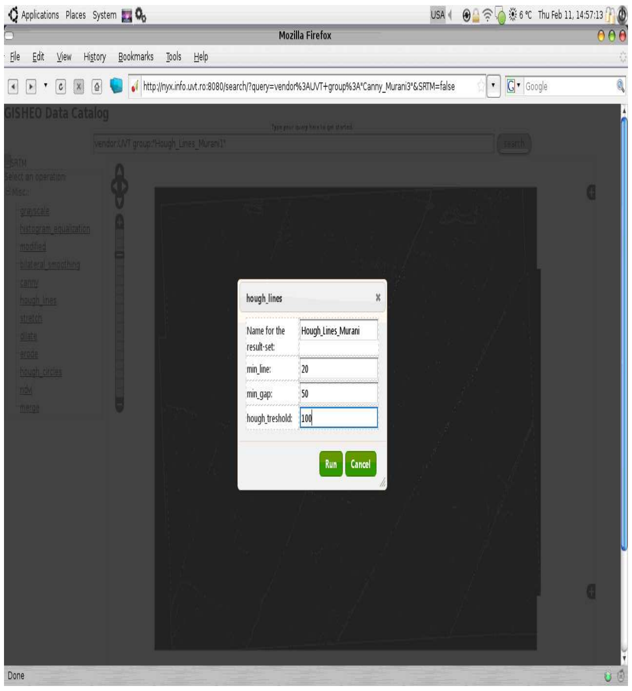

Applications Places System
Thu Feb 11, 14:44:10
Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://nyx.info.uvt.ro:8080/search/?query=vendor%3AUVT+group%3A*Canny_Murani3*&SRTM=false
GISHEO Data Catalog
Type your query here to get started.
vendor:UVT group:*Canny_Murani*
search
SRTM
Select an operation:
Misc.:
grayscale
histogram_equalization
modified
bilateral_smoothing
canny
hough_lines
stretch
dilate
erode
hough_circles
ndvi
merge
Done

Figure 13: Visualizing the result of the Canny edge detector.


Applications Places System
Thu Feb 11, 14:57:13
Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://nyx.info.uvt.ro:8080/search/?query=vendor%3AUVT+group%3A*Canny_Murani3*&SRTM=false
GISHEO Data Catalog
vendor:UVT group:*Hough_Lines_Murani1*
Done

hough_lines

Name for the result-set:	Hough_Lines_Murani
min_line:	20
min_gap:	50
hough_treshold:	100

Run Cancel

Figure 14: Applying the Hough transform on the Canny image.

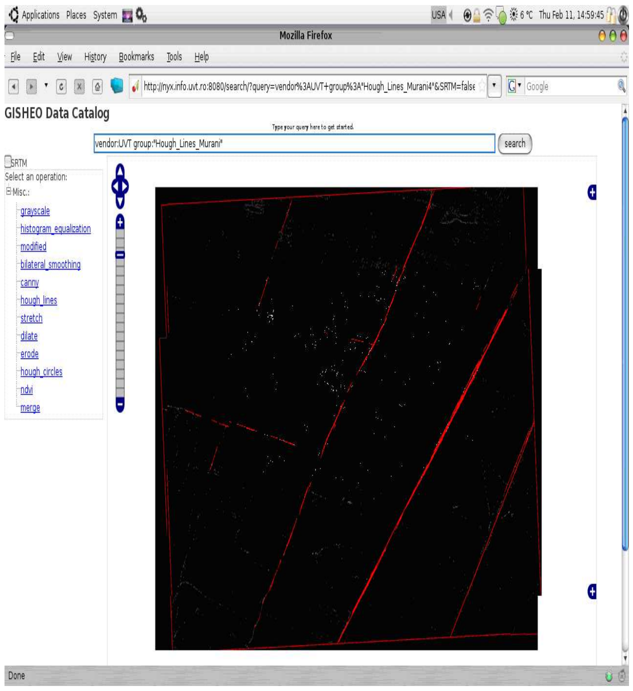

Applications Places System
Thu Feb 11, 14:59:45
Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://nyx.info.uvt.ro:8080/search/?query=vendor%3AUVT+group%3A*Hough_Lines_Murani4*&SRTM=false
GISHEO Data Catalog
Type your query here to get started.
vendor:UVT group:*Hough_Lines_Murani*
search
SRTM
Select an operation:
Misc.:
grayscale
histogram_equalization
modified
bilateral_smoothing
canny
hough_lines
stretch
dilate
erode
hough_circles
ndvi
merge
Done

Figure 15: Visualising the result of the lines detection operation.

ure 8) the importance of selecting a proper sequence of processing steps when a specific study (e.g. the identification of important lines inside an image) is undertaken becomes obvious.

The user has two choices in what concerns the steps required to obtain a final image. By using the interface he/she can either build a workflow made of several basic operations or send basic operations to the server one at a time. In either case the requests are processed by the workflow engine as it treats even basic operations as workflows composed of a single task.

If several data are selected by the user, the actions will be undertaken on all of them in parallel.

5 Conclusions

Starting from a debate of the current issues in satisfying the user and system requirements in Earth observation systems, as well as from the need of training in Earth observation field, a service oriented platform was proposed and described in this paper. It uses the latest technologies both in distributed systems as well as in Earth observation systems and therefore can be considered as a proof of concept of what is currently possible to be done with available technologies.

During the platform design and implementation, innovative solutions were proposed like the custom query language or the specific rule-based language for the workflow engine. While these solutions are not in line with the current standards, they proved to be more efficient in the implementation, as well as to respond better to the requirements of the Earth observation system and to obtain a fast response from the distributed platform.

Acknowledgement

This research is supported by ESA PECS Contract no. 98061 GiSHEO – On Demand Grid Services for High Education and Training in Earth Observation.

References

[1] Aloisio, G., Cafaro, M. (2003). A dynamic Earth observation system. *Parallel Computing* 29 (10), pp. 1357-1362.

[2] Deelman, E., Singh, G., Su, M-H., Blythe, J., Gil, Y., Kesselman, K., Mehta, G., Vahi, K., Berriman, G.B., Good, J., Laity, A., Jacob, J.C., Katz, D.S. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming* 13, pp. 219–237.

[3] eIRG Data Management Task Force (2009). *e-IRG report on data management*. http://www.e-irg.eu.

[4] Forgy, C. (1982). Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artificial Intelligence* 19, pp. 17–37.

[5] Frincu, M.E., Panica, S., Neagul, M., Petcu, D. (2009). Gisheo: On demand Grid service based platform for EO data processing, *Procs. HiperGrid'09*, Politehnica Press, pp. 415-422.

[6] Fusco, L., Cossu, R, Retscher, C. (2008). Open Grid services for Envisat and Earth observation applications. *High Performance Computing in Remote Sensing*, Plaza, A., Chang, C. (Eds.), pp. 237–280.

[7] Hey, T., Tansley, S., Tolle, K. (2009). *The fourth paradigm. Data-itensive scientific discovery*. Microsoft research.

[8] Gasster, S.D., Lee, C.A., Palko, J.W. (2008). Remote sensing Grids: architecture and implementation. *High Performance Computing in Remote Sensing*, Plaza, A., Chang, C. (Eds.), pp. 203–236.

[9] Gorgan D., Stefanut T., Bacu V. (2009). Grid based training environment for Earth observation. *LNCS* 5529, pp. 98–109.

[10] Larson, J.W., Norris, B., Ong, E.T., Bernholdt, D.E., Drake, J.B., Elwasif W.E., Ham, M.W., Rasmussen, C.E., Kumfert, G., Katz, D., Zhou, S., DeLuca, C., Collins, N.S. (2004). Components, the common component architecture, and the climate/weather/ocean community, *Procs. 84th AMS Annual Meeting*, Seattle.

[11] Lee, C, A. (2008). An introduction to Grids for remote sensing applications. *High Performance Computing in Remote Sensing*, Plaza A., Chang C.(Eds.), pp. 183–202.

[12] Muller, R., Greiner, U., Rahm, E. (2004). AGENTWORK: a workflow system supporting rule-based workflow adaptation. *Data & Knowledge Engineering*, 51 (2), pp. 223–256.

[13] Neagul, M., Panica, S., Petcu, D., Zaharie, D., Gorgan, D. (2009). Web and Grid services for training in Earth observation, *Procs. IEEE IDAACS'09*, IEEE Computer Society Press, pp. 241-246

[14] Nemeth, Z., Perez, C., Priol, T. (2005). Workflow enactment based on a chemical metaphor. *Procs. SEFM 2005*, IEEE Computer Society Press, pp. 127-136.

[15] Nico, G., Fusco, L., Linford, J. (2003). Grid technology for the storage and processing of remote sensing data: description of an application. *SPIE* 4881, pp. 677–685.

[16] Oinn, T., Greenwood, M., Addis, M., Apldemir, M.N., Ferris, J., Glover, K., Globe, C., Goderis, A., Hull, D., Marvin, D., Li, P., Lord, P., Popock, M.R., Senger, M., Stevens, R., Wipat, A., Wroe, C. (2006). Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience* 18 (10), pp. 1067–1100.

[17] Panica, S., Neagul, M,, Petcu, D., Stefanut, T., Gorgan, D. (2009). Desiging a Grid-based training platform for Earth observation. *Procs. SYNASC 08*, IEEE Computer Society Press, pp. 394–397.

[18] Paun, G., Grzegorz, R. (2002). A guide to membrane computing. *Theoretical Computer Science* 287 (1), pp. 73–100.

[19] Petcu, D., Challenges of data processing for Earth observation in distributed environments. *SCI* 237, Springer, pp. 9–19.

[20] Petcu, D., Gorgan, D., Pop, F., Tudor, D., Zaharie, D. (2008). Satellite image processing on a Grid-based platform. *International Scientific Journal of Computing* 7 (2), pp. 51–58.

[21] Petcu, D., Zaharie, D., Neagul, M., Panica, S., Frincu, M., Gorgan, D., Stefanut, T., Bacu, V. (2009). Remote sensed image processing on Grids for training in Earth observation. *Image Processing*, Chen Yung-Sheng (ed.), In-Teh, pp. 115–140.

[22] Plaza, A., Plaza, J., Valencia, D. (2006). Ameepar: Parallel morphological algorithm for hyperspectral image classification in heterogeneous NoW. *LNCS* 3391, pp. 888–891.

[23] Plaza, A., Chang, C. (Eds.) (2008), *High Performance Computing in Remote Sensing*. Chapman & Hall/CRC, Taylor & Francis Group, Boca Raton.

[24] Portela, O., Tabasco, A., Brito, F., Goncalves, P. (2008). A Grid enabled infrastructure for Earth observation. *Geophysical Research Abstracts* 10, pp. 1.

[25] Schmuck, F., Haskin, R. (2002). GPFS: A shared-disk file system for large computing clusters, *Procs FAST'02*, pp. 231–244.

[26] Sekiguchi, S. Tanaka, Y. Kojima, I. Yamamoto, N. Yokoyama, S. Tanimura, Y. Nakamura, R. Iwao, K. Tsuchida, S. (2008). Design principles and IT overviews of the GEOGrid. *IEEE Systems Journal* 2 (3), pp. 374–389.

[27] Soltis, S.R., Erickson, G.M., Preslan, K.W., O'Keefe, M.T., Ruwart, T.M. (1997). The Global File System: a file system for shared disk storage. *IEEE Transactions on parallel and distributed systems*.

[28] Szalay, A.S., Blakeley, J.A. (2009). Gray's laws: database-centric computing in science. *The fourth paradigm. Data-itensive scientific discovery*, Hey, T., et al. (eds.), Microsoft research, pp. 5–12.

[29] Teo, Y.M., Tay, S.C., Gozali, J.P. (2003). Distributed geo-rectification of satellite images using Grid computing. *Procs. IPDPS'03*, IEEE Computer Society Press, pp. 152–157.

[30] Votava, P., Nemani, R., Golden, K., Cooke, D., Hernandez, H. (2002). Parallel distributed application framework for Earth science data processing, *Procs. IGARSS 02*, IEEE Press, pp. 717–719.

[31] Wang, J., Sun, X., Xue, Y., Hu, Y., Luo, Y., Wang, Y., Zhong, S., Zhang, A., Tang, J., Cai, G. (2004). Preliminary study on unsupervised classification of remotely sensed images on the Grid. *LNCS* 3039, pp. 981–988.

[32] Weigand, H., Heuvel, W.J.A.M., Hiel, M. (2008). Rule-based service composition and service-oriented business rule management. *Procs. REMOD*, vol. 342, CEUR, pp. 1–12.

[33] Yang, X.J., Chang, Z.M., Zhou, H., Qu, X., Li, C.J. (2004). Services for parallel remote-sensing image processing based on computational Grid. *LNCS* 3252, pp. 689–696.

[34] Yang, C., Guo, D., Ren, Y., Luo, X., Men, J. (2005). The architecture of SIG computing environment and its application to image processing. *LNCS* 3795, pp. 566–572.

[35] Yunck, T., Wilson, B., Braverman, A., Dobinson, E., Fetzer, E. (2008). GENESIS: the general Earth science investigation suite. *Procs. ESTC'04*, http://sciflo.jpl.nasa.gov/.