Chapter 18

FIVE DIMENSION ENVIRONMENTAL DATA RESOURCE BROKERING ON COMPUTATIONAL GRIDS AND SCIENTIFIC CLOUDS

Raffaele Montella and Giulio Giunta
Department of Applied Science
University of Napoli Parthenope, Napoli, Italy

Giuliano Laccetti
Department of Mathematics and Applications
University of Napoli Federico II, Napoli, Italy

1. Introduction

Grid computing has widely evolved over the past years, and its capabilities have found their way even into business products and are no longer relegated to scientific applications. Today, grid computing technology is not restricted to a set of specific grid open source or industrial products, but rather it is comprised of a set of capabilities virtually within any kind of software to create shared and highly collaborative production environments. These environments are focused on computational (workload) capabilities and the integration of information (data) into those computational capabilities. An active grid computing application field is the fully virtualization of scientific instruments in order to increase their availability and decrease operational and maintaining costs. Computational and information grids allow to manage real-world objects in a service-oriented way using industrial world-spread standards.

Boosted by the rapid advance in the multi-core technology, the approach to grid computing has been changing and has evolved towards the right convergence among stability, effectiveness, easy management, convenient behavior and, above all, a better ratio between costs and benefits. This technology led the distributed computing world to a third generation of grids characterized by the increase of the intra site computing power, due to the multi-core technology, and the terabyte up to petabyte scale storage availability. This approach makes convenient the development of virtualization techniques and allows an effective exploitation of massive server factories. Advanced virtual machine software has a slight impact on performance, e.g. using techniques as paravirtualization [17], and it permits to deploy, start, pause, move, stop and un-deploy virtual machines in a high performance, secure and collaborative environment. The software technologies underlying this grid approach, usually defined as cloud computing [3], are quite similar to those developed for the second generation of grids offering features in a software as service way. The true novelty is the dynamical deployment of full virtual machines packed around a software running in a local or remote site. The user asks the resource broker for a service. This service could be even not deployed, but available in a catalogue and then packed into the proper virtual machine and eventually made ready to use. Furthermore, the rapid decline of the cost of highly integrated clusters has spurred the emergence of the data center as the underlying platform for a growing class of data-intensive applications [8,9].

In this scenario metadata augmented data, both stored or produced by an on line acquisition system, have a key role in overcoming what old fashioned computing grids and data grids were in the past. In particular, in a cloud environment the resource is seen as a high level entity, while brokering on application needs is performed automatically using a self describing service approach.

We developed a grid aware component based on a Resource Broker Service implementing a wrap over the OpenDAP (Open source Project for a Network Data Access Protocol) [4] scientific data access protocol and ensuring effective and efficient content distribution on the grid. In a previous work [11] we described the behavior of a Grads Data Distribution Service (GDDS) that could be considered the ancestor of the Five Dimension Data Distribution Service (FDDDS) presented in this work. This service relies on a different legacy OpenDAP engine and provides wider environmental data format capabilities and better performance.

Most of grid middleware implementations assume instruments as data sources. Instruments can be considered off-line and only processed data are usually published, leveraging on common storage facilities as Replica Location, Reliable File Transfer and GridFTP technologies. With this kind of approach, acquired data are concentrated, rather than distributed, with poor benefits in terms of dynamic load balanced access.

In the field of environmental data acquisition, the latter issue can be crucial when a large number of users require an access to the data during extreme weather events, such as hurricanes, flooding or natural disasters as volcanic eruptions, earthquakes and tsunamis.

The challenge of integrating instruments into a grid environment is strategically relevant and it can lead to a more efficient and effective use of the instruments themselves, a reduction of the general overhead and to an improvement of the throughputs. Usually instruments produce huge amount of data that could be stored elastically using cloudily available storage services, reducing management costs and charging users only for resources that they use. From this point of view, data produced by instruments, and stored on cloud computing services, can be advertised on a grid index serviced and provisioned on-demand using OpenDAP based services.

In the following we discuss the implementation and the integration in the Globus Toolkit 4 of FDDDS and we describe how FDDDS and the Resource Broker (RBS) services work. We also focus on the implementation of the mapping component among the resources exposed by GT4 index service and the Condor ClassAd representation, and we provide some examples related to weather forecast model evaluation.

Our RB can answer to queries such as the following one (looking for the best 5D environmental dataset fitting specified requirements):

```
[
      Rank=other.ConnectionSpeed;
      Requirements=other.Type=”dataset”;
      other.Time>=’03:07:2008 00:00:00’ &&
      other.Time<=’09:07:2008 00:00:00’ &&
      other.Lat>=36 &&
      other.Lat<=42 &&
      other.Lon>=8 &&
      other.Lon<=20 &&
      other.Variables=u10m,v10m; &&
      other.DataOrigin==”wrf”
]
```

In this ClassAd data oriented query, the user is looking for a dataset containing the east-component and the north-component of the wind vector at 10m from the sea level (u10m and v10m, respectively). The requested data have to be produced by the Weather and Forecast model and related to the Southern Italy domain area, expressed as latitude and longitude ranges. The best dataset provider is selected by ranking the ConnectionSpeed of the hosts producing the web service. Therefore, if two or more data providers are discovered, the best performing one is selected. Finally, the user submits a specific request to FDDDS for sub-setting (i.e. extracting sub-cubes from data cubes) the desired data. Taking into account local data management policies, the sub-set data could be made available as advertised resource, distributed, replicated, cached or eventually even deleted if not used within a time threshold. In a grid based on cloud environment, FDDDS instances could be dynamically deployed in order to fit actual user needs.

If data are directly produced by instruments and stored using a locally provided or cloud available storage, the developed Instrument Service (InS) adverts data sources on the grid index service, so a ClassAd query as the following is possible:

```
[
      Type="DataConsumer";
      Rank=1/gis.getDistance(
            14.22,40.85,other.Longitude,other.Latitude
            );
      Requirements=
            other.Type=="Instrument" &&
            other.Desc=="WeatherStation" &&
            other.Sensor=="windDir,windSpeed"
]
```

In this query the user is looking for a weather station instrument provided by the data channel acquiring wind direction and speed as close as possible to the point located at longitude 14.22° and latitude 40.85°.

In Section 2, the design and the behavior of our RBS are briefly described. In Section 3 the implementation of GDDS and related issues of performance, flexibility and portability are addressed. In Section 4 the FDDDS architecture is introduced as evolution of previously developed systems. Section 5 describes the Instrument Service design and implementation. Section 6 describes an approach to weather forecast evaluation focusing on a novel algorithm based on time shifted ensembles. Section 7 is dedicated to the integration of data provided by FDDDS and data provided by InS. Finally Section 8 contains some conclusions and highlights on future work.

2. Resource discovery and selection using a resource broker service

The Resource Broker Service [10] is the key component in our grid web service infrastructure. It is based on a Latent Semantic Indices native matchmaking algorithm. We have integrated the resource broker service into the GT4 context, where each resource is published in the Index Service, identified by an EPR (End Point Reference) and selectable using standard GT4 command line tools, such as wsrf-query, and xpath queries.

In the RBS architecture the most important component is the collector process. The generic collector parses index service entry elements and stores them in a host (grid element) oriented format suitable for the native resource brokering algorithm. The collector is charged with managing in the most appropriate way different kinds of resources, such as the Default Index Service, the Managed Job Factory Service and the Reliable File Transfer Factory Service that are parsed and mapped to the Globus.Service.Index, Globus.Service.GRAM and Globus.Service.RTF properties respectively. The collector finds out properties and performs a mapping between one or more properties to new ones and then it stores the results in a local data structure. The collector is fully extensible and customizable using a documented API.

When the Resource Broker service is loaded into the GT4 Web Services container, an instance of the collector component is created and initialized performing queries to the VO main index service.

The RBS accepts queries in native notation; each selection criterion uses expressions like “equal”, “different”, “greater than”, “greater or equal to”, “less than”, “less or equal to” and “max” / “min” to maximize or minimize a property and “dontcare” to ignore a pre-set condition. The broker returns a match by pointing the consumer directly to the selected resource with an End Point Reference (EPR). The selected resource is tagged as claimed to prevent another resource broker query from selecting the same resource, so that a potential overbooking is avoided. The resource remains claimed until a new update event occurs and the resource status reflects its actual behavior. Notification messages are filtered, to avoid a degradation in performance due to the data renewal event management. In order to automatically control the resource lifetime, the effective update availability is not persistent, but it is renewed whenever a status change event exceeds a given threshold.

3. Anagram based GrADS Data Distribution Service

OpenDAP is a community initiative with the goal to create an open-source solution for serving distributed scientific data and for allowing ocean and atmosphere researchers to access environmental data anywhere on the Internet from a wide variety of new and existing programs. The OpenDAP project can capitalize on years of development of data analysis and display packages that use those APIs, allowing users to continue to use programs which they are already familiar with and to develop network versions of commonly used data access Application Program Interface (API) libraries, such as NetCDF, HDF, JGOFS, and others. The OpenDAP architecture relies on a client/server model, with a client that sends requests for data out onto the network to some server answering with the requested data. This is exactly the model used by the world wide web, where clients submit their requests to web servers for data that make up web pages, even though there is still lack of support to the web service technology and to computational data grids. We previously developed a GT4 based web service in order to provide a web grid service fashioned access to environmental data, leveraging on grid features as the Grid Security Infrastructure and the Reliable File Transfer for secure and high performance data management. Our initial design was focused on serving data provided by the Grid Analysis and Display System (GrADS) [2]. This software tool is a free and open source interactive desktop tool, developed in ANSI C and ported on different platforms and operating systems, that is widely used for its easy access, manipulation, and visualization of earth science data stored in various formats as binary, GRIB, NetCDF, or HDF-SDS (Scientific Data Sets). GrADS uses a 5-Dimension data environment: longitude, latitude, vertical level, time and parameters (variables). A plain text descriptor file acts as metadata for both station and gridded data distributed over regular, non-linearly spaced, Gaussian, or variable resolution grids. Different datasets may be integrated and graphically overlaid, with their correct spatial and time registration, through a large variety of graphical techniques and, moreover, the output can be exported in either postscript or image formats.

The GrADS-DODS Server (GDS) combines GrADS and OpenDAP to create an open-source solution for serving distributed scientific data [15]. The GDS provides a wide range of clients with remote dataset access via the OpenDAP protocol and with some analysis tool which has been OpenDAP enabled. Metadata and subsets are retrieved transparently from the server as needed. Data extraction and sub-setting are the main features of GDS, but GDS also provides powerful and multi-stage complex analysis tools, using GrADS software as a computing engine component. In order to reuse as much as possible of the standard GDS distribution, we developed an adapting framework over the Anagram engine, a software component which GDS [16] is based on. Our GDDS implements a “same binary” conservative approach along the way we followed in the grid enabling process of a set of environmental models, as for example the Weather Research and Forecast model (WRF) [6]. Since the Anagram engine heavily builds on module late binding at runtime and a deep integration of the servlet management into the service implementation, we have developed an adapter framework in order to obtain a better behavior when using it as a component, without modifying the legacy distribution. This solution inherits from GDS the need to invoke an external process, i. e. the GrADS executable, for each data access. Translated in the grid enabled version, this means that at every invocation of the operation provider, the host machine has to configure a scratch directory where to work with a local instance of GrADS. This causes a loss of performance, especially in stressfully working conditions in which many consumers ask for data to be extracted by huge dataset, and a lack of portability because of the need to recompile the GrADS source for specific operating system and architecture. Moreover, the number of data types accessible by GrADS is limited. For instance, only few kinds of NetCDF files (following restricted conventions) can be directly accessed (Figure 1).

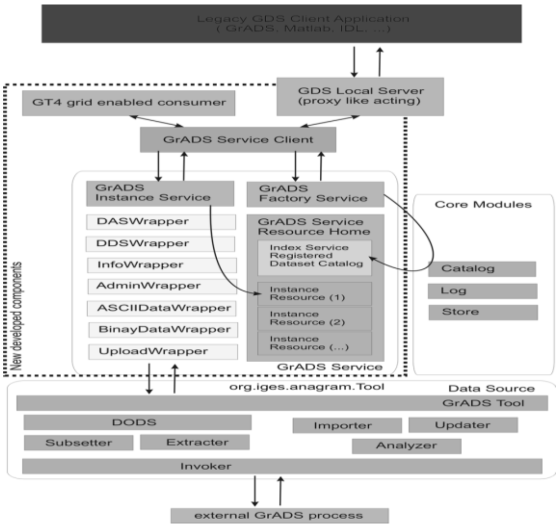


Figure 1. The GrADS Service block diagram

4. Hyrax based Five Dimension Distribution Data Service

The Five Dimensional Data Distribution Service relies on the Hyrax OpenDAP server [5]. Hyrax is a data server which combines the efforts at UCAR/HAO to build a high performance DAP-compliant data server for the Earth System Grid II project with existing software developed by OpenDAP. This server is based on the Java servlet mechanism to manage web requests from DAP format-specific software. This approach improves the performance for small sized data requests.

The servlet front end, instead of launching an external local process as in GDS, looks at each request and formulates a query to a second server, Back End Server (BES), which may be or may be not on the same machine. BES handles the reading of the selected data from the data storage and returns DAP-compliant responses to the front end, which, in turn, could send these responses back to the requestor or might process them to build more complex responses. We notice that this architecture makes possible a better integration in a web grid service ecosystem. In implementing FDDDS we had to extend the client OpenDAP class APIs in order to completely disjoin them from the front end and to provide the needed interfaces to the grid.

We have completely integrated the Back End Service into the web service environment, by reusing any kind of configuration files within the GT4 service architecture. In this way the migration from the classic client server OpenDAP protocol based architecture to the service oriented grid environment application occurs in a really soft fashion.

FDDDS has a self decorating dataset metadata which provide a grid oriented data representation and enforce the interaction and the collaboration among different and geographically spread components, such as instruments, and resource brokers. The native BES provides security tools based on X509 certificates. In our implementation we relied on this feature to ensure the access to the BES only from the local host machine and from FDDDS, delegating to this component and to the underlying GSI the task of authentication, authorization and encryption. This allows that only certificates that have been authenticated by grid users can perform OpenDAP operations including new dataset uploading and administration management.

When FDDDS is loaded into the container, it contacts the BES for retrieving the dataset list. For each dataset a web service resource is created, with the properties extracted from the dataset metadata and then published on a specified Index Service. Then the RBS collector component can carry out a ClassAd representation of the resource involved in the matchmaking process. If a suitable dataset resource is found, the resource broker returns the EPR referencing to the best FDDDS service instance, which is selected taking into account the network behavior between the consumer and the producer endpoints. Then the consumer invokes the getData, getDAS or getDDX to retrieve dataset metadata while getData retrieves data in binary format. The getData operation provider accepts data selection and subsetting parameters via a standard OpenDAP query string specifying the variable name, the running dimensions and the step intervals. In the same way, the getData operation provider allows to perform a sort of remote analysis, using the BES engine and specifying datasets and expressions, as well as to choose the (to be returned) data format (DAP or NetCDF).

FDDDS uses SOAP messages only for an operation provider invocation and for returning computed results, while for transferring data a dedicated transfer service is invoked. This prevents a decrease of performance that can be caused by the transport protocol, especially when the requested data size is very large, as in the case of wide space and temporal domains with many variables. The getData operation provider returns an EPR to the resource associated with the result of the sub-setting and allocated in a temporary storage area. It follows that efficient grid oriented file transfer protocols, as GridFTP, can be used to improve the performance of the distribution system. The sub-set data can be made available for request by advertising them on the Index service. Tracking the log for requested data, the RB notifies FDDDS to distribute a popular in queries sub-set dataset over the grid. FDDDS has in charge the management of such a dynamically created dataset, empting caches and destroying it if needed.

The use of the OpenDAP protocol is very common in the environmental scientists community because of a great amount of compliant applications. FDDDS precludes a direct use of distributed data unless upgraded versions of such applications would provide a full WSRF compliant grid web service access. However, this can be achieved through the use of a local server, based on the original Hyrax front end acting as a proxy interface between OpenDAP server compliant applications (such as IDL, Matlab, GrADS, Ferret and more) and the grid world, represented by the resource broker service, the FDDDS service and the Grid Security Infrastructure.

This proxy-like acting local server accepts connections only from the loop-back network interface and translates the standard OpenDAP query string into a direct or resource brokered grid interaction. The local OpenDAP server implements an FDDDS client which handles certificates, EPR and resource broker requests in a fully transparent way (Figure 2).

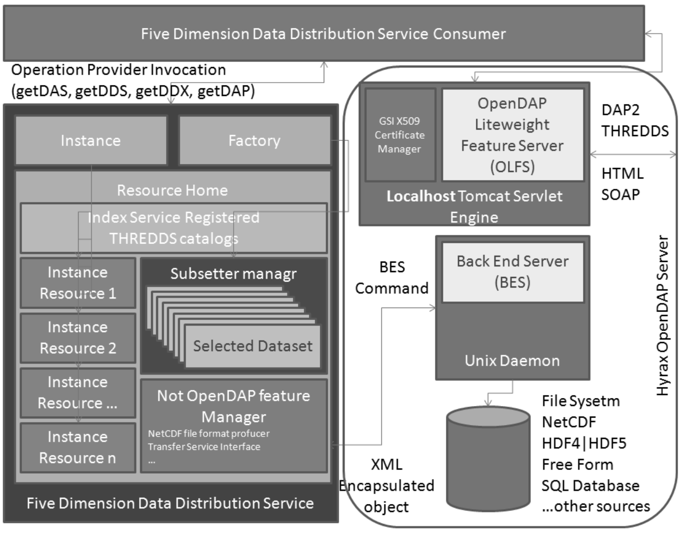

Five Dimension Data Distribution Service Consumer
Operation Provider Invocation
(getDAS, getDDS, getDDX, getDAP)
Instance
Factory
Resource Home
Index Service Registered THREDDS catalogs
Instance Resource 1
Instance Resource 2
Instance Resource ...
Instance Resource n
Subsetter managr
Selected Dataset
Not OpenDAP feature Manager
NetCDF file format profucer
Transfer Service Interface
...
Five Dimension Data Distribution Service
GSI X509 Certificate Manager
OpenDAP Liteweight Feature Server (OLFS)
Localhost Tomcat Servlet Engine
DAP2
THREDDS
HTML
SOAP
BES Command
Back End Server (BES)
Unix Daemon
XML Encapsulated object
File Sysetm
NetCDF
HDF4|HDF5
Free Form
SQL Database
...other sources
Hyrax OpenDAP Server

Figure 2. The Five Dimension Data Distribution Service block diagram

In a performance analysis, we compared the Grads Data Distribution Service with the Five Dimension Data Distribution Service by setting up a testbed of real data produced by weather forecast operational runs.

In the current setup, each WRF model run produces a 11 gigabytes data file storing the coarse domain forecast results. The size of the data set on the fine domain amounts to 2 gigabytes for each model's run. The coarse domain output file stores a five-dimension data set, comprised of, 19 vertical levels, 655x594 cells, 27 2D and 3D variables, and 144 time steps.

In order to test the services, a getData operation provider is invoked by requesting the u10m variable for all time steps (144, six days, one time step per hour) and subsetting an area of 2^n square cells with $3 \leq n \leq 9$. The invocation time includes the subsetting and the data transfer via gridFTP. In this conditions, for small sized subsets the performance of both services are comparable, while for more demanding subsetting tasks the FDDDS shows a higher efficiency (Figure 3).

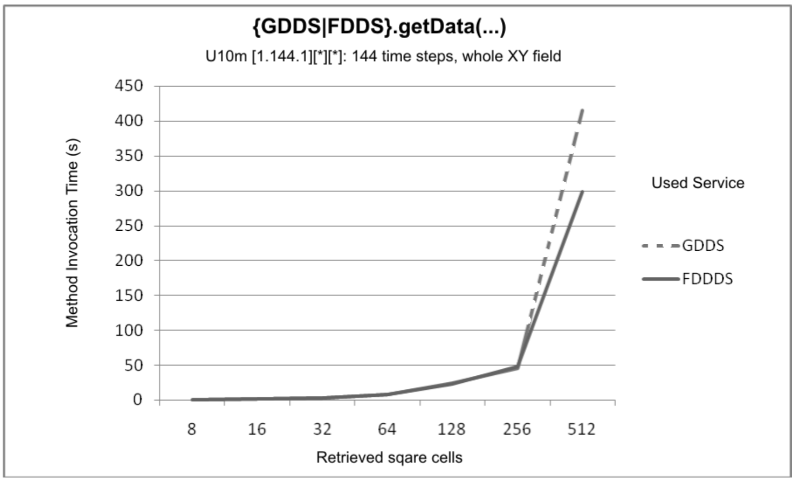


Figure 3. Performance analysis results

5. Design and implementation of an Instrument Service for NetCDF data acquisition

Our ultimate goal is the instruments sharing on computing grids. In order to abstract different kinds of instruments (such as weather stations, surface current radars and wind profilers), with a wide variety of hardware interfaces and acquisition data rates, a suitable design leveraging on a plug-in based framework is needed.. We have developed an Abstract Instrument Framework (AIF) to decouple the acquisition Java interface from the grid middleware technology [12] and to introduce a streaming protocol after the SOAP interaction started the communication. Environmental data acquisition instruments interact with their proxy hardware in different ways. Weather station data loggers could work in real-time or in batch mode depending on the type of data link. Wind profilers and surface currents radars operate in a similar way using a power workstation as a proxy machine. Wind profile equipment workstations are a sort of a very powerful data logger in which vertical profile integrated values are stored. The sea surface current instruments use workstations acting as an automatic integration and post processing computing node producing surface time integrated values. We used a layered approach relying on virtualization. The instrument framework defines the behavior of a generic instrument built assembling sensors for data acquisition and actuators for instrument handling leveraging on the component abstraction technique.

Each abstract component leaves unimplemented each low level hardware interaction with the instrument and it provides tools for data delivery as SOAP typed values or streaming protocols.

Weather stations are usually equipped with an autonomous simple data logger with remote interaction capabilities for instrument setup, sensor calibration and data retrieving in both batch and real-time mode. We operate different kind of weather stations built on different sensors and data loggers. Weather stations fit straightforwardly in this framework model, because they are a collection of sensors. Data channels may be operated in two ways, depending on the hardware sensor and the required measurement. The real sensor components implement the real data access either with a real-time request or retrieving them form a data logger which specifies the type and the nature of the acquired data.

Typically sea surface currents radar control stations are powerful workstations where all operations on data and sensors (remote antennas) are managed by a proprietary closed source software. In this case there is no direct access to the sensor, but only to the automatically post-processed data. The sea surface currents radar sensor wraps provide surface data averaged in time e localized on a geo-referenced matrix. The virtualization is obtained through interfacing real sensor components related to sea current speed and direction directly with data produced by the standard equipment. The wind profile instrument works in a very similar manner and the profiled data, averaged in time and distributed along the vertical axis, are stored by the proxy workstation acting as an enhanced data logger which provides data transfer tools based on standard protocols.

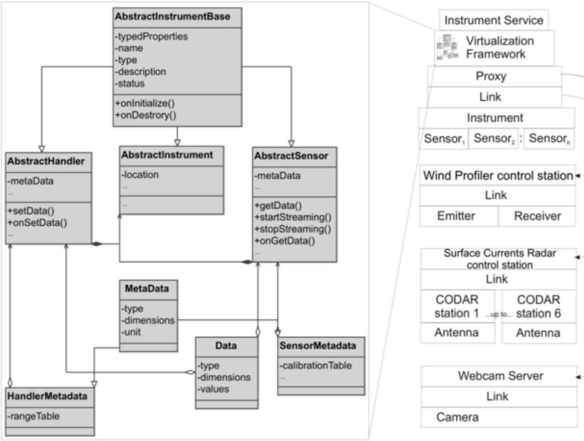


Figure 4. The Abstract Instrument Framework class diagram
and the Instrument Service architecture

The Instrument Service exposes operation providers to start and stop data streaming using a specified protocol (published, for each sensor, on the index service as a metadata field) with a protocol plug in interface designed to easily customize the data distribution. In this case the standard SOAP message exchanging between the web service consumer and the web service producer is performed only to control data transmission, while the actual data transfer is carried out using a more efficient protocol.

The Instrument Service automatically publishes Instruments, Sensors and Handlers metadata - and even data - on the GT4 Index Service, so that a grid service consumer could leverage on the WSRF Notification feature to interact with a data acquisition device. RBS component interacts with the Index Service and allows any instrument shared on the grid to participate in the dynamic resource allocation process. An instrument, as any grid shared data source FDDDS, is discoverable and selectable via ClassAd queries.

A matchmaker algorithm can match ClassAds to different kinds of entities in a specified manner: The expression `other.Type=Instrument` is evaluated as true if the ClassAd which is matched with contains a property named `Type` with the string value equal to `Instrument`. In the matchmaking protocol, two ClassAds match if each one has an attribute requirement that evaluates to true in the context of the other ClassAd; then matched grid elements activate a separate claiming protocol to confirm the match and to establish the allocation of resources.

The representation of an instrument as a grid resource is automatically done by RBS and considered as a sort of dynamically varying stored data. The following example shows the ClassAd representation of a weather station:

```
[
      Type="Instrument";
      Desc="WeatherStation";
      Area="Napoli";
      Longitude="14.32";
      Latitude="40.50";
      TimeStamp="23-12-2007 03:30:00";
      Sensor="windDir,windSpeed,airTemp,airPressure";
      Values="335,3.280,14,1013";
      Units="°N,ms-1,°K,HPa";
      Rank=1;
      Requirements=other.Type=="DataConsumer"
]
```

This instrument is defined as a weather station resource in the Napoli area located at longitude 14.32°, latitude 40.50°. The Instrument Service framework publishes the Sensor metadata, then it collects the instrument features and finally publishes on the grid index service the instrument geographical position, the data currently acquired by each sensor and the actuators feedback parameters.

6. A weather forecast quality evaluation scenario

The assessment of the quality of an operational weather forecasting model is a crucial process for stating the reliability of the overall operational forecasting system.

A forecast is an estimate of the future state of the atmosphere. It is created by estimating the current state of the atmosphere using observations, and then calculating how this state will evolve in time by means of a numerical weather prediction computer model. As the atmosphere is a chaotic system, very small errors in its initial state can lead to large errors in the forecast. This means that one cannot create a perfect forecast system because it is not possible to observe every detail of the atmosphere's initial state. Tiny errors in the initial state will be amplified, so there is always a limit to how far ahead we can predict any detail.

In a weather forecast operational scenario, the ensemble methodology is a common technique for the assessment of the model behavior by means of a sensitivity analysis on initial and boundary conditions. To test how small discrepancies in the initial conditions may affect the outcome of the forecast, an ensemble system can be used to produce many forecasts. The complete set of forecasts is referred to as the ensemble, and any individual forecast as an ensemble member.

The ensemble forecasts give a much better idea on which weather events may occur at a particular time. By comparing the ensemble members the forecaster can decide how likely a particular weather event may be. Shortly, if the forecasts exhibit a large divergence, then one may conclude that there is a large uncertainty about the future weather, while if most of the forecasts are similar then much confidence can be taken in predicting a particular event.

Ensemble modeling requires high computational power since a large number of simulations must be run concurrently, so that grid computing technology is a key and widely adopted solution [18].

Technical issues related to our grid application for operational weather forecasts are discussed in [1]. Briefly, a Job Flow Scheduler Service (JFSS) orchestrates the grid application behavior; it relies on a Job Flow Description Language which specifies the relations among web service consumers and uses the RBS for dynamic resource allocation, data and instrument discovery and selection. With the Five Dimension Data Distribution Service, we set up a grid application which integrates a grid enabled WRF weather model with a set of grid based validating tools leveraging on FDDDS in a operational scenario.

We have developed a weather forecast quality control and validation algorithm aimed at saving computing time and avoiding multiple model runs, as in a standard ensemble simulation. The algorithm uses an approach we called "Time Shift Ensemble". The WRF model is run each day, with initial and boundary conditions provided by an NCEP global model. The output of a daily simulation is a forecast on the next six simulated days. Thus the first simulated day (hours ranging from 0 to 23) has a set of six forecasted data: the 0h..23h dataset generated by the current day run; the 24h..47h dataset returned by the previous day run and so on, up to the dataset produced by the run of 6th previous day.
We have considered the Geo-potential height (GpH) computed by the model at 500 HPa level as a good atmosphere status descriptor parameter, as usual in the meteorology community [14]. The GpH is a vertical "gravity-adjusted" coordinate that corrects the geometric height (elevation above mean sea level) using the variation of gravity with latitude and elevation. GpH is usually referred to a pressure level, which would correspond to the GpH necessary to reach the given pressure.
At each point of the spatial domain, we compute the difference between the hourly forecasted GpH produced by the current run and the GpH values produced by each of the five simulations that we ran at the previous days. These set of values can be considered as the discrepancies of the results of an ensemble built on 6 model's runs with different initial/boundary conditions. The quality of the forecast can be deduced from the behavior of those time series [13]. To be more precise, let $F_{s,f}$ denote the GpH forecast produced by the run at day s (s=0 for the current day, s=-1 the previous one, and so on) for the forecasted day f (f = 0 denotes the first day of the simulation and f=5 the last simulated day, with a total simulated time equal to 144 hours). The current day is forecasted for the first time as the last day of the run executed at the day -5 ($F_{-5,5}$), the second time as $F_{-4,4}$ up to the simulation $F_{-1,1}$ (Figure 5).

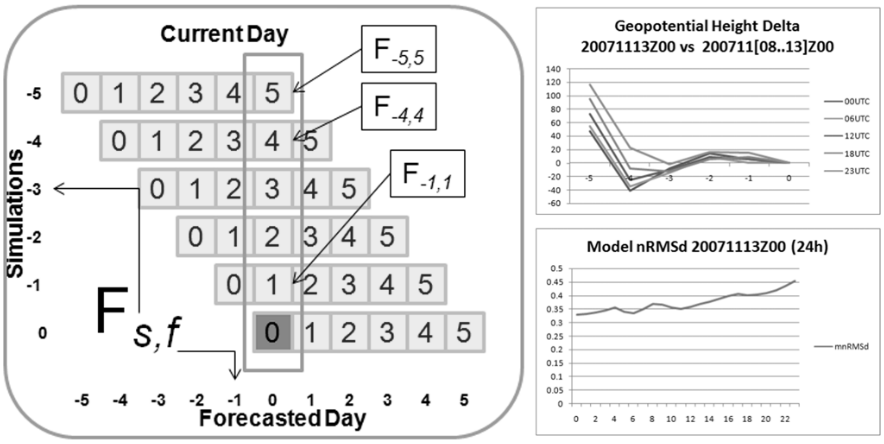


Figure 5. The validation algorithm and some results

Figure 5 (right-up) also reports the behavior of the five time series of the computed discrepancies (for a given simulated day and at a specific spatial point) in the case of a reliable simulation, while the right-down side of Figure 5 shows the related time series of normalized Root Mean Square error (nRMSd) between the $F_{-5..-1}$ and the current day run's values F_0.

7. Implementation of the grid application

Here we give some insight into relevant implementation details of our grid application. The application workflow starts with an invocation to FDDDS which collects initial and boundary conditions needed for the initialization of WRF (Figure 6).

Data are downloaded from the NCEP ftp server using an automatic process. Since forecasts have to be computed each day on the same domain, another invocation of FDDDS allows to download terrain data previously produced by the GeoGrid Service (GGS), which is a wrap over the WRF geogrid module. Then the Ungrib Service (UGRBS) is invoked to convert initial and boundary conditions in a file format suitable for the metgrid WRF module. The process continues with a call to the MetGrid Service (MGS) which interpolates initial and boundary conditions over the domain and, finally, to the Real Service (RS), that is wrapped on the Real WRF module and that prepares data for the WRF Service, a service shell over the WRF main module. The WRF main model produces results that are stored using FDDDS. This component makes the produced data available on the grid, either using the RBS or directly through the EPR. To perform our validation tests we set up an application with only one domain covering all Europe and part of northern Africa, with 256x120 square cells (30x30 Km^2) centered at longitude 14.22° and latitude 40.85°.

The validation process starts collecting pre-processed data related to the same current simulated day and it invokes RBS and FDDDS to get information on where the data are available. These data are used by the Validation Service (VALS), that wraps around a tool implementing the evaluation algorithm described in the previous section. The VALS component returns a map of the Geopotential Height nRMSd and stores the computed data in a FDDS service selected via RBS.

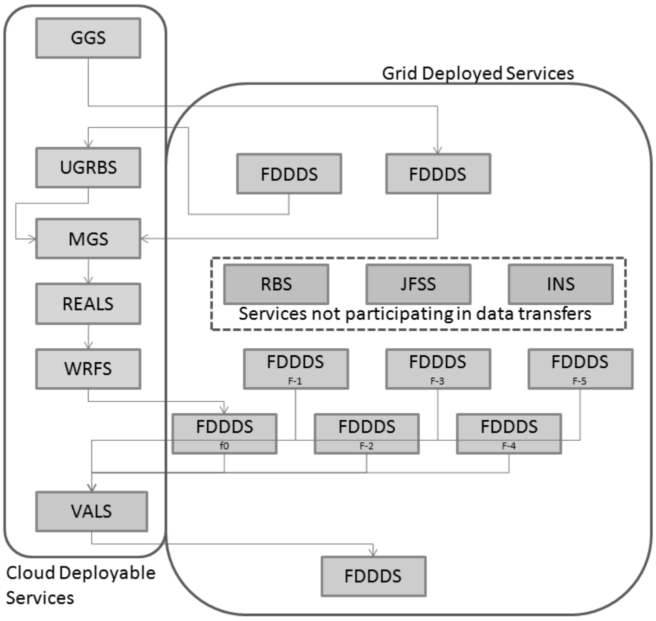


Figure 6. The grid/cloud application schema

Figure 6 shows the application architecture from a deployment point of view. The services classified as deployable on a cloud computing infrastructure are the ones characterized by large need of computing power. These services can be conveniently managed by a batch execution using virtual machines [19], where the computing power, required by the web services underlying the application modules of the grid application (WRF: GGS, UGRBS, REALS, WRFS; Evaluation: VALS, METS), is dynamically allocated on demand and eventually released. We deployed the computing power demanding services on a public infrastructure as a service oriented computing cloud.

The services classified as deployed on the grid are data oriented. In this case the need for the elasticity provided by the cloud infrastructure is addressed to the computing intensive part of the application.

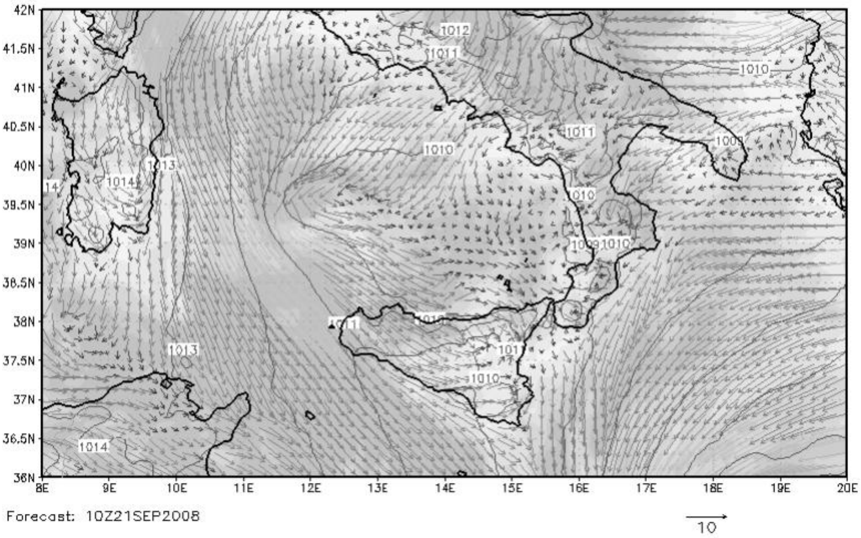
The use of the cloud infrastructure is convenient because resources, i.e. virtual machines, have to be instanced only for the needed time. Using the cloud for data storing is technically possible, but some drawbacks can raise when a large amount of data are allocated permanently in the cloud without the need of storage elasticity; we notice that in our application the needed amount of storage could be estimated in advance. The evaluation algorithm requires 6 files of about 11 gigabyte each. After any run of the model, the oldest dataset is deleted and the last produced dataset is added to the storage. However, the application requirements for the storage result in a total cost of ownership which is less critical than the requirement for computing power, so that a self hosted storage turns out to be more effective than a cloud storage solution.

8. Conclusions and future work

We have described some recent results in distributing 5D environmental data using GT4 WSRF compliant tools in a high performance grid environment. The application has been deployed in an Infrastructure as a Service public cloud. We have briefly discussed how the computing and storage resources can be made available on a computing grid or dynamically allocated on a computing cloud. In particular, the application we discussed here has a critical issue in computing power needs, so that the WRF related services can be deployed in the cloud. On the other hand, storage does not call for an elastic solution and hosting data on the cloud appeared unsuitable.

We have shown that a suitable integration of legacy software, i.e. the Back End Server of the Hyrax OpenDap server suite and the gridFtp parallel file transfer protocol, can improve the operational throughput and can allow a useful, on demand data sub-setting and data collecting process. The Five Dimension Distribution Service relies on a previously designed resource broker collector and matchmaking algorithm, which use the grid ClassAd notation automatically extracted from metadata. This approach also opens a wide range of applications in the field of the environmental model validation, where issues like component integration and security are crucial since a validation algorithm must involve several community tools in order to achieve scientific relevance.

We are aware that a more extensive performance analysis of the application has to be performed, for instance regarding the cache behavior in a real grid environment and the effect on the overall performance of the FDDDS dynamical deployment on a cloud system. Our next goal is to develop a GT4 WSRF compliant web service, wrapping the MET [7] model evaluation tool. The MET offers grid-to-point, grid-to-grid and advanced spatial forecast verification techniques in an unified and modular toolkit, that builds on capabilities available in other verification systems. Tools provided by MET can be grouped by function to describe the overall structure of the MET: data handling, statistical calculations and data analysis. This architecture is suitable for our web service infrastructure in which grid applications are straightforwardly arranged using the JFDL or other similar tools. The idea is to build up a double evaluation system based on VALS and on the service wrapped over MET, which can be able to operate with observed data either downloaded from distribution services or directly sampled via InS, and with data stored using FDDDS.



Forecast: 10Z21SEP2008

Universita' degli Studi di Napoli "Parthenope"

Figure 7. Weather forecast on a Southern Italy domain

In an operational scenario, such as that one producing the results shown in Figure 7, the model evaluation has to be performed automatically in order to provide thematic maps about the forecast’s confidence. This result could be achieved using machine learning algorithms, which can return decision support rules.. The choice of the most suitable classification algorithm and the related issues of its tuning and assessment will be addressed in a future work.

References

1. Ascione I., G. Giunta, R. Montella, P. Mariani, A. Riccio. A Grid Computing Based Virtual Laboratory for Environmental Simulations. Proceedings of 12th International Euro-Par 2006, Dresden, Germany, August/September 2006. LNCS 4128, Springer 2006
2. Doty, B.E. and J.L. Kinter III. Geophysical Data Analysis and Visualization using GrADS. Visualization Techniques in Space and Atmospheric Sciences, eds. E.P. Szuszczewicz and J.H. Bredekamp, 1995, NASA, Washington, D.C., 209-219.
3. Foster, I., T. Freeman, K. Keahey, D. Scheftner, B. Sotomayor, X. Zhang. Virtual Clusters for Grid Communities. CCGRID 2006, Singapore. May 2006
4. Gallagher J., N. Potter, T. Sgouros. DAP Data Model Specification DRAFT. www.opendap.org. November 6, 2004 Rev.: 1.68

5. Gallagher J., N. Potter, P. West, J. Garcia and P. Fox. OPeNDAP's Server4: Building a High Performance Data Server for the DAP Using Existing Software. AGU meeting in San Francisco. 2006.
6. Giunta G., G. Laccetti, R. Montella, A grid-based service oriented environmental modeling laboratory for research and production applications, chapter of Parallel Processing and Applied Mathematics 2007 (R. Wyrzykowski, J. Dongarra, K. Karczewski, J. Wasniewski, editors), Lecture Notes in Computer Science n. 4967, Springer Verlag, 2008, pagg. 951-960.
7. Holland L., J. H. Gotway, B. Brown, and R. Bullock. A Toolkit For Model Evaluation. National Center for Atmospheric Research Boulder, CO 80307
8. Llorente I. M. Towards a New Model for the Infrastructure Grid, Panel From Grids to Cloud Services in the International Advanced Research Workshop on High Performance Computing and Grids, Cetraro, Italy, July 2008
9. Llorente I. M. Cloud Computing for On-demand Resource Provisioning, International Advanced Research Workshop on High Performance Computing and Grids, Cetraro, Italy, July 2008
10. Montella R.. Development of a GT4-based Resource Broker Service: an application to on-demand weather and marine forecasting, volume LNCS 4459 of LNCS. Springer, May 2007.
11. Montella R., G. Giunta, and A. Riccio. Using grid computing based components in on demand environmental data delivery. ACM proceedings about Upgrade Content Network HPDC2008 workshop. Monterey Bay - CA, USA, June 2007.
12. Montella R., G. Agrillo, R. Di Lauro. Abstract Instrument Framework: Java interface for instrument abstraction. DSA Technical Report. Napoli. April 2008
13. Montella R., G. Agrillo, D. Mastrangelo, M. Menna. A Globus Toolkit 4 Based Instrument Service For Environmental Data Acquisition And Distribution. Proceedings of Upgrade Content Workshop HPDC2008. Boston. June 2008
14. Warren R. Development and illustrative outputs of the community integrated assessment system (cias), a multi-institutional modular integrated assessment approach for modelling climate change. Environmental Modelleng and Software, 20:1–19, 2007.
15. Wielgosz J. and J. A. B. Doty. The grads-dods server: an open-source tool for distributed data access and analysis. 2003.
16. Wielgosz J. Anagram: a modular java framework for high-performance scientific data servers. 2004.
17. Youseff L., R. Wolski, B. Gorda, C. Krintz. Paravirtualization for HPC Systems XHPC. Workshop on XEN in High-Performance Cluster and Grid Computing, Dec. 2006
18. Ramakrishnan L., B. Blanton, H. Lander, R. Luettich, D. Reed, and S. Thorpe. Real-time storm surge ensemble modeling in a grid environment. 2006.
19. Sotomayor B., K. Keahey, I. Foster. Combining Batch Execution and Leasing Using Virtual Machines, ACM/IEEE International Symposium on High Per-

formance Distributed Computing 2008 (HPDC 2008), Boston, MA. June 2008.

Index terms (alphabetically):

Cloud computing
Environmenta data
Grid computing
Resource Brokering
Weather Forecast
Web Service