School of Engineering & Design

Electronic & Computer Engineering

MSc in Data Communications Systems

Grid Monitoring

Theofylaktos Papapanagiotou

Dr.Paul Kyberd

March 2011

A Dissertation submitted in partial fulfillment of the

requirements for the degree of Master of Science

School of Engineering & Design

Electronic & Computer Engineering

MSc in Data Communications Systems

Grid Monitoring

Student's name: Theofylaktos Papapanagiotou

Signature of student:

Declaration: I have read and I understand the MSc dissertation guidelines on plagiarism and cheating, and I certify that this submission fully complies with these guidelines.

Abstract

EGI has replaced EGEE as the main European Grid Initiative. Multi Level Monitoring architecture suggested central points in regional level, where metrics from each information system of the grid will be aggregated. MyEGI, MyEGEE and Nagios replace SAM in availability monitoring. Performance monitoring is approached using Ganglia as the source of performance metrics, and WSRF/BDII as the carier of that information.

Both Globus and gLite resource brokers come with their favorite information service. Grid Monitoring Architecture suggests the model by which the information should be discovered and transfered. Monitoring and Discovery Service is responsible to provide that information. Two different methods exist about the way that the information is transfered, BDII and WSRF. Both implement the Glue schema, support Information Providers, and export the metrics in standard formats.

Linux kernel load average is the main metric that is taken by Ganglia, and through the information providers, is passed to Nagios, LDAP and the container that supports the WSRF. Ganglia distribute the metrics to all its nodes using XDR over the multicast network. Nagios stores the historical data using NDOUtils to its database repository. Ganglia python client is integrated with BDII LDAP to provide real-time metrics of Gmond to information consumers. WSRF transforms through XSLT the XML taken by Gmond and passes it to the framework's Index to be discovered and aggragated.S

Finally, data are represented in graphs using RRDtool through pnp4nagios plugin of Nagios system. LDAP queries using PHP provide the real-time data from BDII. DOM library of PHP is used to parse data using XPath queries in WebMDS frontend of WSRF.

Contents

Listings

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Context

Performance monitoring of a grid is a key part in grid computing. Based on the reports of grid performance, decisions on capacity planning are made. Visualization of performance status in different levels helps scientists and managers focus on the exact point of the infrastructure where a bottleneck on service exists. Current interfaces deliver performance graphs without following the standard topology schema introduced by the grid information system.

WSRF aggregation framework and GLUE schema are examined, to understand the gathering process of metrics. BDII LDAP is also examined, as the carrier of the information data. Ganglia's hierarchical delegation to create manageable monitoring domains is an important aspect. Performance metrics are taken using Linux kernel's load average. Performance in the aspect of how many jobs are served by each site is not examined in this project. This project examines which of the two information services better transfer the metrics for the multi level monitoring architecture.

A starting point was to build a lab to gather performance data and start working on the development of the integration parts. It is assumed that the environment is a grid site, that already has the components needed to work together. Ganglia daemons on each node, presented by the GLUE schema on site BDII, Nagios/MyEGI monitoring frameworks. A web interface is avail-

able to present the work of the integration of Ganglia into Nagios/MyEGI, BDII and WSRF.

1.2 Aims & Objectives

This project aims to evaluate grid performance monitoring tools, and information services to distribute that data. It will follow the chain of data generation, to distribution, provision, transformation and display of data using some custom built interfaces.

Using a testbed, Ganglia agent should be installed in every worker node of the Computing Element. Globus and gLite middlewares should also be installed, to provide the Information Services for data aggregation and transfer.

Resource Providers should be integrated on BDII and WSRF Information Services, to get, parse/transform and deliver the data to the front interface. Authentication mechanism is not part of this project, but in order to respect the procedures, a wrapping interface such as WebMDS should be installed. Standards and specifications about data organization in the information system such as the Glue schema will be covered.

Information Services should be selected for the different levels of the Multi Level Monitoring infrastucture, such as site, regional or top level. Nagios and ganglia integration should also be evaluated.

By taking metrics there is an effect on performance of the monitored system, that may be considered. Aggregation methods before displaying the metrics, should also be used.

1.3 Organization

1.3.1 Tools

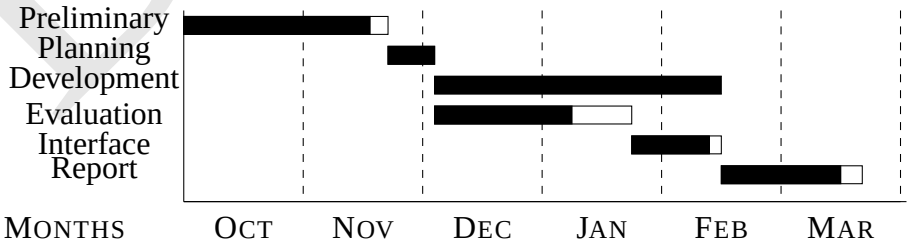
This project was developed in LaTeXusing Vi editor, all figures are vector graphics designed in Dia. Its releases may be found in Google Code, where

Mercurial was used for source control. Citation management through Mendeley software. Papers obtained by becoming member of IEEE, ACM and USENIX. A grid site testbed and tools to study existing monitoring tools was build in Operating Systems Laboratory of Technological Education Institute of Piraeus.

1.3.2 Time-plan (Gantt Chart)

Task	Start date	End date	Days
Preliminary	09/29/10	10/24/10	20
- Identify Concepts	09/29/10	10/08/10	8
- Gain Access	10/08/10	10/24/10	12
Planning	11/12/10	12/04/10	17
- Explore existing technologies	11/12/10	11/28/10	12
- Write Interim Report	11/28/10	12/04/10	5
Experimental-Development	12/04/10	02/14/11	51
- Evaluate performance monitoring tools	12/04/10	12/25/10	15
- Information Services evaluation	12/17/10	12/29/10	8
- Build a testbed and test cases	12/29/10	31/01/11	20
- Develop interface	01/02/11	02/14/11	14
Report	02/16/11	03/29/11	32
- Begin Writing	02/17/11	03/01/11	11
- Submit Draft & Make Changes	03/01/11	03/14/11	9
- Prepare Final	03/14/11	03/29/11	11

Table 1.1: Key activities necessary to complete the project



Chapter 2

Literature Review

2.1 Grid Computing

Grid computing [1] is the technology innovation in high performance computing. A large number of scientists work on the operations of this huge co-operative project of EU. Monitoring & information architecture [2] has been standardized in the initial state of that project, to succeed in building a large-scale production grid of 150.000 cores. Use of grid computing nowadays takes place in academic and research environments. Also, applications in industry-based needs such as promising Power Grid control [3] are emerging.

Grid computing may be the infrastructure over which Cloud Computing may reside. Cloud computing promise that it will change how services are developed, deployed and managed. The elastic demands of education and research community is a good place where cloud computing may be developed. Many datacenters all over Europe which are currently serving grid computing infrastructure for LHC, could later share the resources to help some other big academic projects scale up as needed.

2.2 Resource Brokers

Resource Brokers [4] were developed to manage the workload on Computer elements and Resource elements. Globus is a non-service based RB, and gLite

RB which is service based. A Workload Management System (WMS) exists in gLite to do the distribution and management of the Computing and Storage oriented tasks.

The use of information system is based on the equivalent middleware that resource brokers rely on. From resource broker's point of view, the relevant information is the data store and query. There are two main categories of information systems in middlewares. The Directory-based and the Service-based. They are used for resource mapping by the brokers when they access the resource data.

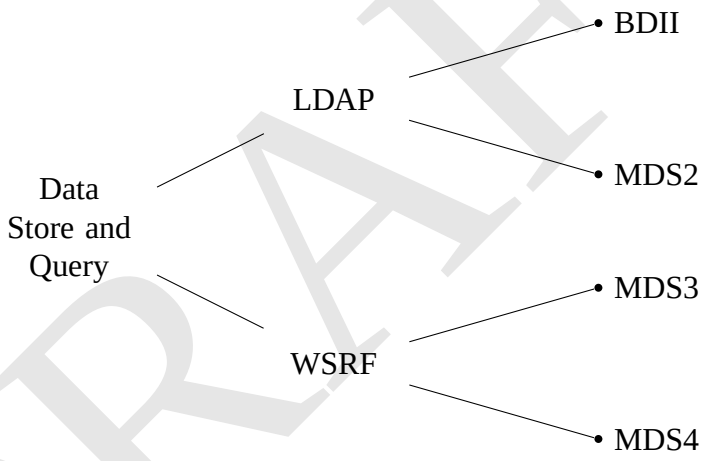


Figure 2.1: Grid Resource Brokers grouped by Information Systems[4]

2.2.1 Globus

Globus Toolkit is an open source toolkit used to build grids. It provides standards such as OGSA, OGSI, WSRF and GSI, and the implementations of OGF protocols such as MDS and GRAM.

Monitoring and Discovery Service (MDS) is part of Globus Toolkit, and provides the information for the availability and status of grid resources. As a suite of Web Services, it offers a set of components that help to the discovery and monitoring of the resources that are available to a Virtual Organization.

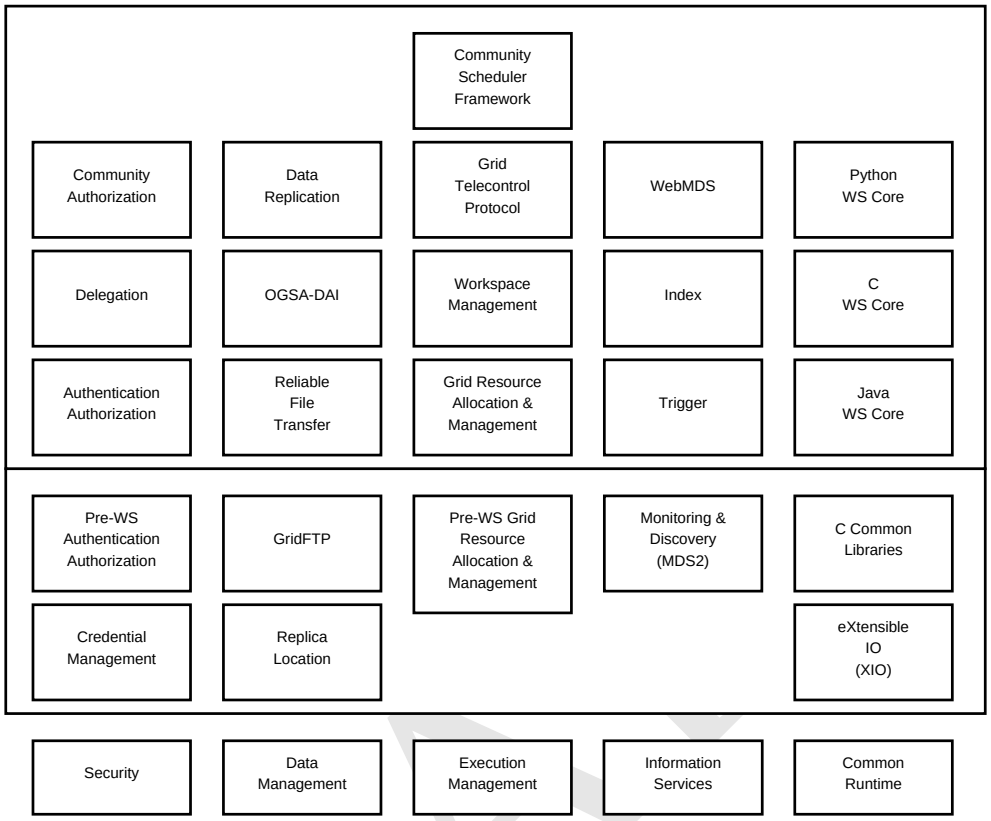


Figure 2.2: Globus Toolkit version 4 (GT4)

2.2.2 gLite

gLite is a middleware which was created to be used in the operation of the experiment LHC in CERN. The user community is grouped in Virtual Organizations, and the security model is GSI. A grid using gLite consists of User Interface, Computer Element, Storage Element, Workload Management System and Information Service.

The information service in version 3.1 of gLite is similar to MDS of Globus middleware, except that the GRIS and GIIS are provided by BDII (see Section BDII) which is an LDAP based service.

2.3 Information Services

A Grid Monitoring Architecture [5] was proposed in early 2000's. Information systems were developed to create repositories of information needed to be stored for monitoring and statistical reporting reasons. Such an organized system later was specified by the Aggregated Topology Provider (ATP) defi-

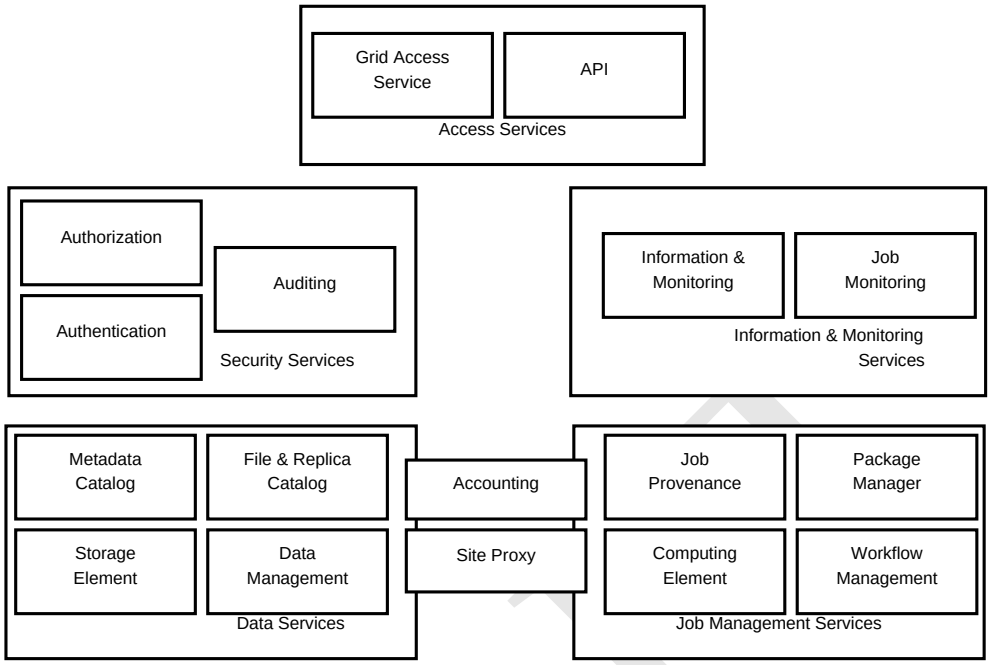


Figure 2.3: gLite architecture

nition. The largest world grids adopt that model, forming OIM in OSG (USA) and GOCDB as that information base in EGEE (Europe). Message Bus was also defined as a mean to transfer the underlying data, and many tools came up such as Gstat, GOCDB and BDII with Glue specification. Grid performance monitoring and keeping of such an information system has also impact in the performance of the system itself [6], so various methods were developed to give the solution to the scaling and performance problem, such as MDS2 (GIIS & GRIS), GMA and R-GMA [7], which offers relational environment [8], has experience on production systems [9] and scales to reach huge needs such as CMS project [10, 11].

2.3.1 MDS

Monitoring and Discovery Services is about collecting, distributing, indexing and archiving information of the status of resources, services and configurations. The collected information is used to detect new services and resources, or to monitor the state of a system.

Globus Toolkit was using LDAP-based implementation for its information system since its early versions, back in 1998 [12]. MDS2 in Globus Toolkit

fully implemented referral with a combined GRIS and GIIS, using mds-vo-name=local to refer to the GRIS and all other strings to refer to a GIIS. It was widely accepted as a standard implementation of a grid information system [13], with good scalability and performance [14].

MDS 4 consists of the Web Services Resource Framework and a web service data browser, WebMDS. The WSRF Aggregator Framework includes:

1. MDS-Index, which provides a collection of services monitoring information and an interface to query such information.

2. MDS-Trigger, which provides a mechanism to take action on collected information.

3. MDS-Archive, is planned for future release of MDS, to provide access to archived data of monitoring information.

External software components that are used to collect information (such as Ganglia)[15] are called Information Providers.

2.3.2 Glue

As long as Information Services are used to connect different infrastructures, the schema of its structure had to be standardized. To inter-operate EU and USA grids, DataTAG developed the GLUE schema implementation. GLUE specification quickly was adopted by the communities and currently its recommended LDAP DIT is specified in GLUE specification v.2.0 from GLUE Working Group of OSG.

Many objectclasses of the Glue schema define a Computer Element, a Storage Element, etc. As seen in Figure 3.3 in later chapter, performance monitoring attributes such as processor load, are defined in objectclasses that extend Computer Element objectclass.

2.3.3 BDII

BDII is used by gLite as the Information Index Service of the LHC experiment. It is LDAP based and may be at top-level or site-level. The GIIS has been replaced by site BDII, which is fundamental for a site in order to be visible in the grid.

Top-level BDII contains aggregated information about the sites and the services they provide. Site BDII collects the information from its Computer Elements, Storage Elements, etc. It also collects information for each configured service that is installed on the site.

Information about the status of a service and its parameters is pushed on BDII using external processes. An information provider is also used (such as in WSRF) to describe the service attributes using the GLUE schema.

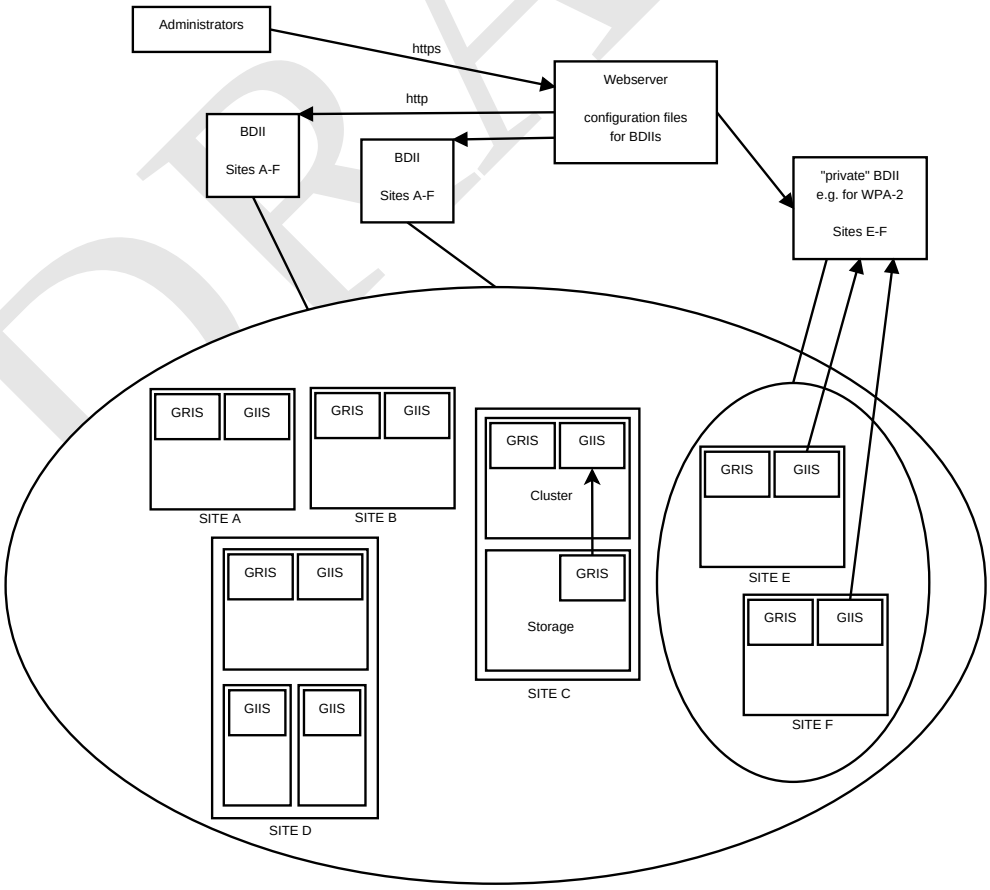


Figure 2.4: Berkeley Database Information Index

2.4 Performance Monitoring

After EGEE, the European Grid Initiative was formed to lead in the explosion of the european grid computing community to regional initiatives. Performance and availability monitoring tools and views also follow that format. The result is the phase out of SAM [16] and the adoption of Nagios as the tool for regional grid performance monitoring.

A taxonomy effort has been made [17] to present the differences of performance monitoring systems of the grid, and later a more general [18] taxonomy paper was published to give a more general view of these tools. GridICE was generally used to aggregate the performance metrics of ROCs in high level reports [19]. Later GridICE was left, as SAM did, to meet the milestone of EGI to have a regional monitoring tool (Nagios) to report the reliability of the joined sites and report the values for SLA reasons.

Grid performance can be also measured using benchmark tools in different levels of the grid architecture, using the micro-benchmarks at the Worker Node level, the Site (CE) level and the Grid VO level. Different metrics and benchmarks exist, such as the measurement of the performance of CPUs in **MIPS using EPWhetstone** and the evaluation of the performance of a CPU in **FLOP/s and MB/s using BlasBench**. GridBench [20] provides a framework to collect those metrics using its own description language, **GBDL**.

GcpSensor [21] introduce a new performance metric called WMFLOPS. It uses PAPI [22] (Performance API) to access the hardware performance counters. For data distribution it uses MDS information system which provides dynamic metrics for CPU load average for 1, 5 and 15 minutes load.

Linux kernel provides built-in functions to monitor system performance using a metric called $load$. It is a method of individual system performance reporting based on the counter of processes running or waiting in the queue of the OS scheduler. This differs from the percentage load average report.

This project focuses on mathematically compute of the performance of a grid based on the metrics that are taken at the Worker Node level.

2.4.1 Ganglia

Ganglia is a monitoring tool which provides a complete real time monitoring environment. It is used by both academia and industry community to monitor large installations of clusters and grids. Any number of host metrics may be monitored in real time using the monitoring core, a multithreaded daemon called Gmond. It runs on every host that is in scope of monitoring. Its four main responsibilities are:

1. Monitor the changes that happen in the host state
2. Multicast over the network, the changes that has been made
3. Listen to network for changes that other ganglia nodes are multicasting and
4. Answer the status of the whole cluster to specific requests, using XML.

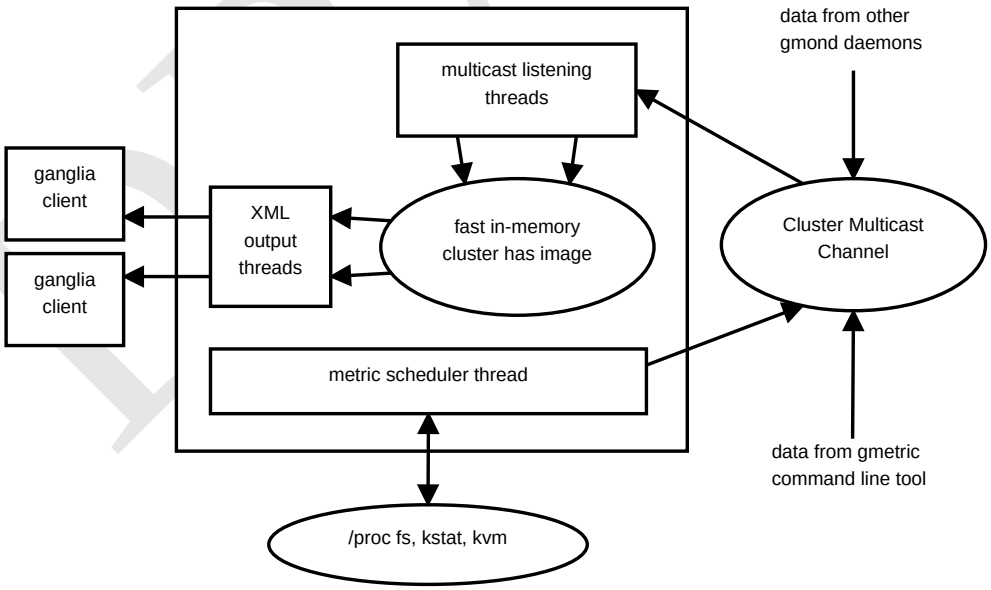


Figure 2.5: Ganglia Data Flow

All the data that are gathered from the multicast channel are written to a hash table in memory. The metric data of each node that runs gmond and sends information over the multicast channel are been processed and saved. To send data over the multicast channel, Gmond uses external data representation (XDR). When there is a request over a TCP connection, the response is in XML.

2.4.2 Nagios

Nagios is a monitoring system which provide a scheduler to check hosts and services periodically, and report their status in a CGI developed interface. Its large acceptance in industry to monitor service availability has created a large community of developers of custom check commands (plug-ins). It was also accepted from the grid community as the replacement of SAM in front of its metrics to periodically parse data from information providers about services availability.

Nagios has a strong back-end, which offers a message bus to integrate with other Nagios installations, to offer the scalability needed to connect site, regional and top level Nagios installations. Information Providers of other Information Services may be customized to be used as Nagios plugins.

Its web based front-end allows the integration with GOCDB to handle authentication, using VOMS to HTPASSWD system service. Tickets about problems or scheduled downtimes are also handled using Nagios.

Finally, its backend may be scaled-out by using NDOUtils as a service to offer database for the logging of check operations and history. PNP4Nagios is a plug-in that offers visualization of the performance metrics, using the RRDTool. Its distributed monitoring solutions recently were expanded by the Distributed Nagios eXecutor and the Multi-Nagios Tactical Overview System [23]

2.5 European Grid Infrastructure

Latest EGI directive to form regional operation tools forced the use of Nagios [24] as the main tool of availability & performance (an so reliability) monitoring of the grid. Each NGI/ROC (regional level) has its own interface, and hierarchically there is a Super Nagios interface to report the top level view of general system availability. Nagios offers extensions such as NRPE to remotely invoke check commands in inaccessible/private installations. Another

important add-on to Nagios is the NdoUtils, which offers an SQL store of history data to the monitoring interface. Nagios Configuration Generator was introduced to help the automatically generation of the configuration based on the information system of nodes and services.

Finally, there has been proposed an integration of SAM views to a Nagios customized interface, to offer the last good known SAM interface to the old users. Nagios also integrates with GGUS, a ticketing system that european grid initiative uses. Monitoring infrastructure in EGI is fully distributed using regional Nagios servers and the corresponding regional MyEGI portals.

2.5.1 NGS and GridPP

Brunel University takes part in regional and european initiatives. 5 different Computer Elements exist, and 3 Storage Elements, consisting the UKI-LT2-Brunel site. LT2 stands for London Grid, a co-operation with other London Universities. GridPP and NGS are two collaboration groups that Brunel University is member of, and papers on the web interface [25] and real time visualization of the grid status were presented [26] by GridPP

In GridPP, regional monitoring tools exist to provide distributed monitoring services in UK. Regional Nagios and MyEGI/MyEGEE instances co-exist in Oxford University that offer service availability monitoring for all UK sites. Ganglia installations exist in site level deployments, and a Ganglia frontend which aggregates Tier-1 sites is offered through RAL.

Chapter 3

Design/Methods

3.1 Approach Adopted

Grid performance monitoring in this project is examined using **GMA**, an architecture introduced to provide the standards for a distributed monitoring system. The technologies that will be discussed here are about the Information Infrastructure that provides the metrics to the users/applications.

The metrics are generated using **Linux kernel**'s load average functions. **Ganglia** is used to take these metrics and synchronize all cluster nodes with the relevant information, over the **multicast channel**.

Nagios is configured using a **custom script** that takes the information for the cluster nodes, and periodically queries the **Gmond** to get the metrics for the discovered nodes. The results are stored in its repository and using RRDTool and PNP4Nagios, graph reports are generated on demand.

To pass the information, two different information systems are examined, **BDII and WSRF**. Both are used in modern grid implementations and are described in **MDS specification**. BDII queries event source (Gmond) using Perl/Python LDAP libraries. The results taken, fill the directory schema which has been extended using **Glue schema** specification for Processor Load in Computing Element structure.

MDS4 introduces the use of **WSRF** in grid information system. A **Ganglia Information Provider** using **XSLT** takes the XML output from Gmond

and aggregates the metrics using WSRF **Aggregation Framework**. In front of it, a Tomcat instance serves the **WebMDS** frontend to allow **XPath** queries to the results that have been aggregated.

Finally, two sample small applications has been developed to provide a homogeneous interface that displays the same information using the two different information systems.

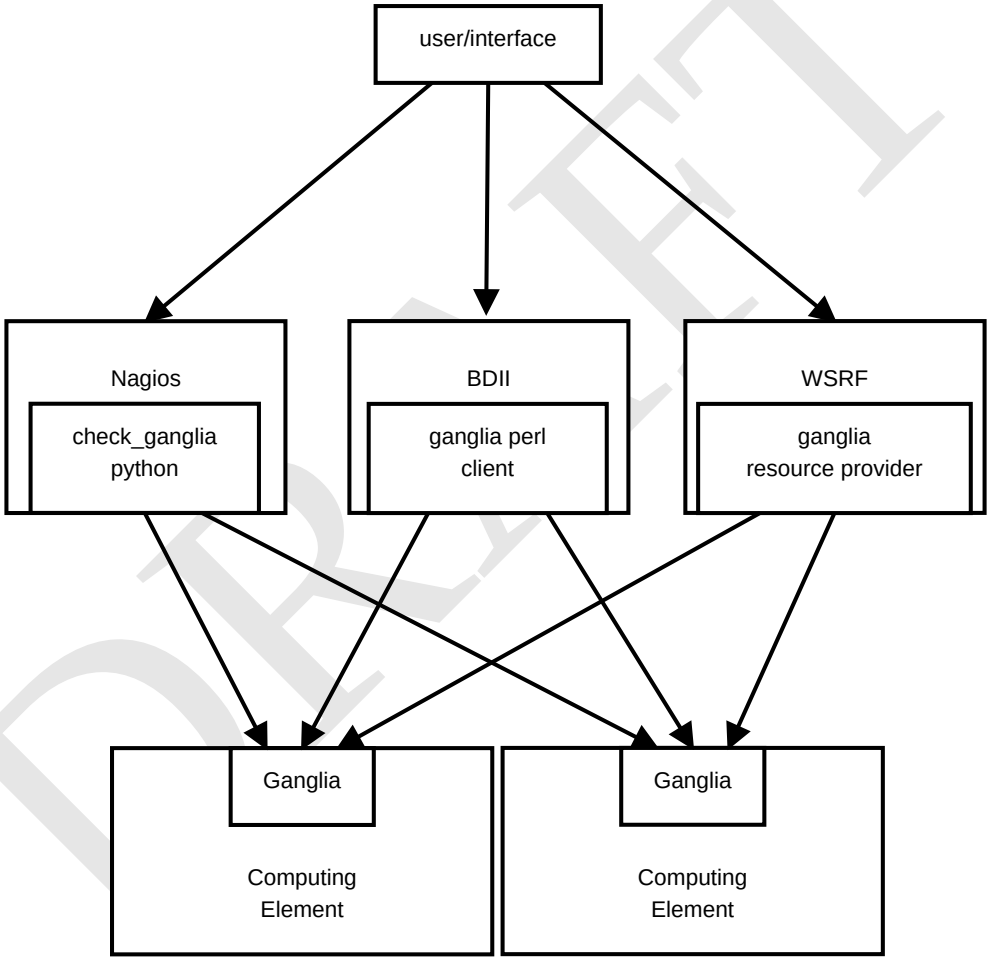


Figure 3.1: Overview of Information Systems used to monitor the grid

3.2 Design Methods

3.2.1 Grid Monitoring Architecture

By definition [3] Grid Monitoring Architecture consists of three components, as shown in Figure 3.2:

1. **Directory Service** which supports the publishing and discovery of the information

2. **Producer component**: which is responsible for the availability of the performance data that takes from the event source and

3. **Consumer component**: the one that requests the performance data and receives the metrics from the producer.

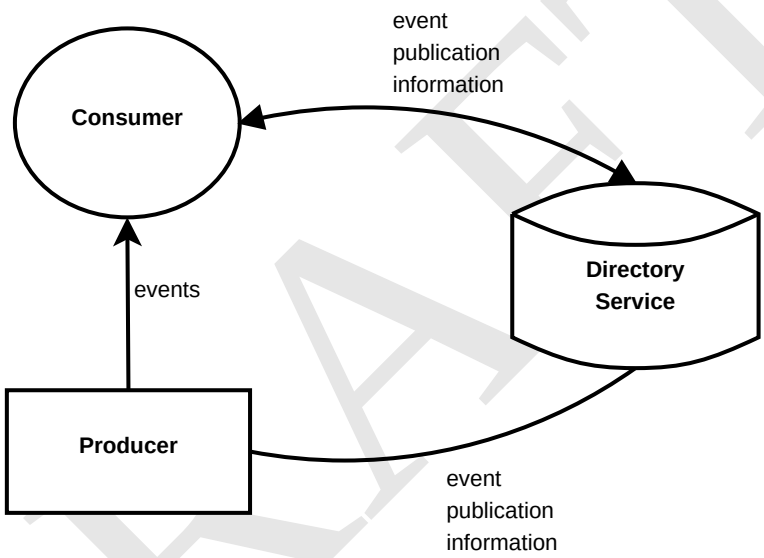


Figure 3.2: Grid Monitoring Architecture

In GMA, all metrics that are transmitted by the producer are handled as events with a timestamp, so performance data should be accurate. These events are transmitted to the consumer directly, and not through the directory service (whose role is just to advertise producers to consumers and vice versa). The GMA recommends that the structure of these data should be following a schema definition.

Grid Monitoring Architecture supports two models to handle the communication between producers and consumers:

- **Streaming publish/subscribe model**

- **Query/Response model**

The directory service is used by producers to discover consumers and by consumers to discover producers. The information of the availability of each

producer/consumer is published to the directory service. Each component may initiate a connection to another type of component which has been discovered in the directory service. Even though the role of the directory service is centralized in the discovery of components between each other, the performance data messages are transferred between the producer/consumer directly and not via the Directory Service.

3.2.2 GLUE Schema

GLUE schema came to provide the interoperability needed between US and European Physics Grid Projects. As a standard, a common schema was introduced to describe and monitor the grid resources. Major components include:

- Computing Element (CE)
- Storage Element (SE)
- Network Element (NE)

The implementation of Glue schema may be using LDAP, XML or SQL. The MDS implementation of the Glue schema in this project includes the core Information Provider and the Ganglia Interface for the cluster information.

3.2.3 Information Infrastructure

Because grid computing applications usually operate in large scale installations, there are performance requirements for the information infrastructure, such as performance, scalability, cost and uniformity. Rapid access to configuration information that is frequently used, should be enhanced using **caching to query periodically each host or index server for the metrics**.

The number of components in a grid infrastructure scales up to hundreds of thousands of nodes, and these components should be available for queries by many different tools. That information should be discoverable using information indexes.

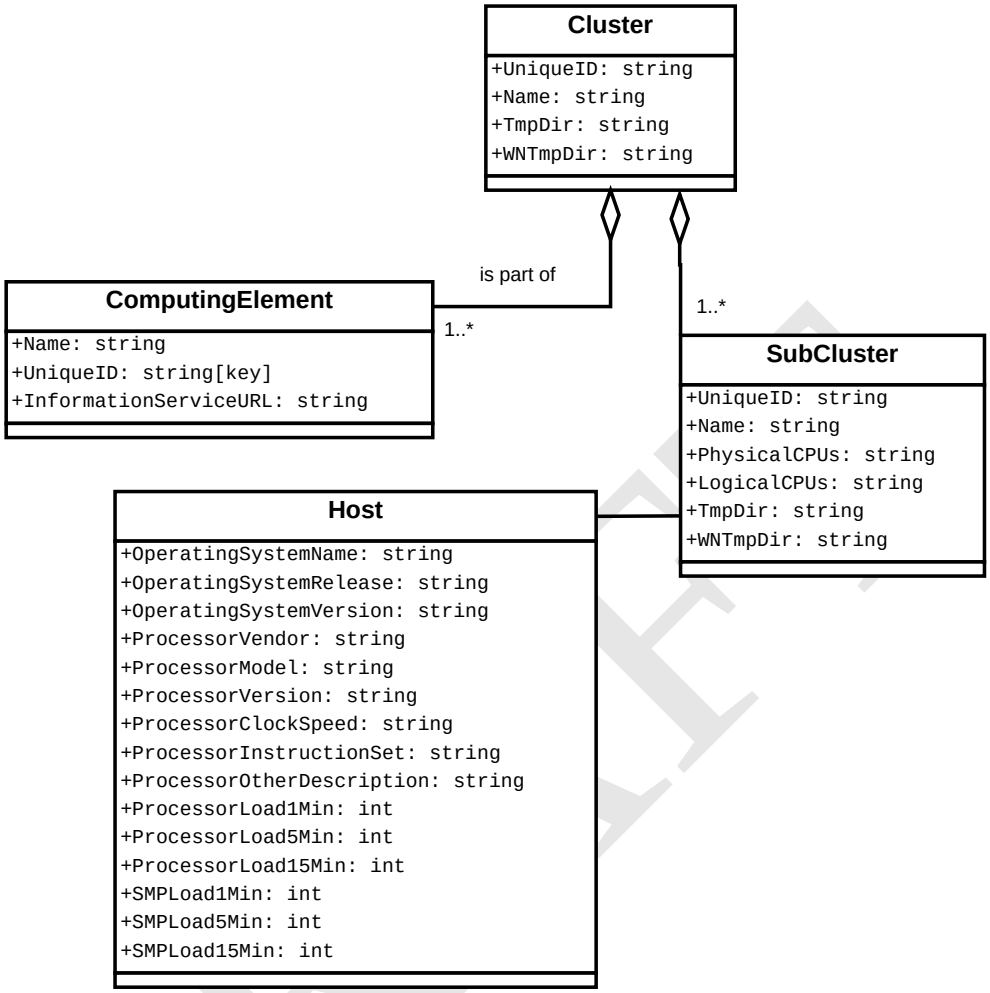


Figure 3.3: GLUE schema 2.0 extension for Host and SMP Load

Deployments, maintenance and operations in a large installation of many systems have operational costs for human resources. The information system should automatically discover and serve the availability paths for applications and grid resources/services.

Because of the large number of different heterogeneous networks of nodes and clusters, there is a need of uniformity. Uniformity helps developers to build applications that give better configuration decisions, by simplification, to build APIs for common operations and data models for the representation of that information. Resources are divided in groups of computing, storage, network elements, etc.

The solution proposed by GLUE standard and X.500 (Directory Service) is the key feature to scale, and get uniformity. It may be used to provide extensible distributed directory services. It is optimized for reads, its binary-

tree like hierarchy and usually back-end data structure provides a framework that well organizes the information that need to be delivered by an Information Infrastructure.[27]

3.3 Data-acquisition Systems

3.3.1 Metrics

CPU load is taken using the pseudo /proc/loadavg file which in turn is filled by Linux kernel's CALC_LOAD macro. This function takes 3 parameters: The load-average bucket, a y constant that is calculated using formula

$$y = \frac{2^{11}}{2^{((5log_2(e))/60x)}}$$

for values $x = 1$, $x = 5$ and $x = 15$ (where x represent the minutes and y the exponent constant), and the number of how many processes are in the queue, in running or uninterruptible state.

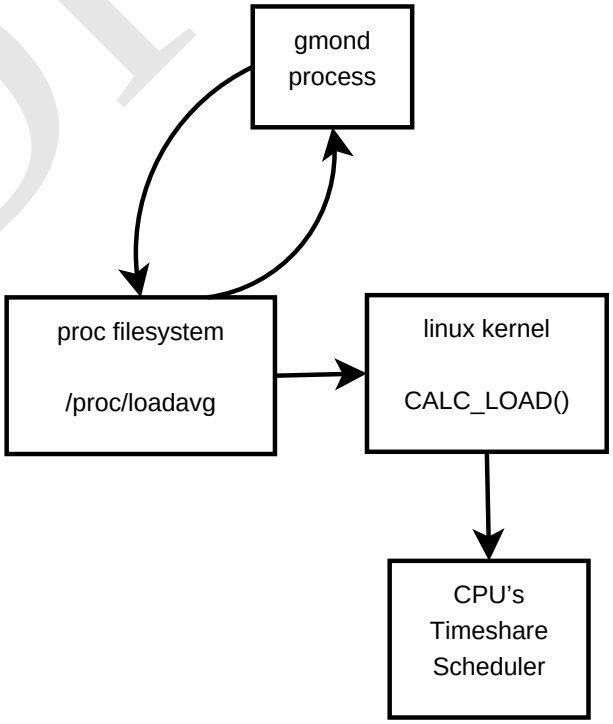


Figure 3.4: Load Average calculation

3.3.2 Ganglia

The metrics about load in one, five and fifteen minutes are taken from Gmond daemon through the proc filesystem as seen in Figure 3.4. These values are multicasted using a UDP message on the network, only if the value has been changed from the previous one taken. There is also a time threshold that after that time the value is been sent again, even if it haven't changed, so new hosts on the network may gather the data needed for their Gmond. Each host of a cluster have the information about the metrics of itself and each other node, so it stores the whole cluster state. Using loopback interface, every Gmond sends its metrics to itself.

If a TCP connection on the Gmond listening port 8649 is made, Gmond writes a full cluster state of metrics in XML including its DTD. There is a typical access list in the configuration called trusted hosts, where every node of that cluster is allowed to connect to get the XML.

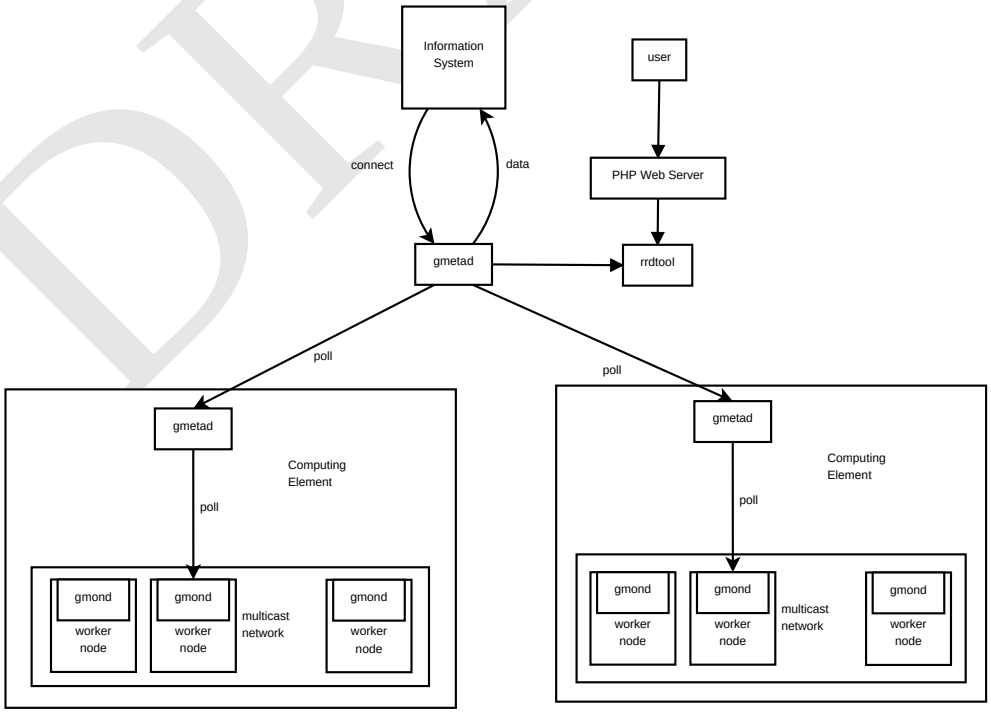


Figure 3.5: Ganglia Network Communications

Installation and configuration

In order to install ganglia, some dependencies were needed to be installed on each node of the Computing Element. In the testbed, there were an installation of LTSP [28] and the quick deployment of ganglia succeeded. Ganglia sources compiled for Gmond on the nodes and Gmetad on the systems that Ganglia Web interface needed to be installed. Finally on worker nodes, iptables should accept connections on 8649/TCP port.

```
# yum install rrdtool perl-rrdtool rrdtool-devel apr-↩
   devel libconfuse libconfuse-devel expat expat-devel↩
    pcre pcre-devel
# GANGLIA_ACK_SYSCONFDIR=1 ./configure --with-gmetad
# make
# make install
# mkdir -p /var/lib/ganglia/rrds
# chown -R nobody /var/lib/ganglia
```

```
# yum install apr-devel libconfuse libconfuse-devel ↩
   expat expat-devel pcre pcre-devel
# GANGLIA_ACK_SYSCONFDIR=1 ./configure
# make
# make install
# iptables -A RH-Firewall-1-INPUT -m state --state NEW↩
    -m tcp -p tcp --dport 8649 -j ACCEPT
```

Gmond and Gmetad default configuration may be generated using the daemon itself. Gmond may be configured using multicast to communicate metrics between nodes or unicast to solve problems with jitter when deployed in environments like amazon ec2 that do not support multicast.

3.3.3 Nagios

Nagios is the core monitoring tool that is used for grid computing monitoring as Multi Level Monitoring architecture proposes, to meet the needs of EGEE/EGI. Following SAM and Gridview, Nagios instances have been deployed in many levels of grid infrastructure, enhancing the functionality of scheduling and execution of site tests. The message bus that uses is MSG, which offers an integration between Nagios and the other monitoring tools of grid.

CERN provides MSG-Nagios-bridge, a mechanism to transfer test results between different levels of Nagios deployment (regional, project, site). MSG-Nagios-bridge submit tests to other Nagios installations and consume results from them.

A Regional Metric Store is also used by Nagios. It is a database that provides a back-end to Nagios current and historical metrics, and connects with the frontend and the message bridge. The adapter that provides such functionality called NdoUtils, and may have a MySQL/PostgreSQL or Oracle back-end.

In the front-end, users are allowed to discover the nodes and services provided in the monitoring levels by regions, projects and sites, using CGI scripts that are part of the Nagios core distribution. Access control, between levels of Nagios instances and between users and Nagios installations, is performed using the standard methods of grid, which is GOCDB as described in ATP. User authentication is done by user certificates.

To integrate Ganglia with Nagios as shown in Figure 3.6, a custom script has been created. This script queries the Gmond source for the current state of nodes of the cluster. The returned result is being transformed to a Nagios configuration file to configure the host check of the cluster nodes. The Nagios service checks for these hosts are preconfigured. Script source may be found in Listing 3.1.

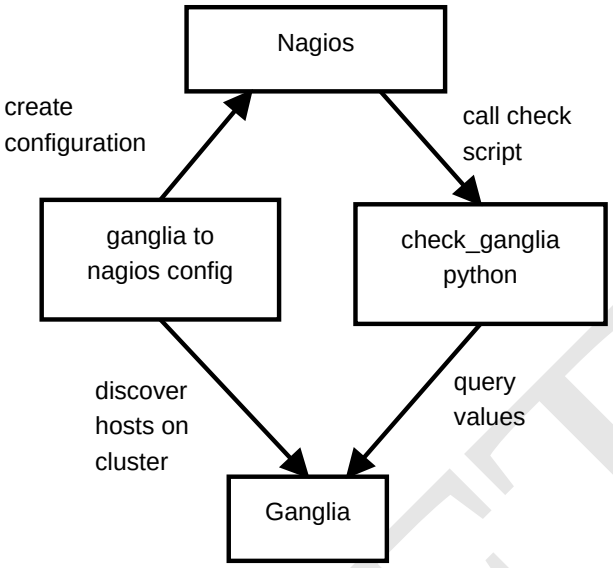


Figure 3.6: Nagios configuration and check ganglia values

Listing 3.1: Ganglia to Nagios script

```
#!/bin/bash
if [ ! $1 ]
then
        echo "Please HOST argument"
        echo "ex. ganglia_to_nagios 10.0.0.1"
        exit
fi
/usr/src/redhat/SOURCES/ganglia-python-3.3.0/ganglia.↩
   py --host $1 --live | while read host
do
        echo ";$host.oslab.teipir.gr
define host{
        use     gmond-host
        host_name       $host.oslab.teipir.gr
        alias           $host
        address         $host.oslab.teipir.gr
        hostgroups      worker-nodes
}
"
done > /etc/nagios/teipir/hosts.cfg
```

When a nagios check command is executed, results are stored in a file, and Performance Data are calculated by a perl script. To scale this process, the Bulk Mode method is used to move the file to a spool directory which takes place immediately with no important performance impact to the system, because its only an inode operation. The NPCD (Nagios Performance C Daemon) is a daemon that is running on the Nagios host and its role is to monitor a spool directory for new files and pass the names of the files to process_perfdata.pl. The script processes the performance data, and this operation is fully Nagios independent so it may be scaled-out more easily. Results are finally delivered to RRDTool, and graphs are being generated. This process is presented in Figure 3.7

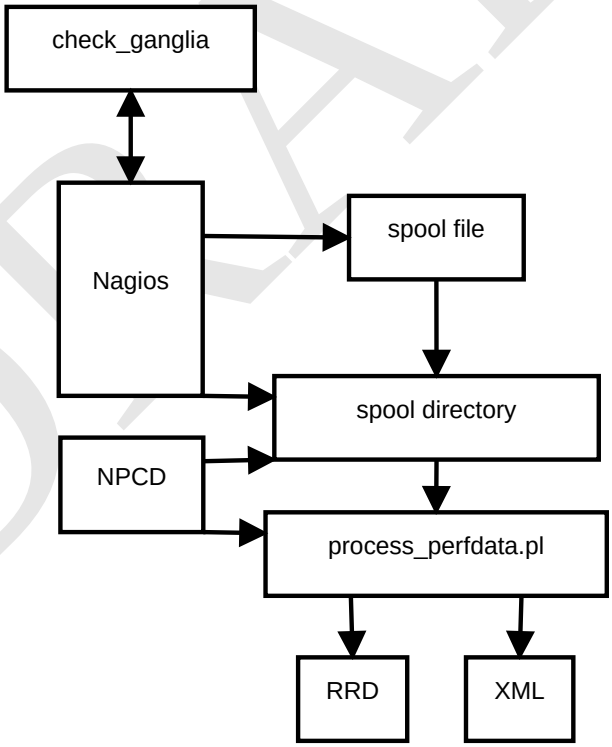


Figure 3.7: PNP 4 Nagios data flow

3.4 Range of cases examined

To deliver Ganglia metrics, two different information systems were evaluated:

- **BDII**, which is used by gLite and is based on **LDAP** and

- **WSRF**, the framework that Globus uses to aggregate and deliver information using **Web Services**.

Both Information Services are following the MDS specification and are using the Glue Schema to present the results of the metrics that are aggregated in its store.

3.4.1 LDAP based

To evaluate the LDAP based information service, a system should have the gLite installed and the BDII service running. To do this, a Scientific Linux installation were used, and CERN repositories were added. The installation of gLite-UI automatically installs BDII, and by using **yum command** the needed packages were installed. An ldapsearch returned the top elements of the BDII as shown below:

```
[root@osweb~]# ldapsearch -x -h osweb.teipir.gr -p ↩
   2170 -b "" -s base "(objectClass=*)" namingContexts
namingContexts: o=grid
namingContexts: o=glue
namingContexts: o=infosys
```

To test the connection to the Gmond service over TCP, and the transformation to MDS, two different ways were used:

1. The official ganglia python client, and
2. A perl script that is doing the same transformation

```
# /usr/bin/ganglia --host mon.oslab.teipir.gr --format↩
   =MDS
dn: mds-vo-name=local, o=grid
objectclass: GlueTop
```

```
objectclass: GlueGeneralTop
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 1
```

```
# /root/ganglia_ip -h mon.oslab.teipir.gr -p 8649 -o ↩
   mds
dn: GlueHostName=gr02.oslab.teipir.gr, mds-vo-name=↩
   local, o=grid
objectclass: GlueHost
GlueHostName: gr02.oslab.teipir.gr
GlueHostUniqueID: RDLAB-TEIPIR-gr02.oslab.teipir.gr
objectclass: GlueHostProcessorLoad
GlueHostProcessorLoadLast1Min: 1
GlueHostProcessorLoadLast5Min: 1
GlueHostProcessorLoadLast15Min: 0
```

As we can see, the LDIF exported by these tools, follows the schema defined by the Glue specification, whose attributes and objectclasses were extended by Glue-CE ProcessorLoad as shown in Table 3.1.

Common Name	**Attribute**	**Objectclass**
Hostname	GlueHostName	GlueHost
Unique ID assigned to the host	GlueHostUniqueID	GlueHost
Processor Load, 1 Min Average	GlueHostProcessorLoadLast1Min	GlueHostProcessorLoad
Processor Load, 5 Min Average	GlueHostProcessorLoadLast5Min	GlueHostProcessorLoad
Processor Load, 15 Min Average	GlueHostProcessorLoadLast15Min	GlueHostProcessorLoad
SMP Load, 1 Min Average	GlueHostSMPLoadLast1Min	GlueHostSMPLoad
SMP Load, 5 Min Average	GlueHostSMPLoadLast5Min	GlueHostSMPLoad
SMP Load, 15 Min Average	GlueHostSMPLoadLast15Min	GlueHostSMPLoad
Number of CPUs	GlueHostArchitectureSMPSize	GlueHostArchitecture
Processor Clock Speed (MHz)	GlueHostProcessorClockSpeed	GlueHostProcessor
Network Interface name	GlueHostNetworkAdapterName	GlueHostNetworkAdapter
Network Adapter IP address	GlueHostNetworkAdapterIPAddress	GlueHostNetworkAdapter
The amount of RAM	GlueHostMainMemoryRAMSize	GlueHostMainMemory
Free RAM (in KBytes)	GlueHostMainMemoryRAMAvailable	GlueHostMainMemory

Table 3.1: GLUE schema for Host Processor Information Provider

Finally, BDII was configured using **yaim** with site-info definitions as shown below:

```
# cat site-info.def
CE_HOST="osweb.teipir.gr"
SITE_BDII_HOST="osweb.teipir.gr"
SITE_EMAIL="theofpa@teipir.gr"
SITE_LAT=37.979166
SITE_LONG=23.674719
SITE_DESC="TEI of Piraeus"
SITE_LOC="Athens, Greece"
SITE_WEB="http://oslab.teipir.gr"
SITE_SECURITY_EMAIL=$SITE_EMAIL
SITE_SUPPORT_EMAIL=$SITE_EMAIL
SITE_OTHER_GRID="EGEE"
BDII_REGIONS="oslab.teipir.gr"
# /opt/glite/yaim/bin/yaim -c -s site-info.def -n ↩
   BDII_site
```

In order to integrate Ganglia with MDS in early versions of Globus and BDII of gLite, the schema of OpenLDAP should be extended using the Glue-CE definitions from the DataTAG web site (MDS version 2.4). The Ganglia Information Provider that was used is a ganglia client on perl, and not the python client given by the ganglia development team it shelf.

gLite has a dedicated directory for information providers, where the wrappers of each provider reside. An one-line wrapper to call the perl script was created, to use the information provider with BDII as shown below:

```
[root@osweb ~]# ps -ef |grep bdi[i]
edguser  24951     1  0 Feb16 ?        00:00:23 /usr/↩
   sbin/slapd -f /opt/bdii/etc/bdii-slapd.conf -h ldap↩
   ://osweb.teipir.gr:2170 -u edguser
edguser  24990     1  0 Feb16 ?        00:01:00 /usr/↩
   bin/python /opt/bdii/bin/bdii-update -c /opt/bdii/↩
```

```
   etc/bdii.conf -d
[root@osweb ~]# grep PROVIDER /opt/bdii/etc/bdii.conf
BDII_PROVIDER_DIR=/opt/glite/etc/gip/provider
[root@osweb ~]# cat /opt/glite/etc/gip/provider/glite-↩
   provider-ganglia-wrapper
#!/bin/bash
/opt/bin/ganglia_ip -h 195.251.70.54 -p 8649 -o mds
```

3.4.2 Web Service based - WSRF

Globus on the other hand, since version 4 and later provides the Web Service Resource Framework that offers a scalable information system with build-in aggregation framework and index service as shown in Figure 3.8. WSRF is an OASIS organization standard and follows the Glue schema and MDS specification.

Globus Toolkit version 4.0.7 was used to install WSRF, by extracting its binary distribution in the target system. A PostgreSQL database was installed and a special user and DB was created to host the RFT schema and data in order to have a minimal globus environment and start the container to service WSRF. A custom start/stop script was created for that container and the file rpprovider-config-gluece.xml was created as shown in Listing 3.2.

Listing 3.2: Ganglia Resource Provider for WSRF Index

```
<ns1:ResourcePropertyProviderConfigArray xsi:type="↩
   ns1:ResourcePropertyProviderConfigArray" xmlns:ns1=↩
   "http://mds.globus.org/rpprovider/2005/08" ↩
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-↩
   instance">
 <ns1:configArray xsi:type="↩
    ns1:resourcePropertyProviderConfig">
  <ns1:resourcePropertyName xsi:type="xsd:QName" ↩
```

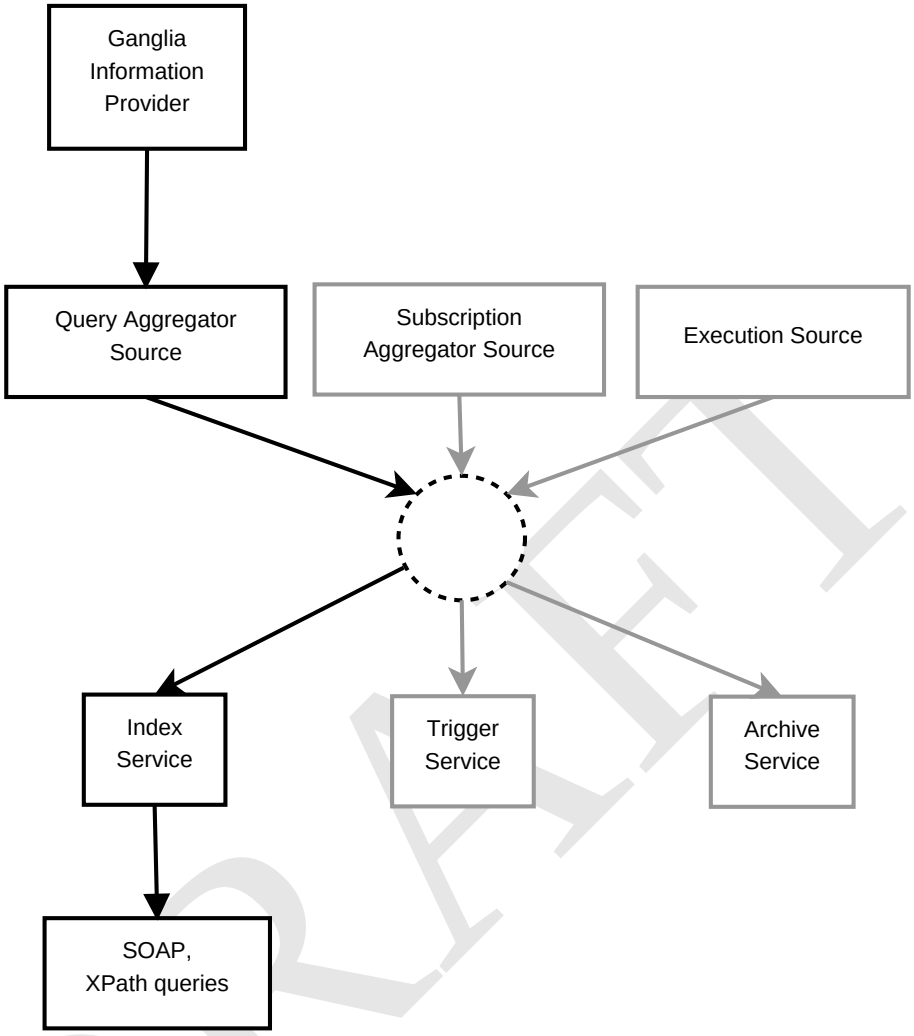


Figure 3.8: Web Service Resource Framework

```
    xmlns:mds="http://mds.globus.org/glue/ce/1.1">↩
    mds:GLUECE</ns1:resourcePropertyName>
<ns1:resourcePropertyImpl xsi:type="xsd:string">org.↩
    globus.mds.usefulrp.rpprovider.↩
    GLUEResourceProperty</ns1:resourcePropertyImpl>
<ns1:resourcePropertyElementProducers xsi:type="↩
    ns1:resourcePropertyElementProducerConfig">
  <ns1:className xsi:type="xsd:string">org.globus.↩
      mds.usefulrp.glue.GangliaElementProducer</↩
      ns1:className>
  <ns1:arguments xsi:type="xsd:string">195.251.70.55↩
      </ns1:arguments>
  <ns1:arguments xsi:type="xsd:string">8649</↩
```

```
        ns1:arguments>
    <ns1:period xsi:type="xsd:int">60</ns1:period>
    <ns1:transformClass xsi:type="xsd:string">org.↩
        globus.mds.usefulrp.rpprovider.transforms.↩
        GLUEComputeElementTransform</ns1:transformClass↩
        >
  </ns1:resourcePropertyElementProducers>
  <ns1:resourcePropertyElementProducers xsi:type="↩
     ns1:resourcePropertyElementProducerConfig">
    <ns1:className xsi:type="xsd:string">org.globus.↩
        mds.usefulrp.rpprovider.producers.↩
        SchedulerInfoElementProducer</ns1:className>
    <ns1:arguments xsi:type="xsd:string">libexec/↩
        globus-scheduler-provider-fork</ns1:arguments>
    <ns1:transformClass xsi:type="xsd:string">org.↩
        globus.mds.usefulrp.rpprovider.transforms.↩
        GLUESchedulerElementTransform</↩
        ns1:transformClass>
    <ns1:period xsi:type="xsd:int">300</ns1:period>
  </ns1:resourcePropertyElementProducers>
 </ns1:configArray>
</ns1:ResourcePropertyProviderConfigArray>
```

File rpprovider-config-gluece.xml was included by the server-config.wsdd of the container as shown in Listing 3.3.

Listing 3.3: Web Service Deployment Descriptor for WSRF Index

```
<service name="DefaultIndexService" provider="Handler"↩
    use="literal" style="document">
 <parameter name="providers"
  value="org.globus.mds.usefulrp.rpprovider.↩
     ResourcePropertyProviderCollection
```

```
  org.globus.wsrf.impl.servicegroup.↩
     ServiceGroupRegistrationProvider
  GetRPProvider
  GetMRPProvider
  QueryRPProvider
  DestroyProvider
  SetTerminationTimeProvider
  SubscribeProvider
  GetCurrentMessageProvider"/>
 <parameter name="rpProviderConfigFile" value="/etc/↩
    globus_wsrf_mds_index/rpprovider-config-gluece.xml↩
    "/>
 <parameter name="handlerClass" value="org.globus.axis↩
    .providers.RPCProvider"/>
 <parameter name="scope" value="Application"/>
 <parameter name="allowedMethods" value="*"/>
 <parameter name="className"
  value="org.globus.mds.index.impl.DefaultIndexService↩
     "/>
 <wsdlFile>share/schema/mds/index/index_service.wsdl</↩
    wsdlFile>
</service>
```

When the container started, a user proxy certificate was initialised and an XPath query was issued to test the integration:

Listing 3.4: WSRF command line query

```
[root@osweb ~]# /opt/globus/bin/grid-proxy-init
Your identity: /C=GR/O=HellasGrid/OU=teipir.gr/CN=↩
   Theofylaktos Papapanagiotou
Enter GRID pass phrase for this identity:
Creating proxy ..................................... ↩
```

```
   Done
Your proxy is valid until: Thu Feb 17 11:11:59 2011
[root@osweb ~]# /opt/globus/bin/wsrf-query -s https://↩
   osweb.teipir.gr:8443/wsrf/services/↩
   DefaultIndexService "//*[local-name()='Host']"
<ns1:Host ns1:Name="gr02.oslab.teipir.gr" ns1:UniqueID↩
   ="gr02.oslab.teipir.gr" xmlns:ns1="http://mds.↩
   globus.org/glue/ce/1.1">
 <ns1:Processor ns1:CacheL1="0" ns1:CacheL1D="0" ns1:↩
    CacheL1I="0" ns1:CacheL2="0" ns1:ClockSpeed="2527"↩
     ns1:InstructionSet="x86"/>
 <ns1:MainMemory ns1:RAMAvailable="75" ns1:RAMSize="↩
    495" ns1:VirtualAvailable="1129" ns1:VirtualSize="↩
    1559"/>
 <ns1:OperatingSystem ns1:Name="Linux" ns1:Release="↩
    2.6.18-194.26.1.el5"/>
 <ns1:Architecture ns1:SMPSize="1"/>
 <ns1:FileSystem ns1:AvailableSpace="34082" ns1:Name="↩
    entire-system" ns1:ReadOnly="false" ns1:Root="/" ↩
    ns1:Size="38624"/>
 <ns1:NetworkAdapter ns1:IPAddress="10.0.0.32" ns1:↩
    InboundIP="true" ns1:MTU="0" ns1:Name="gr02.oslab.↩
    teipir.gr" ns1:OutboundIP="true"/>
 <ns1:ProcessorLoad ns1:Last15Min="0" ns1:Last1Min="0"↩
     ns1:Last5Min="0"/>
```

XPath

XPath is used to parse an XML document and get a part of it using an address scheme. XPath considers XML document as a tree, consisting of nodes. Its purpose as a language is to get from that document, the nodes that are addressed using the XPath query.

Its syntax is compact, non-XML and much like the filesystem addressing, so it facilitates the use of XPath within URIs.

Example queries used in this project are:

The following is used in the PHP code that queries the WebMDS for all nodes of the XML of the WSRF containing nodes with name $Host$:

```
//*[local-name()='Host']
```

Another example is a more complex query that asks the WSRF for all nodes by the name $Host$ that contains a sub-node named $ProcessorLoad$ and its $Last15Min$ attribute has value larger than 20:

```
//glue:Host[glue:ProcessorLoad[@glue:Last15Min>20]]
```

Finally the following example may return only the $ProcessorLoad$ node of the $Host$ that has the attribute Name set to $xenia.oslab.teipir.gr$:

```
//glue:Host[@glue:Name='xenia.oslab.teipir.gr']/glue:↩
  ProcessorLoad
```

WebMDS

WebMDS is a web interface to query WSRF resource property information. It consists of forms and views of raw XML or organized in tables of results. This user friendly frontend comes as a part of Globus Toolkit version 4 and it can be deployed in any application server. Behind this application reside the data that the WSRF aggregation framework provides through the Index Service.

Figure 3.9 display the data flow of the WSRF Information System case. The PHP from Brunel's web server calls WebMDS and get the result in XML which parses using DOM. WebMDS is deployed in the Tomcat container, and

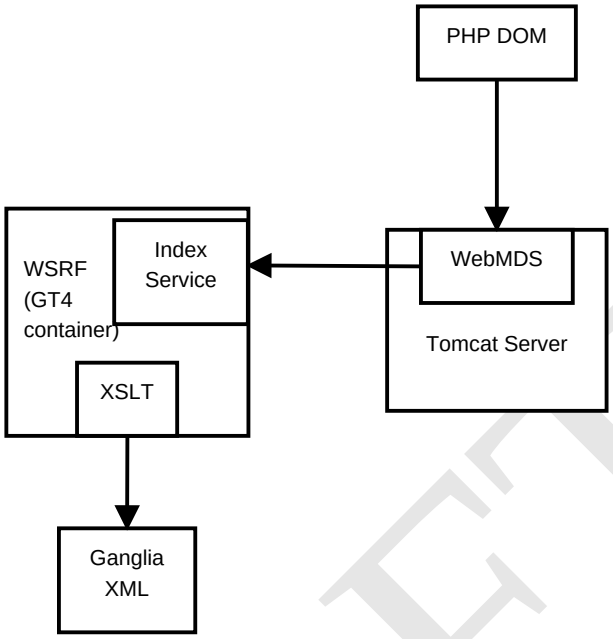


Figure 3.9: WebMDS application

calls the Index Service of WSRF which is deployed in the GT4 container. WSRF connects (if cache has expired) to the Gmond process and transforms the data received using XSLT.

For this project an Apache Tomcat server was installed in the box that globus toolkit was running, and the **webmds application** from the GT4 home was deployed. In webmds configuration file, the global option to allow user specified queries using XPath was enabled.

```
[root@osweb ~]# cat /opt/globus/lib/webmds/↩
   globalconfig.xml
<WebmdsGlobalConfig>
  <newStyleErrors>true</newStyleErrors>
  <allowUserSpecifiedQuery>true</↩
     allowUserSpecifiedQuery>
</WebmdsGlobalConfig>
```

Chapter 4

Results

4.1 Events source

Results are examined on the generation of metrics, during the aggregation using various information services and they are presented using ready and custom developed interfaces.

4.1.1 Unix stuff

As described in subsection Metrics of the previous chapter, Linux provides through the **proc pseudo-filesystem** a simple file interface to the metrics taken from the scheduler of processes that are queued in the processor.

The three metrics about CPU load on 1, 5 and 15 minutes average are displayed as follow:

```
[root@gr03 ~]# cat /proc/loadavg
2.29 0.73 0.32 1/230 3584
```

which may also be displayed using the **uptime command**:

```
[root@gr03 ~]# uptime
 00:01:20 up  1:41,  3 users, load average: 2.29, ↩
    0.73, 0.32
```

When we examine the Linux kernel source code, there is a macro command named $CALC_LOAD$ which takes the options that have been discussed and returns the result of the metric. The definition of the macro can be seen in file **include/linux/sched.h**, Listing 4.1.

Listing 4.1: Linux kernel CALC_LOAD macro

```
extern unsigned long avenrun[];         /* Load ↩
    averages */
extern void get_avenrun(unsigned long *loads,
                        unsigned long offset, int ↩
                            shift);

#define FSHIFT     11           /* nr of bits of ↩
    precision */
#define FIXED_1    (1<<FSHIFT) /* 1.0 as fixed-point */
#define LOAD_FREQ  (5*HZ+1)     /* 5 sec intervals */
#define EXP_1      1884         /* 1/exp(5sec/1min) as ↩
    fixed-point */
#define EXP_5      2014         /* 1/exp(5sec/5min) */
#define EXP_15     2037         /* 1/exp(5sec/15min) */

#define CALC_LOAD(load,exp,n) \
        load *= exp; \
        load += n*(FIXED_1-exp); \
        load >>= FSHIFT;

extern unsigned long total_forks;
extern int nr_threads;
DECLARE_PER_CPU(unsigned long, process_counts);
extern int nr_processes(void);
extern unsigned long nr_running(void);
```

```
extern unsigned long nr_uninterruptible(void);
extern unsigned long nr_iowait(void);
extern unsigned long nr_iowait_cpu(int cpu);
extern unsigned long this_cpu_load(void);
```

4.1.2 Ganglia

When gmond starts, it listens on port 8649/TCP by default, to accept TCP connections and throw XML report for the whole cluster. It also binds to the multicast address on port 8649/UDP to get other hosts messages for metrics changes, and also multicast its own metrics. Listing 4.2 shows the opened sockets of Gmond daemon, and Listing 4.3 display a sample xml output when connecting to 8649/TCP to transfer metrics through XML.

Listing 4.2: Gmond networking

```
[root@gr01 ~]# lsof -i 4 -a -p `pidof gmond`
COMMAND   PID   USER   FD   TYPE DEVICE SIZE NODE NAME
gmond   11900 nobody    4u  IPv4  33699       UDP ↩
   239.2.11.71:8649
gmond   11900 nobody    5u  IPv4  33701       TCP ↩
   *:8649 (LISTEN)
gmond   11900 nobody    6u  IPv4  33703       UDP gr01↩
   .oslab.teipir.gr:39991->239.2.11.71:8649
```

Listing 4.3: Gmond XML cluster report

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="↩
   yes"?>
<GANGLIA_XML VERSION="3.1.7" SOURCE="gmond">
  <CLUSTER NAME="RDLAB" LOCALTIME="1297198943" OWNER="↩
     TEIPIR" LATLONG="unspecified" URL="unspecified">
```

```
    <HOST NAME="gr02.oslab.teipir.gr" IP="10.0.0.32" ↩
      REPORTED="1297198934" TN="8" TMAX="20" DMAX="0"↩
       LOCATION="unspecified" GMOND_STARTED="↩
      1296569542">
      <METRIC NAME="load_one" VAL="0.01" TYPE="float" ↩
        UNITS=" " TN="50" TMAX="70" DMAX="0" SLOPE="↩
        both">
      <EXTRA_DATA>
      <EXTRA_ELEMENT NAME="GROUP" VAL="load"/>
      <EXTRA_ELEMENT NAME="DESC" VAL="One minute load ↩
        average"/>
      <EXTRA_ELEMENT NAME="TITLE" VAL="One Minute Load↩
         Average"/>
      </EXTRA_DATA>
      </METRIC>
    </HOST>
  </CLUSTER>
</GANGLIA_XML>
```

Worker nodes are configured to transfer metrics data using multicast. Each Gmond daemon of each Computing Element node, by Ganglia definition has to know the state of the whole Computing Element cluster. Using standard UNIX commands to listen to the data transferred on the multicast network, a sample transfer of load_one metric is observed. As described in Subsection 3.3.2, metric data are multicasted by Gmond when there is a change in the value, or when the time threshold is reached.

Listing 4.4: XDR sample

```
[root@gr01 ~]# tcpdump -A -i eth2 dst host 239.2.11.71
22:38:26.062266 IP gr01.oslab.teipir.gr.39991 > ↩
   239.2.11.71.8649: UDP, length 56
```

```
E..T..@.........G.7!..@.X........gr01.oslab.teipir.gr↩
    ....load_one........%.2f..
```

Using Ganglia build-in command $gstat$, a nice output of Processor Load metrics is shown for the whole cluster in Listing 4.5

Listing 4.5: Gstat output

```
[root@gr01 ~]# gstat -al1
gr03.oslab.teipir.gr     2 (    0/   87) [  0.00,  ↩
   0.00,  0.00] [   0.0,   0.0,   0.0,  99.9,   0.1] ↩
   OFF
gr01.oslab.teipir.gr     1 (    0/   75) [  0.00,  ↩
   0.00,  0.00] [   0.0,   0.0,   0.0,  99.9,   0.0] ↩
   OFF
gr02.oslab.teipir.gr     1 (    0/   99) [  0.00,  ↩
   0.00,  0.00] [   0.0,   0.0,   0.1,  99.9,   0.0] ↩
   OFF
```

4.2 Aggregation and transfer

Metrics taken from the event source are passed to the information service using the information/resource provider.

4.2.1 WSRF

In WSRF the $wsrf - query$ command is executed using the URL of the container where the WSRF was been deployed. The container should run in a host with a host certificate signed by a certificate authority, and a certificate proxy should be initialized with a valid user certificate that is requesting the information.

```
/opt/globus/bin/wsrf-query -s https://osweb.teipir.gr:8443/wsrf/services/DefaultIndexService "//*[local-name()='Host']"
```

The result of the above query returns a list of all cluster nodes. An abstract result is displayed here in Listing 4.6.

Listing 4.6: WSRF query output

```
<ns1:GLUECE xmlns:ns1="http://mds.globus.org/glue/ce/1.1">
 <ns1:Cluster ns1:Name="OSLAB" ns1:UniqueID="OSLAB">
  <ns1:SubCluster ns1:Name="main" ns1:UniqueID="main">
   <ns1:Host ns1:Name="gr03.oslab.teipir.gr"
   ns1:UniqueID="gr03.oslab.teipir.gr"
   xmlns:ns1="http://mds.globus.org/glue/ce/1.1">
    <ns1:Processor ns1:CacheL1="0" ns1:CacheL1D="0"
    ns1:CacheL1I="0" ns1:CacheL2="0" ns1:ClockSpeed="2392"
    ns1:InstructionSet="x86"/>
    <ns1:MainMemory ns1:RAMAvailable="299"
     ns1:RAMSize="1010"
    ns1:VirtualAvailable="2403" ns1:VirtualSize="3132"
     "/>
    <ns1:OperatingSystem ns1:Name="Linux"
    ns1:Release="2.6.18-194.26.1.el5"/>
    <ns1:Architecture ns1:SMPSize="2"/>
    <ns1:FileSystem ns1:AvailableSpace="201850"
    ns1:Name="entire-system" ns1:ReadOnly="false"
    ns1:Root="/" ns1:Size="214584"/>
    <ns1:NetworkAdapter ns1:IPAddress="10.0.0.33"
    ns1:InboundIP="true" ns1:MTU="0"
```

```
    ns1:Name="gr03.oslab.teipir.gr" ns1:OutboundIP="↩
       true"/>
    <ns1:ProcessorLoad ns1:Last15Min="45" ↩
       ns1:Last1Min="337"
    ns1:Last5Min="126"/>
   </ns1:Host>
  </ns1:SubCluster>
 </ns1:Cluster>
</ns1:GLUECE>
```

WebMDS and XPath

The role-based access control model of the grid security context allow queries only by authenticated and authorized users. Building a testbed with full grid security in mind would be out of this project scope, so XML result of Web Service calls is taken through WebMDS instead of WSDL discovery and SOAP messaging.

Using the following XPath query in the WebMDS form, a request to get the metrics of Host node with name $ltsp.oslab.teipir.gr$ was sent.

```
//glue:Host[@glue:Name='ltsp.oslab.teipir.gr']
```

WebMDS match the query in its cache and replies only the specific node that was requested. If the cache was expired (its default value is 60 seconds) the information system uses the Resource Provider to fetch the XML from Gmond and transform it using XSLT to Glue schema and serve the new values as shown in Listing 4.7.

Listing 4.7: WebMDS results from XPath query

```
<WebmdsResults>
 <ns1:Host ns1:Name="ltsp.oslab.teipir.gr"
 ns1:UniqueID="ltsp.oslab.teipir.gr">
```

```
    <ns1:Processor ns1:CacheL1="0" ns1:CacheL1D="0"
    ns1:CacheL1I="0" ns1:CacheL2="0" ns1:ClockSpeed="↩
       1600"
    ns1:InstructionSet="x86_64"/>
    <ns1:MainMemory ns1:RAMAvailable="17806" ↩
       ns1:RAMSize="20121"
    ns1:VirtualAvailable="22137" ns1:VirtualSize="24508↩
       "/>
    <ns1:OperatingSystem ns1:Name="Linux"
    ns1:Release="2.6.32-24-server"/>
    <ns1:Architecture ns1:SMPSize="8"/>
    <ns1:FileSystem ns1:AvailableSpace="34243"
    ns1:Name="entire-system" ns1:ReadOnly="false" ↩
       ns1:Root="/"
    ns1:Size="251687"/>
    <ns1:NetworkAdapter ns1:IPAddress="192.168.0.101"
    ns1:InboundIP="true" ns1:MTU="0" ns1:Name="ltsp.↩
       oslab.teipir.gr"
    ns1:OutboundIP="true"/>
    <ns1:ProcessorLoad ns1:Last15Min="9" ns1:Last1Min="↩
       1"
    ns1:Last5Min="9"/>
  </ns1:Host>
</WebmdsResults>
```

4.2.2 BDII

On the other information service, using *ldapsearch* command and specifying the base DN for the search, the host URI and the desired attributes to return, we may get the values asked from the BDII as seen on Listing 4.8.

Listing 4.8: BDII LDAP search for Glue CE ProcessorLoad attributes

```
# ldapsearch -H ldap://osweb.teipir.gr:2170 -x \
-b GlueHostName=ainex.local,Mds-Vo-name=local,o=grid \
GlueHostProcessorLoadLast1Min ↩
   GlueHostProcessorLoadLast5Min \
GlueHostProcessorLoadLast15Min

# ainex.local, local, grid
dn: GlueHostName=ainex.local,Mds-Vo-name=local,o=grid
GlueHostProcessorLoadLast1Min: 27
GlueHostProcessorLoadLast15Min: 22
GlueHostProcessorLoadLast5Min: 20
```

The above information has been given by the BDII LDAP instance which used the wrapper of Ganglia Resource Provider, a customized Perl script to export MDS format as shown in Listing 4.9.

Listing 4.9: Perl Ganglia Information Provider for MDS

```
[root@mon ~]# ./ganglia_ip -h mon -p 8649 -o mds | ↩
   grep -A 22 host=gr03

dn: host=gr03.oslab.teipir.gr, cl=RDLAB, \
 mds-vo-name=local, o=grid
objectclass: GlueHost
GlueHostName: gr03.oslab.teipir.gr
GlueHostUniqueID: RDLAB-TEIPIR-gr03.oslab.teipir.gr
objectclass: GlueHostProcessorLoad
GlueHostProcessorLoadLast1Min: 2.57
GlueHostProcessorLoadLast5Min: 1.48
GlueHostProcessorLoadLast15Min: 0.58
objectclass: GlueHostSMPLoad
GlueHostSMPLoadLast1Min: 2.57
```

```
GlueHostSMPLoadLast5Min: 1.48
GlueHostSMPLoadLast15Min: 0.58
objectclass: GlueHostArchitecture
GlueHostArchitectureSMPSize: 2
objectclass: GlueHostProcessor
GlueHostProcessorClockSpeed: 2392
objectclass: GlueHostNetworkAdapter
GlueHostNetworkAdapterName: gr03.oslab.teipir.gr
GlueHostNetworkAdapterIPAddress: 10.0.0.33
objectclass: GlueHostMainMemory
GlueHostMainMemoryRAMSize: 1035104
GlueHostMainMemoryRAMAvailable: 306280
```

Using the Ganglia official python client (Listing 4.10) which is distributed with the source code of Ganglia Client, rejected as an option because Perl is easier in handling regular expressions for string operations and transformation.

Listing 4.10: Python Ganglia client MDS export

```
[root@mon ~]# /opt/ganglia/bin/ganglia --format=MDS | ↩
   grep -A 30 host=gr03


dn: host=gr03.oslab.teipir.gr, scl=sub2, cl=datatag-↩
   CNAF, \
 mds-vo-name=local, o=grid
objectclass: GlueHost
GlueHostName: gr03.oslab.teipir.gr
GlueHostUniqueID: RDLAB-TEIPIR-gr03.oslab.teipir.gr
objectclass: GlueHostArchitecture
GlueHostArchitecturePlatformType: x86-Linux
GlueHostArchitectureSMPSize: 2
objectclass: GlueHostProcessor
```

```
GlueHostProcessorClockSpeed: 2392
objectclass: GlueHostMainMemory
GlueHostMainMemoryRAMSize: 1035104
GlueHostMainMemoryRAMAvailable: 306280
objectclass: GlueHostNetworkAdapter
GlueHostNetworkAdapterName: gr03.oslab.teipir.gr
GlueHostNetworkAdapterIPAddress: 10.0.0.33
GlueHostNetworkAdapterMTU: unknown
GlueHostNetworkAdapterOutboundIP: 1
GlueHostNetworkAdapterInboundIP: 1
objectclass: GlueHostProcessorLoad
GlueHostProcessorLoadLast1Min: 2.57
GlueHostProcessorLoadLast5Min: 1.48
GlueHostProcessorLoadLast15Min: 0.58
objectclass: GlueHostSMPLoad
GlueHostSMPLoadLast1Min: 2.57
GlueHostSMPLoadLast5Min: 1.48
GlueHostSMPLoadLast15Min: 0.58
objectclass: GlueHostStorageDevice
GlueHostStorageDeviceSize: 209555000
GlueHostStorageDeviceAvailableSpace: 197120000
GlueHostStorageDeviceType: disk
```

4.3 Presentation

Finally, to present the information, two custom interfaces have been developed in PHP. Both programs reside in Brunel University webserver which supports the needed libraries and network connections. The source code is available under that page, and when called using BDII or WSRF, the result is exactly the same and looks like Table 4.1.

Hostname	1min	5min	15min
ltsp.oslab.teipir.gr	0.01	0.09	0.09
xenia.oslab.teipir.gr	0.00	0.06	0.06
gr201.oslab.teipir.gr	0.52	0.59	0.42
gr130.oslab.teipir.gr	0.00	0.06	0.16
gr131.oslab.teipir.gr	1.22	0.79	0.40
gr212.oslab.teipir.gr	0.06	0.09	0.09
gr180.oslab.teipir.gr	2.06	1.49	0.59
gr181.oslab.teipir.gr	0.71	0.57	0.32

Table 4.1: Sample output from both calls with DOM or LDAP

4.3.1 DOM

The first interface uses PHP DOM functions to call the WebMDS and get the XML as it would happened by calling the WSRF Web Service with native SOAP messages and authentication. Listing 4.11 is an abstraction of the code deployed in Brunel's webserver and returns only one value. The purpose of this listing is to present the use of functions and not the HTML stuff.

Listing 4.11: PHP DOM call to WebMDS

```
$url="http://osweb.teipir.gr:8080/webmds/webmds?info=↩
   indexinfo&xsl=&xmlSource.indexinfo.param.xpathQuery↩
   =%2F%2F*[local-name%28%29%3D%27Host%27]";
$file  = file_get_contents($url);
$dom = DOMDocument::loadXML($file);
$host = $dom->getElementsByTagName('Host');
$procload = $host->item($k)->getElementsByTagName('↩
   ProcessorLoad');
echo ($procload->item($i)->getAttribute('Last1Min'))↩
   /100;
```

4.3.2 LDAP

The second interface uses LDAP functions to connect to BDII instance and get the results from objects that instantiates the GlueHostProcessorLoad class.

Listing 4.12 displays the method to bind anonymously, form the query and get a sample result.

Listing 4.12: PHP LDAP call to BDII

```
$ds=ldap_connect("osweb.teipir.gr","2170");
if ($ds)
{
    $r=ldap_bind($ds);
    $sr=ldap_search($ds, "mds-vo-name=local,o=grid", "↩
       (&(objectClass=GlueHostProcessorLoad))");
    if ($sr)
    {
         $info = ldap_get_entries($ds, $sr) or die("↩
            could not fetch entries");
         echo ($info[0][gluehostprocessorloadlast1min↩
            ][0])/100;
    }
ldap_close($ds);
}
```

4.3.3 Nagios

Nagios calls check_ganglia periodically (configured as check interval) and logs the state of each service and host check. Load averages are described as services. Listing 4.13 shows the log file of nagios service check for these values straight from Ganglia.

Listing 4.13: Nagios log with load check

```
[1297634400] CURRENT SERVICE STATE: xenia.oslab.teipir↩
   .gr;load_fifteen;OK;HARD;1;CHECKGANGLIA OK: ↩
   load_fifteen is 0.00
```

```
[1297634400] CURRENT SERVICE STATE: xenia.oslab.teipir↩
   .gr;load_five;OK;HARD;1;CHECKGANGLIA OK: load_five ↩
   is 0.00
[1297634400] CURRENT SERVICE STATE: xenia.oslab.teipir↩
   .gr;load_one;OK;HARD;1;CHECKGANGLIA OK: load_one is↩
    0.00
```

NPCD is configured in bulk mode so a file may be found in the spool directory before the interval passes to process it with perl script to RRD database. Listing 4.14 contains the performance data metrics.

Listing 4.14: NPCD temporary file in spool directory

```
[root@osweb ~]# cat /var/spool/pnp4nagios/host-↩
   perfdata.1297973378
DATATYPE::HOSTPERFDATA  TIMET::1297973368 HOSTNAME::↩
   osweb.teipir.gr HOSTPERFDATA::rta=0.057000ms↩
   ;3000.000000;5000.000000;0.000000 pl=0%;80;100;0  ↩
   HOSTCHECKCOMMAND::ncg_check_host_alive  HOSTSTATE::↩
   UP HOSTSTATETYPE::HARD
```

Finally, Nagios Web interface displays the aggregated values of host performance state in multiple views, as shown in Table 4.2.

Host	Service	Status	Last Check	Status Information
gr129	load_fifteen	OK	02-08-2011 20:17:23	CHECKGANGLIA OK: load_fifteen is 0.17
	load_five	OK	02-08-2011 20:18:12	CHECKGANGLIA OK: load_five is 0.27
	load_one	OK	02-08-2011 20:17:43	CHECKGANGLIA OK: load_one is 0.02
gr130	load_fifteen	OK	02-08-2011 20:14:23	CHECKGANGLIA OK: load_fifteen is 1.77
	load_five	WARNING	02-08-2011 20:14:15	CHECKGANGLIA OK: load_five is 4.75
	load_one	CRITICAL	02-08-2011 20:14:43	CHECKGANGLIA OK: load_one is 11.60

Table 4.2: Example Nagios service status details for ganglia check

Chapter 5

Analysis

5.1 Methods Adopted

In complex distributed systems such as grids, performance bottlenecks may be located using monitoring data. From the processor usage on a single node of a computing element to the total usage of processed jobs in a large cluster, performance data help to focus on the problem that impacts the overall performance.

In order to succeed in grid monitoring, some requirements should be considered. A very large amount of data should be delivered real-time, from many heterogeneous sources on different networks or even countries. These data must be accurate and consistent. There should be synchronized timestamps on the generation of each metric, to the measurement value that should be comparable between different architectures. Hosts time synchronization is achieved with network time protocol, so all metrics are taken on the time that they actually report. Metrics should have error bounds to preserve accuracy, and the consistency issue is solved using coordination of that activity, so the impact of a metric to other sensors is controlled.

The flow of the monitoring process initialization is described from the GMA standard. The application-consumer queries the directory service in order to declare its interest to get metrics for a specific host/cluster. The sensors of the elements that is equivalent to the specific query generates the metrics

that will be given to the consumer from the producer, which in turn queries the directory service to find the consumer. The producer is the one that initializes the connection to the consumer in order to deliver the measurements, even if the consumer had asked the directory service for this. [29]

5.1.1 Performance Metrics

Linux kernel process scheduler provides the three numbers for the processor load during the last one, five and fifteen minutes period. There are many ways to calculate the system load in UNIX systems, updating the values on each scheduler event or periodically sampling the scheduler state.

Scheduler efficiency impact on total system's performance, and periodic sampling lead to more accurate load averages. Linux uses the second one, with the period of five seconds, five clock ticks. Each "clock tick" is represented by the HZ variable which is the pulse rate of a kernel activity. This method is more performance-wise from the first one where recalculation of load occurs in every scheduler event.

There is a significant difference between processor load and processor usage of a system. Processor usage is a percent representation of the use of CPU by processes, and not used in this project. CPU load is a better metric for the system performance because it does not have a top value of 100% but is based in processes running and waiting in the scheduler queue.

Transport and sample

Gmond code uses the ganglia libmetrics library which in case of Linux operating system parses the $/proc/loadavg$ pseudo-file to get Linux kernel calculated system load average as shown in Listing 5.1. When this value is taken, it is pushed to the multicast network using UDP datagram. If this value has not changed from the previous one, it is not pushed until a timeout is reached.

Listing 5.1: libmetrics code to get load average

```
timely_file proc_loadavg = { {0,0} , 5., "/proc/↩
   loadavg" };
/* ... */
g_val_t
load_one_func ( void )
{
   g_val_t val;
   val.f = strtod( update_file(&proc_loadavg), (char ↩
      **)NULL);
   return val;
}
```

5.1.2 Information Systems

WSRF

The XML that Ganglia Resource Provider took from Gmond process through TCP, using XSLT technology is transformed on WSRF to another XML document, that is following the Glue-CE schema. In directory globus_wsrf_mds_usefulrp of globus configuration root, there is the file ganglia_to_glue.xslt where we can focus on the transformation rules. A snippet of interest for the case of ProcessorLoad class is seen in Listing 5.2.

Listing 5.2: WSRF XSLT for Ganglia Information Provider

```
<glue:ProcessorLoad>


<xsl:attribute name="glue:Last1Min">
  <xsl:call-template name="emitProperNumeric">
    <xsl:with-param name="numeric"
    select="floor(100 * METRIC[@NAME='load_one']/@VAL)↩
       "/>
```

```
  </xsl:call-template>
</xsl:attribute>

<xsl:attribute name="glue:Last5Min">
  <xsl:call-template name="emitProperNumeric">
    <xsl:with-param name="numeric"
    select="floor(100 * METRIC[@NAME='load_five']/@VAL↩
       )"/>
  </xsl:call-template>
</xsl:attribute>

<xsl:attribute name="glue:Last15Min">
  <xsl:call-template name="emitProperNumeric">
    <xsl:with-param name="numeric"
    select="floor(100 * METRIC[@NAME='load_fifteen']/↩
       @VAL)"/>
  </xsl:call-template>
</xsl:attribute>

</glue:ProcessorLoad>
```

In Listing 5.2, XPath queries are visible and a multiplication with 100 on CPU load values to get an integer is executed. This is happening in order to avoid the transfer of float values. In the frontend using DOM, a division with 100 is done to get the original CPU load average number back.

BDII

On the other hand, in BDII Processor Load values left as decimal numbers because its easy for LDAP to handle string values of numbers. The transformation from Gmond or Gmetad information provided using XML, is done by Perl XML::Parse and the output is given in MDS format. Some changes to the original Perl script written by Stratos Efstathiadis [30] were performed to

match test-bed's LDAP schema.

5.1.3 Nagios

Nagios was installed in the testbed using YAIM tool. Some user certificates from TEIPIR and Brunel was added to access the interface through the VOMS2HTPASSWD tool.

Custom check command was introduced to configure Nagios with ganglia. Ganglia original check_ganglia.py client was used as shown in Listing 5.1.3.

```
define command{
 command_name  check-ganglia
 command_line  check_ganglia.py -h $HOSTNAME$ -m ↩
    $ARG1$ -w $ARG2$ -c $ARG3$ -s 195.251.70.54 -p ↩
    8649
 }
```

Nagios service check templates were created, to match all nodes of hostgroup named $worker-nodes$, one for each of the three metrics that will be aggregated in Nagios. Under the configuration listed in Listing 5.1.3, options for warning and critical thresholds of Processor Load applied, to notify the contacts that will be set to monitor these services through the rest of Nagios system.

```
define service{
   use                  wn
   hostgroup_name       worker-nodes
   service_description  load_one
   check_command        check-ganglia!load_one!4!5
   action_url           https://osweb.teipir.gr/nagios↩
      /html/pnp4nagios/index.php?host=$HOSTNAME$&srv=↩
      $SERVICEDESC$
```

```
}
```

5.2 Grid Performance and Site Reliability

Grid monitoring in general as proposed by Multi-Level Monitoring in EGEE SA1, is about availability and reliability monitoring. There are threshold values for these metrics for a production site and SLA calculation make use of these metrics.

The main purpose of performance monitoring, when examined from this point of view, is to count jobs that are successfully being submitted to a CE. Site reliability is calculated using that metric.

For a system engineer, performance monitoring for system administration has to do with Processor Load metrics and not with jobs failed or executed successfully. With that in mind, a system engineer may take capacity management decisions to scale-out the Computing Element, or even scale-up a single node of it.

This is the main goal of this project, to provide tools that allow the aggregation of Processor Load information to system engineers, and not job counting tools such as SAM, Gridview and MyEGEE or MyEGI. Ganglia is the event source that publish the metrics taken from Linux kernel as described, and multiple information services or even Nagios may be used to provide that information to users.

5.3 Information Services Scaling

5.3.1 LDAP

LDAP as the core technology of MDS2 has been investigated [14] and proved that scales and performs good when the data are kept in cache. The performance of the information system, when it is accessed by a large number of concurrent users, degrades dramatically if data caching is not used.

Compared with WSRF, it **performs faster** [31] in small number of users, because of the LDAP base of this information service. LDAP back-end is a Berkeley DB and is implemented in C, versus Java and SOAP over HTTP and XML of WSRF. Caching in LDAP although is reported to have problems.

5.3.2 WSRF

On the other side, MDS4 (WSRF) scales better than MDS2 (LDAP) in **large number** of concurrent users. Both throughput and response in queries in large number of queries are consistent, which allows the use of MDS4 for robust operations.

WSRF also supports **many query languages**, unlike BDII which supports only LDAP queries. By default it is distributed only with support of XPath queries, but its architecture which is based in Java container and SOAP/XML standards, allow the distribution of the Information Service with other query languages as well.

Adding a Resource Provider in WSRF was much easier than adding an Information Provider in BDII, which makes MDS4 more **extensible** than MDS2. Even though in BDII a simple wrapper had to be installed in BDII GIP configuration directory, the Perl script (or the Python one provider by Ganglia) had to be changed to reach the running BDII instance of Glue schema. MDS4 and WSRF is based in XML and XSLT transformation mechanisms that allow the easy addition of other Providers of a site. It is even allowed to aggregate Index Services from other sites using the $downstream$ option to register them in the local Index Service.

BDII LDAP based configuration is complex due to OID namespaces and naming authorities and is not portable with other information systems. WSRF configuration is much easier and **flexible** because of the XML-based information model, which generally is considered more friendly.

Reliability is another factor where WSRF is better than LDAP. BDII after a long period of heavy activity has memory usage problems and requests may

have delay in being served, so periodically service restarts are needed, unlike WSRF Index which has been tested under heavy load and keeps serving without problems [31].

Although WSRF in many factors is dominating, during this project some problems occured when some nodes of the Computing Element are down. Some deserialization exceptions appear in the container log file for a few minutes until the WSRF learn about the new state of the cluster from Gmond.

Chapter 6

Conclusions

Grid monitoring is an important factor when working with Grid Systems. Reports extracted from monitoring systems, support decisions for capacity management, and prove that a system meets requirements needed for Service Level Agreements.

Metrics for Computing Element performance monitoring usually are:

1. **Total Jobs** in running or waiting to run state.
2. Individual per working node **Processor Load**.
3. **Benchmarking** metrics such as FLOP/s.

This project focused in the calculation, aggregation and transfer to present the metrics for Processor Load of working nodes of a Computing Element. Running and waiting jobs monitoring is extremely analyzed under availability monitoring research.

Calculation using the number of processes waiting in the queue of kernel scheduler were used. It is explained why this metric is more accurate than the classic percentage CPU usage.

Aggregation and transfer of metrics where examined in many different levels, from the multicasted XDR of Gmond to the resource providers of the information service that were used.

Information Service is core technology that used in Grid Computing. It has evolved in parallel with Middleware. Current version of the Monitoring

and Discovery Service standard has reached the MDS4, introducing the use of Web Services. MDS2 was based in LDAP, which is still used by some systems to discover services.

Nagios bulk aggregation features using NPCD and MSG-Nagios-Bridge also provide a method to aggregate Ganglia metrics without the use of the MDS.

Presentation were simply examined using:

1. **DOM technology** to parse XML taken from WSRF through WebMDS interface using XPath

2. simple **LDAP** communication to take the metrics from the BDII Information Service

6.1 Conclusions

Every aspect of grid monitoring keeps **scalability in mind**. Distribution of metrics in all worker nodes using Gmond is the key to keep redundant that information.

BDII Information Service is great for use in site level environment, as LDAP performs faster than WSRF in less than a hundred users.

Site monitoring needs Nagios and Ganglia for a few hundreds of nodes. Nagios web interface is enough for site-level host and service view. Ganglia web interface supports good aggregation of many clusters, to the Region-level.

WSRF caching feature, Indexer and Aggregator Framework scales better and may be used for regional and top level monitoring.

Heterogeneous environments should **rely on standards** in order to interoperate and stay reliable. Even the Information Services mentioned above stick to the Glue schema.

A tool may seem to be great for its purpose, but after focusing on a need is may be discontinued, such as SAM. This example is taken from the availabil-

ity monitoring, as its frontend was replaced by MyEGI and Nagios keeping its back-end in MRS.

6.2 Further Work

6.2.1 Aggregation

Additional investigation is required to be carried out for the aggregation of collected information. WSRF offer the Aggregator Framework, which downstream information from many EPRs and deliver it through the Index service. Scaling such information in the Regional monitoring level may introduce issues that only in that scale will be visible.

6.2.2 Benchmarking

Performance metrics which are produced using benchmarking software is a good standard way of measuring and advertising the performance of Computing Elements. Such metrics are not efficient for regularly runs and frequent changes because they are cost effective performance wise. It is good although to present the performance ability of a CE. Rare periodic execution and push in the information service, will offer a good metric to select one CE of another.

6.2.3 Storage element performance monitoring

Storage Element performance monitoring is an interesting area for research. Vendors that produce enterprise level storage systems also offer tools to monitor the performance of the storage system. The most used metric is I/O Operations per second and throughput in GByte per second, in every aspect of a storage system.

Storage systems are also provided by different vendors and may also be deployed as custom solution. Fiber Channel solutions versus Clustered File System cases are examined [32] as a chapter of distributed systems, storage oriented.

Hadoop is the most commonly used distributed data storage. In CMS, an experiment that is known for its huge needs in data storage, Hadoop has been adopted [33] as the distributed file system. Grid computing community has adopted the use of Total and Used GB per Storage Element (as in Computing Element), but there is still enough work on the performance monitoring.

Bibliography

[1] M. Li and M. Baker, *The grid: core technologies*. John Wiley & Sons Inc, 2005.

[2] S. Fisher, "DataGrid information and monitoring services architecture: design, requirements and evaluation criteria'," tech. rep., Technical Report, DataGrid, 2002.

[3] G. Taylor, M. Irving, P. Hobson, C. Huang, P. Kyberd, and R. Taylor, *Distributed monitoring and control of future power systems via grid computing*. IEEE, 2006.

[4] A. Kertész and P. Kacsuk, "A Taxonomy of Grid Resource Brokers," in *6th Austrian-Hungarian Workshop on Distributed and Parallel Systems (DAPSYS*, pp. 318–366, Basil Blackwell, 2006.

[5] B. Tierney, R. Aydt, D. Gunter, W. Smith, M. Swany, V. Taylor, and R. Wolski, "A grid monitoring architecture," in *The Global Grid Forum GWD-GP-16-2*, Citeseer, 2002.

[6] X. Zhang, J. L. Freschl, and J. M. Schopf, "A performance study of monitoring and information services for distributed systems," in *12th IEEE International Symposium on High Performance Distributed Computing, 2003. Proceedings*, pp. 270–281, 2003.

[7] A. J. Wilson, R. Byrom, L. A. Cornwall, M. S. Craig, A. Djaoui, S. M. Fisher, S. Hicks, R. P. Middleton, J. A. Walk, A. Cooke, and Others, "Information and monitoring services within a grid environment," in *Computing in High Energy and Nuclear Physics (CHEP)*, Citeseer, 2004.

[8] S. Fisher, "Relational model for information and monitoring," in *Global Grid Forum, GWD-Perf-7*, vol. 1, 2001.

[9] R. Byrom, B. Coghlan, A. Cooke, R. Cordenonsi, L. Cornwall, and Others, "Production services for information and monitoring in the grid," *AHM2004, Nottingham, UK*, 2005.

[10] D. Bonacorsi, D. Colling, L. Field, S. Fisher, C. Grandi, P. Hobson, P. Kyberd, B. MacEvoy, J. Nebrensky, H. Tallini, and S. Traylen, "Scalability tests of R-GMA-based grid job monitoring system for CMS Monte Carlo data production," *IEEE Transactions on Nuclear Science*, vol. 51, pp. 3026–3029, December 2004.

[11] R. Byrom, D. Colling, S. Fisher, C. Grandi, P. Hobson, P. Kyberd, B. MacEvoy, J. Nebrensky, and S. Traylen, *Performance of R-GMA for Monitoring Grid Jobs for CMS Data Production*. IEEE, 2005.

[12] G. Von Laszewski and I. Foster, "Usage of LDAP in Globus," *Mathematics and Computer Science Division, Argonne National Laboratory*, 1998.

[13] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid information services for distributed resource sharing," in *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, 2001.

[14] X. Zhang and J. M. Schopf, "Performance analysis of the globus toolkit monitoring and discovery service, mds2," *Arxiv preprint cs/0407062*, 2004.

[15] Y. Liu and S. Gao, "Wsrf-based distributed visualization," in *Cluster Computing and the Grid, 2009. CCGRID '09. 9th IEEE/ACM International Symposium on*, pp. 615 –619, May 2009.

[16] Luciano Gaido, "DSA1.2.2: Assessment of production service status," 2010.

[17] M. Gerndt, R. Wismuller, Z. Balaton, G. Gombás, P. Kacsuk, Z. Németh, N. Podhorszki, H. L. Truong, T. Fahringer, M. Bubak, and Others, "Performance Tools for the Grid: State of the Art and Future, APART White Paper," 2004.

[18] S. Zanikolas, *Importance-Aware Monitoring for Large-Scale Grid Information Services*. PhD thesis, the University of Manchester, 2007.

[19] S. Andreozzi, N. De Bortoli, S. Fantinel, A. Ghiselli, G. L. Rubini, G. Tortone, and M. C. Vistoli, "Gridice: a monitoring service for grid systems," *Future Gener. Comput. Syst.*, vol. 21, pp. 559–571, April 2005.

[20] G. Tsouloupas and M. Dikaiakos, "Gridbench: a tool for benchmarking grids," in *Grid Computing, 2003. Proceedings. Fourth International Workshop on*, pp. 60 – 67, Nov. 2003.

[21] D. Guo, L. Hu, M. Zhang, and Z. Zhang, "Gcpsensor: a cpu performance tool for grid environments," in *Quality Software, 2005. (QSIC 2005). Fifth International Conference on*, pp. 273 – 278, 2005.

[22] S. Browne, J. Dongarra, N. Garner, G. Ho, and P. Mucci, "A portable programming interface for performance evaluation on modern processors," *International Journal of High Performance Computing Applications*, vol. 14, no. 3, p. 189, 2000.

[23] T. Audience, A. Documents, and D. M. Overview, "Nagios - Distributed Monitoring Solutions The Industry Standard in IT Infrastructure Monitoring Nagios - Distributed Monitoring Solutions," *ReVision*, pp. 9102–9104, 2011.

[24] E. Imamagic and D. Dobrenic, "Grid infrastructure monitoring system based on Nagios," in *Proceedings of the 2007 workshop on Grid monitoring*, p. 28, ACM, 2007.

[25] P. R. Hobson, E. Frizziero, C. Huang, M. R. Irving, T. Kalganova, P. Kyberd, F. Lelli, A. Petrucci, R. Pugliese, G. Taylor, and R. Taylor, *Grid Computing Technologies for Renewable Electricity Generator Monitoring and Control*. IEEE, June 2007.

[26] C. Huang, P. R. Hobson, G. A. Taylor, and P. Kyberd, *A Study of Publish/Subscribe Systems for Real-Time Grid Monitoring*. IEEE, March 2007.

[27] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributed computations," in *High Performance Distributed Computing, 1997. Proceedings. The Sixth IEEE International Symposium on*, pp. 365 –375, Aug. 1997.

[28] T. Papapanagiotou and G. Dilintas, "Free software thin client computing in educational environment," in *3rd International Scientific Conference eRA*, (Aegina, Greece), 09/2008 2008.

[29] Z. Balaton, P. Kacsuk, N. Podhorszki, and F. Vajda, "Use cases and the proposed grid monitoring architecture," tech. rep., Computer and Automation Research Institute of the Hungarian Academy of Sciences, 2001. http://www.lpds.sztaki.hu/publications/reports/lpds-1-2001.pdf, 2001.

[30] S. Efstathiadis, "A simple ganglia MDS information provider." http://www.star.bnl.gov/public/comp/Grid/Monitoring/SimpleGangliaIP.html.

[31] J. Schopf, M. D'arcy, N. Miller, L. Pearlman, I. Foster, and C. Kesselman, "Monitoring and discovery in a web services framework: Functionality and performance of the globus toolkit's MDS4," in *Proceedings of the 15th IEEE International Symposium on High Performance Distributed Computing, IEEE Computer Society Press, Los Alamitos*, Citeseer, 2006.

[32] M. Brzezniak, N. Meyer, M. Flouris, R. Lachaiz, and A. Bilas, "Analysis of Grid Storage Element Architectures: High-end Fiber-Channel vs. Emerging Cluster-based Networked Storage," *Grid Middleware and Services*, pp. 187–201, 2008.

[33] B. Bockelman, "Using hadoop as a grid storage element," *Journal of Physics: Conference Series*, vol. 180, no. 1, p. 012047, 2009.

[34] J. Huo, L. Liu, L. Liu, Y. Yang, and L. Li, "A Study on Distributed Resource Information Service in Grid System," in *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, vol. 1, pp. 613–618, 2007.

[35] J. Novak, "Nagios SAM Portal for ROCs," 2009.

[36] S. Zanikolas and R. Sakellariou, "A taxonomy of grid monitoring systems," *Future Generation Computer Systems*, vol. 21, no. 1, pp. 163 – 188, 2005.

[37] B. K. Singh, A. Amintabar, A. Aggarwal, R. D. Kent, A. Rahman, F. Mirza, and Z. Rahman, "Secure grid monitoring, a web-based framework," in *Proceedings of the first international conference on Networks for grid applications*, GridNets '07, (ICST, Brussels, Belgium, Belgium), pp. 15:1–15:7, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2007.

[38] K. Goel and I. Chana, "Agent-Based Resource Monitoring for Grid Environment," *ICON 2008*, p. 62, 2008.

[39] T. Kiss, P. Kacsuk, G. Terstyanszky, and S. Winter, "Workflow level interoperation of grid data resources," in *Cluster Computing and the Grid, 2008. CCGRID '08. 8th IEEE International Symposium on*, pp. 194 – 201, May 2008.

School of Engineering & Design

Electronic & Computer Engineering

Brunel UNIVERSITY WEST LONDON

Industrial Mentor's Report on a MSc Dissertation

Student's name: Theofylaktos Papapanagiotou

Dissertation Title: Grid Monitoring

I confirm / cannot confirm* that the MSc dissertation submitted by the above student truly reflects the respective contributions of the student and others, with whom the student has worked during the MSc project.

∗ delete as appropriate

Comments:

Signed:

Name: Dr.Paul Kyberd

Date: