Agent-Based Resource Monitoring For Grid Environment

A thesis
submitted in partial fulfillment of the requirements
for the award of degree
of

Master of Engineering
in
Software Engineering

By:
Kunal Goel
(Roll No. 80631008)

Under the supervision of:
Ms. Inderveer Chana
Senior Lecturer
CSED

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR UNIVERSITY
PATIALA – 147004

JULY 2008

Certificate

I hereby certify that the work which is being presented in the thesis entitled, **"Agent-Based Resource Monitoring for Grid Environment"**, in partial fulfillment of the requirements for the award of degree of Master of Engineering in Software Engineering submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Ms. Inderveer Chana* and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

(***Kunal Goel)***

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(**Ms. Inderveer Chana)**
Computer Science and Engineering Department
Thapar University
Patiala

Countersigned by

(SEEMA BAWA)
Professor & Head
Computer Science & Engineering. Department
Thapar University
Patiala

(R.K.SHARMA)
Dean (Academic Affairs)
Thapar University,
Patiala.

Acknowledgement

I wish to express my deep gratitude to Ms. Inderveer Chana, Senior Lecturer, Computer Science & Engineering Department for providing her immense help, guidance, simulating suggestions and encouragement all the time. She always provided a motivating and enthusiastic atmosphere to work with; it was a great pleasure to do this thesis under her supervision.

I am equally thankful to Dr. Seema Bawa, Head, Computer Science & Engineering Department, for the motivation and inspiration that triggered me for the thesis work. I am also thankful to Mr. Maninder Singh, Assistant Professor, Computer Science & Engineering Department, a nice person, an excellent teacher and a well-credited researcher, who always encouraged me to keep going with work and always advised me with his invaluable suggestions. I am most indebted to Mr. Prateek Bhatia and Ms. Damandeep Kaur for their help, assistance and suggestions during my work.

I am also thankful to Mr. Balwinder Singh for providing me with the material support. I would also like to express my appreciation to the TUGrid Group. The group held weekly meetings to address various issues and share experiences. Thus, it invoked better understanding of the work and acted as a rostrum for information sharing and problem solving.

I cannot skip mentioning the help and feedback which my colleagues Ms. Shashi, Ms. Anju, Sanmeet, Rohit , Shilpi, Maninder, Neha, Lokesh, Kabir, Vineet and Satyarth have provided throughout my work. I want to thank them all; they were always there at the need of the hour and provided all the help and valuable hints, which I required for the completion of the thesis.

Finally, my special thanks go to my family for their never-ending love and support. Nothing would have ever been possible without my parent's unconditional love and encouragement.

The most valuable thing I acquired from the two years study in TIET is to protect, survive and endure myself in this world with increased confidence and additional faith in my abilities to achieve.

Last but not the least I would like to thank God for not letting me down at the time of crisis and showing me the silver lining in the dark clouds.

Kunal Goel
(80631008)

Abstract

Grid is an emerging technology for enabling resource sharing and coordinated problem solving in dynamic multi-institutional heterogeneous virtual organizations. Grid computing is being adopted in various areas from academic, industry research to government use. Grids are becoming platforms for high performance and distributed computing. Grid computing is the next generation IT infrastructure that promises to transform the way organizations and individuals compute, communicate and collaborate. The goal of Grid computing is to create the illusion of a simple but large and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various combinations of resources.

One of the fundamental operations needed to support location-independent computing is resource monitoring. Grid Resource Monitoring is the process of collecting information concerning the characteristics and status of resources of interest. In the Grid environment, resources may belong to different institutions, have different usage policies and pose different requirements on acceptable requests. Grid resources may dynamically join and leave, their membership varies over time; even in fairly static settings, resource availability is subject to failures. Given this transient nature, users must be supported in finding and keeping track of resources of interest; which is the main purpose of Grid Information Services (GIS). In order for information services to address the mentioned user needs, they must systematically collect information regarding the current and, sometimes, past status of Grid resources. In addition to information services, monitoring is also crucial in a variety of cases such as scheduling, data replication, accounting, performance analysis and optimization of distributed systems or individual applications, self-tuning applications, and many more. Also, given the increasing number of Grid resources, real-time monitoring of their availability and utilization is becoming essential for effective management, particularly regarding the detection of faults and bottlenecks.

Focus of this thesis is to analyze existing Resource Monitoring Systems for Grid environment and an Agent-Based Resource Monitoring System has been proposed, designed and implemented for Grid environment.

Table of Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter focuses on a brief introduction of Grid computing. It presents a brief history of Grid and explains characteristics and benefits of Grid. It also gives the organization of the thesis along with a brief idea about the contents of each of the chapters.

1.1 Grid Computing

Humans continuously keep developing new methodologies to solve the complex science and engineering problems. The requirements of these new scientific methods, however, tend to supersede capabilities of the contemporary underlying technology. High energy physics is one such scientific domain; it has immensely helped in exploring the phenomena that range from the smallest particles of nature to the largest galaxies in the Universe. Some of the greatest breakthroughs in high-energy physics rely on experiments that generates huge amounts of data. The analysis of this data can be facilitated by computing technologies like high-performance computing, simulations, data analysis and distributed computation – which requires availability of enormous computing power. However, with the current pace of advancements in high-energy physics experiments, the available computing power may need to be harnessed much more effectively [1]. Similarly a crisis management team responds to a chemical spill by using local weather and soil models to estimate the spread of the spill, determining the impact based on population location as well as geographic features such as rivers and water supplies, creating a short-term mitigation plan (perhaps based on chemical reaction models), and tasking emergency response personnel by planning and coordinating evacuation, notifying hospitals, and so forth [2]. It also requires huge computational power. A *solution* to these complex scientific problems is *Grid computing [3] [4]*.

Grid Computing has emerged as a new and important field and can be visualized as an enhanced form of Distributed Computing. With the advent of new technology, it has been

realized that parallelizing sequential applications could yield faster results and sometimes at a lower cost. Possessing multiprocessor systems was not possible for everyone. Thus, organizations started looking for a better and feasible alternative. It was found that though every organization possessed a large number of computers, which meant that they had huge processing power, but it remained underutilized. A new type of Computing then came into existence, known as Distributed Computing. In Distributed Computing, the problem to be solved is divided into numerous tasks that are then distributed to various computers for processing. The computers being used are generally connected through a Local Area Network (LAN). Also the problem needs to be divided into modules that can be executed in parallel to each other [2] [3].

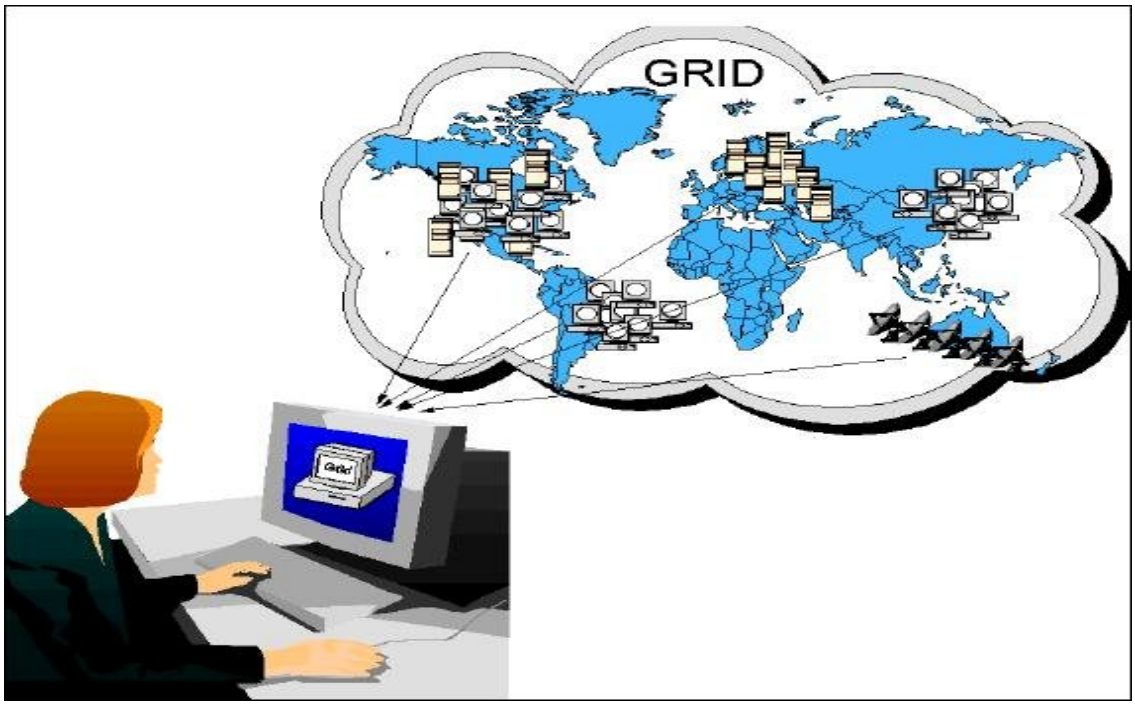


Figure 1.1 The Grid virtualizes heterogeneous geographically disperse resources [5]

Grid computing is the next generation IT Infrastructure that promises to transform the way organizations and individuals compute, Communicate and collaborate. The goal is to create the illusion of a simple yet large and powerful self-managing virtual computer out of a large collection of connected heterogeneous systems sharing various resources. Grid

computing [3] [7] spans multiple organizations, machine architectures and software boundaries to provide unlimited power, collaboration and information access to everyone connected to a Grid. In other words, Grid is a network of heterogeneous resources working in collaboration to solve problems that cannot be addressed by the resources of any one organization [4]. The key to realizing the benefits of Grid computing is standardization, so that the diverse resources that make up a modern computing environment can be discovered, accessed, allocated, monitored, and in general managed as a single virtual system even when provided by different vendors and/or operated by different organizations [6].

Ian Foster defined the Grid as [4]:

"A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities."

IBM's Grid Computing has put forward the following statement for Grid computing [8]:

"Grid is the ability, using a set of open standards and protocols, to gain access to applications and data, processing power, storage capacity and a cast array or other computing resources over the Internet. A Grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of resources distributed across multiple administrative domains based on the resources availability, capacity, performance, cose and users' quality-of-service requirements"

The real and specific problem that underlies the Grid concept is *coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations*. A Virtual Organization (VO) [2] [9] is a set of individuals and institutions defined by a definite set of sharing rules like what is shared, who is allowed to share, and the conditions under which the sharing takes place [2].

1.1.1 History of Grid

The term “the Grid” was coined in the mid 1990s to denote a proposed distributed computing infrastructure for advanced science and engineering [2]. Back in 1965 the developers of an operating system called Multics (an ancestor of UNIX) presented a vision of "computing as a utility" - in many ways uncannily like the Grid vision today. Access to the computing resources was envisioned to be exactly like access to utility such as electricity - something that the client connects to and pays for according to the amount of use. This led to coining of the term “Grid” [10].

The basic idea behind the Grid is sharing computing power. Nowadays most people have more than enough computing power on their own PC, hence sharing is unnecessary for most purposes. However back in the sixties and seventies, sharing computer power was essential. At that time, computing was dominated by huge mainframe computers, which had to be shared by whole organizations. A number of applications needed more computing power than can be offered by a single resource or organization in order to solve them within a feasible/reasonable time and cost. This promoted the exploration of logically coupling geographically distributed high-end computational resources and using them for solving large-scale problems. Such emerging infrastructure was called computational Grid, and led to the popularization of a field called Grid computing [2] [10]. Table 1.1 below duplicate historical background of the Grid:

Table-1.1 Historical Background of the Grid [10]

Technology	**Year**
Networked Operating Systems	1979-81
Distributed Operating Systems	1988-91
Heterogeneous computing	1993-94
Parallel and distributed computing	1995-96
The Grid	1998

Grid Computing is comparable to electrical Grids. As electrical Grids provide consistent, pervasive, dependable, transparent access to electricity irrespective of its source, similarly computing Grids offer services to users without letting them know where the source is located. Analogous to the power Grid, Grid environments create virtual providers, carriers and consumers to avoid single points of failure, and deliver a continuous stream of computing power [10].

1.1.2 Characteristics of Grid

There are many characteristics of Grid as depicted in Figure 1.2 which have been described below [11]:

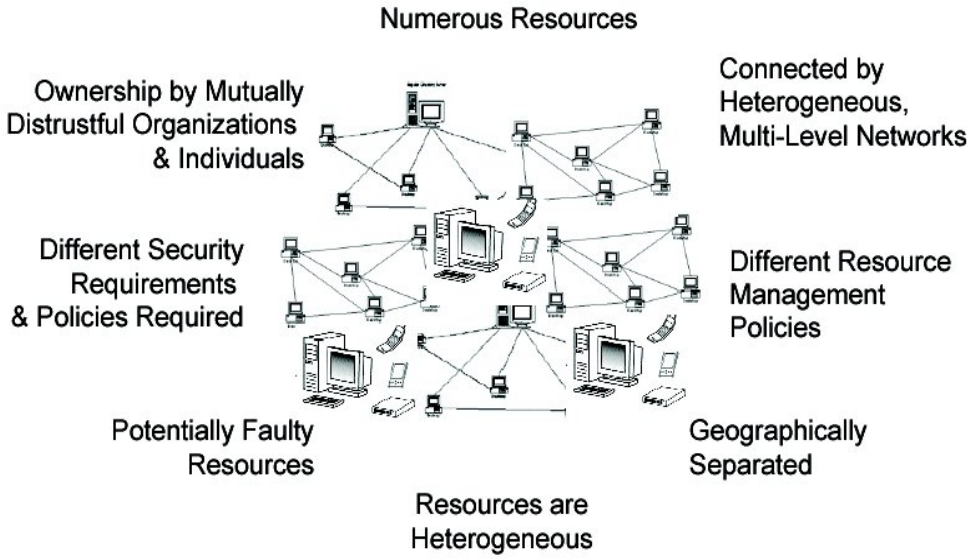


Figure 1.2 Characteristics of Grids [28]

Large scale: A Grid must be able to deal with a number of resources ranging from just a few to millions. This raises the very serious problem of avoiding potential performance degradation as the Grid size increases.

Geographical distribution: Grid's resources may be located at distant places.

Heterogeneity: A Grid involves a multiplicity of resources (both hardware and software) that are heterogeneous in nature and might span numerous administrative domains across wide geographical distances.

Resource sharing: Resources in a Grid belong to many different organizations that allow other organizations (i.e. users) to access them. Nonlocal resources can thus be used by applications, promoting efficiency and reducing costs.

Resource coordination: Resources managed via different policies in a Grid must be coordinated in order to provide aggregated computing capabilities.

Multiple administrations: Resources are owned and managed by different, potentially mutually distrustful organizations and individuals that likely have different security policies and practices [12]. Each organization may establish different security and administrative policies under which their owned resources can be accessed and used. As a result, the already challenging network security problem is complicated even more with the need of taking into account all different policies.

Transparent access: A Grid should be seen as a single virtual computer. Users do not have to worry about system details (e. g. location, operating system, accounts)

Dependable access: A Grid must assure the delivery of services under established Quality of Service (QoS) requirements. The need for dependable service is fundamental since users require assurances that they will receive predictable, sustained and often high levels of performance.

Consistent access: A Grid must be built with standard services, protocols and interfaces thus hiding the heterogeneity of the resources while allowing its scalability. Without such standards, application development and pervasive use would not be possible.

Dynamicity or Adaptability: In a Grid, a resource failure is the rule, not the exception. In fact, with so many resources in a Grid, the probability of some resource failing is naturally high. The resource managers or applications must tailor their behavior dynamically so as to extract the maximum performance from the available resources and services.

1.1.3 Benefits of Grid

Grid computing provides a number of benefits to user community, developer community and enterprise community. It provides an abstraction/virtualization for resource sharing and collaboration across multiple administrative domains. This section describes the benefits of a Grid environment in detail [5].

Exploiting underutilized resources

In most organizations, computing resources are underutilized. Most desktop machines are busy less than 25% of the time (if we consider that a normal employee works 7 hours a day and that 42 hours a week and that there are 168 hours per week) and even the server machines can often be fairly idle. Grid computing provides a framework for exploiting these underutilized resources and thus has the possibility of substantially increasing the efficiency of resource usages. The easiest use of Grid computing would be to run an existing application on several machines. The machine on which the application is normally run might be unusually busy, the execution of the task would be delayed. Grid Computing should enable the job in question to be run on an idle machine elsewhere on the network [14].

Parallel CPU capacity

The potential for massive parallel CPU capacity [13] is one of the most attractive features of a Grid. In addition to pure scientific needs, such computing power is driving a new evolution in industries such as the bio-medical field, financial modeling, oil exploration, motion picture animation, and many others. The common attribute among such uses is that the applications have been written to use algorithms that can be partitioned into independently running parts.

Collaboration of virtual resources

Another important Grid computing contribution is to enable and simplify collaboration among a wider audience. In the past, distributed computing promised this collaboration and achieved it to some extent. Grid computing takes these capabilities to an even wider

audience, while offering important standards that enable very heterogeneous systems to work together to form the image of a large virtual computing system offering a variety of virtual resources. The users of the Grid can be organized dynamically into a number of virtual organizations [2] [15], each with different policy requirements. These virtual organizations can then share their resources collectively as a larger Grid.

Access to additional resources

In addition to CPU and storage resources, a Grid can provide access to increased quantities of other resources and to special equipment, software, licenses, and other services. The additional resources can be provided in additional numbers and/or capacity. Some machines may have expensive licensed software installed that the user requires. His jobs can be sent to such machines more fully exploiting the software licenses. Some machines on the Grid may have special devices. Most of us have used remote printers, perhaps with advanced color capabilities or faster speeds. Similarly, a Grid can be used to make use of other special equipment. For example, a machine may have a high speed, self-feeding, DVD writer that could be used to publish a quantity of data faster. Some machines on the Grid may be connected to scanning electron microscopes that can be operated remotely. In this case, scheduling and reservation are important.

Reliability

High-end conventional computing systems use expensive hardware to increase reliability. They are built using chips with redundant circuits that vote on results, and contain much logic to achieve graceful recovery from an assortment of hardware failures. The systems are operated on special power sources that can start generators if utility power is interrupted. All of this builds a reliable system due to the duplication of high-reliability components.

Thus, if there is a power or other kind of failure at one location, the other parts of the Grid are not likely to be affected. Grid management software can automatically resubmit jobs to other machines on the Grid when a failure is detected. In critical, real-time situations, multiple copies of the important jobs can be run on different machines

throughout the Grid. Their results can be checked for any kind of inconsistency, such as computer failures, data corruption, or tampering; thus offering much more reliability.

Management

The goal to virtualize the resources on the Grid and more uniformly handle heterogeneous systems will create new opportunities to better manage a larger, more disperse IT infrastructure. It will be easier to visualize capacity and utilization, making it easier for IT departments to control expenditures for computing resources over a larger organization.

Resource Balancing

A Grid federates a large number of resources contributed by individual machines into a greater total virtual resource. For applications that are Grid-enabled, the Grid can offer a resource balancing effect by scheduling Grid jobs on machines with low utilization. This feature can prove invaluable for handling occasional peak loads of activity in parts of a larger organization. This can happen in two ways:

- An unexpected peak can be routed to relatively idle machine.
- If the Grid is already fully utilized, the lowest priority work being performed on the Grid can be temporarily suspended or even cancelled and performed again later to make room for the higher priority work.

Other more subtle benefits can occur using a Grid for load balancing. When jobs communicate with each other, the Internet, or with storage resources, an advanced scheduler could schedule them to minimize communications traffic or minimize the distance of the communications. This can potentially reduce communication and other forms of contention in the Grid.

Finally, a Grid provides an excellent infrastructure for brokering resources. Individual resources can be profiled to determine their availability and their capacity, and this can be factored into scheduling on the Grid. Different organizations participating in the Grid can build up Grid credits and use them at times when they need additional resources. This can

form the basis for Grid accounting and the ability to more fairly distribute work on the Grid.

1.2 Motivation

Grid resources are potentially very large in number and variety; individual resources are not centrally controlled, and they can enter and leave the Grid environment at any time. For these reasons, Resource Monitoring in large-scale Grids can be very challenging. Grid Resource Monitoring is first and foremost requirement for efficient sharing of resources among the various organizations in the Grid environment. It is defined as an act of collecting information concerning the characteristics and status of resources of interest. However, due to the dynamic, heterogeneous and distributed nature of the Grid environment, Resource Monitoring becomes challenging job. In addition to providing information about characteristics and status of resources, Monitoring is also crucial in a variety of cases such as scheduling, data replication, accounting, performance analysis and optimization of distributed systems or individual applications, self-tuning applications, and many more. Because of these reasons focus of this thesis work is on analyzing existing Grid Resource Monitoring systems and to propose an efficient Agent-Based Resource Monitoring System for Grid environment.

1.3 Organization of Thesis

The chapters in the thesis have been organized as follows:

Chapter 2 Describes in detail what Grid computing is. It explains the Grid Architecture, various kinds of Grids, and different topologies of the Grid, Grid Resource Management and Grid Resource Monitoring in detail. Finally it discusses the problems found in existing Grid resource monitoring systems and proposes a solution for that.

Chapter 3 Describes the design of proposed Agent Based Resource Monitoring System, Architecture and basic components.

Chapter 4 Includes the implementation details of the Grid setup and Agent Based Resource Monitoring system.

Chapter 5 summarizes the work presented in this thesis followed by the features that can be incorporated in future for the enhancement of the Grid resource monitoring.

Chapter 2
Literature Review

This chapter describes Grid computing in detail. It explains the Grid Architecture, various kinds of Grids based on the kind of services they provide, and different topologies of the Grid depending on the number of organizations that are a part of a particular Grid environment, Grid Resource Management. Next it discusses the most crucial part of Resource management; Grid Resource Monitoring in detail. And compares the Monitoring approach used in existing Grid middleware available today and some other commercially used Grid monitoring tools. Finally it discusses the problems in existing monitoring systems and proposes a solution for that.

2.1 Grid Computing Concepts

Grid computing has evolved into an important discipline within the computer industry as an evolved version of distributed computing through an increased focus on resource sharing, co-ordination, manageability, and high performance. These characteristics lead to the definition of an architecture for Grid establishment and management and for resource sharing among participants. A new architecture model has been developed for the establishment and management of cross-organizational resource sharing. This new architecture, called *Grid architecture.*

2.1.1 Grid Architecture

Grid architecture identifies the basic components of a Grid system. The Grid architecture defines the purpose and functions of its components, while indicating how these components interact with one another. The main focus of the architecture is on interoperability among resource providers and users in order to establish the sharing relationships. This interoperability, in turn, necessitates common protocols at each layer

of the architectural model [16], which leads to the definition of a Grid protocol architecture as shown in Figure 2.1:

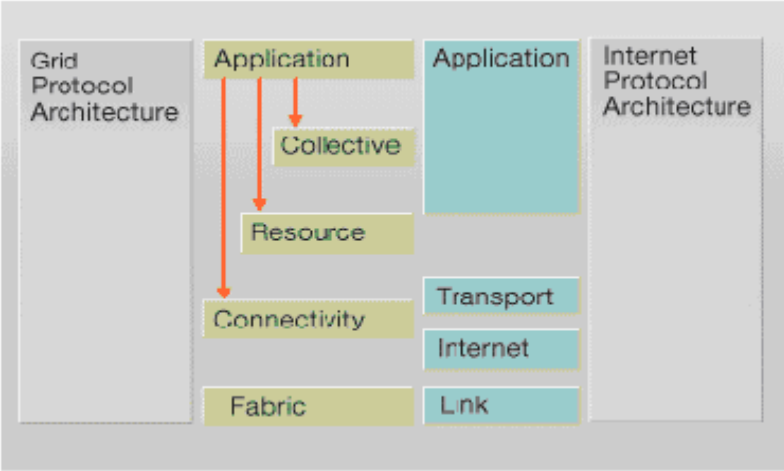


Figure 2.1 The layered Grid Architecture and its relationship to the Internet Protocol Architecture [16]

This protocol architecture defines common mechanisms, interfaces, schema, and protocols at each layer, by which users and resources can negotiate, establish, manage, and share resources. Figure 1 shows the component layers of the Grid architecture and the capabilities of each layer. Each layer shares the behavior of the underlying component layers [16]. The following describes the core features of each of these component layers, starting from the bottom of the stack and moving upward.

Fabric layer: The fabric layer defines the interface to local resources, which may be shared. This includes computational resources, data storage, networks, catalogs, software modules, and other system resources.

Connectivity layer: The connectivity layer defines the basic communication and authentication protocols required for Grid-specific networking service transactions. *Communication protocols* enable the exchange of data between Fabric layer resources. *Authentication protocols* build on communication services to provide cryptographically secure mechanisms for verifying the identity of users and resources.

Resource layer: This layer uses the communication and security protocols (defined by the connectivity layer) to control secure negotiation, initiation, monitoring, accounting, and payment for the sharing of functions of individual resources. The resource layer calls the fabric layer functions to access and control local resources. This layer only handles individual resources, ignoring global states and atomic actions across the resource collection pool, which are the responsibility of the collective layer. Two primary classes of Resource layer protocols can be distinguished:

Information protocols are used to obtain information about the structure and state of a resource, for example, its configuration, current load, and usage policy (e.g., cost).

Management protocols are used to negotiate access to a shared resource, specifying, for example, resource requirements (including advanced reservation and quality of service) and the operation(s) to be performed, such as process creation, or data access. Since management protocols are responsible for instantiating sharing relationships, they must serve as a "policy application point," ensuring that the requested protocol operations are consistent with the policy under which the resource is to be shared. Issues that must be considered include accounting and payment. A protocol may also support monitoring the status of an operation and controlling (for example, terminating) the operation.

Collective layer: While the Resource layer is focused on interactions with a single resource, the next layer in the architecture contains protocols and services (and APIs and SDKs) that are not associated with any one specific resource but rather are global in nature and capture interactions across collections of resources. For this reason, we refer to the next layer of the architecture as the *Collective* layer. This protocol layer implements a wide variety of sharing behaviors using a small number of resource-layer and connectivity-layer protocols.

Application layer: The application layer enables the use of resources in a Grid environment through various collaboration and resource access protocols.

2.1.2 Types of Grid

Depending upon the design objectives and target applications for particular Grid environment Grid systems can be classified into three categories as shown in Figure 2.2.

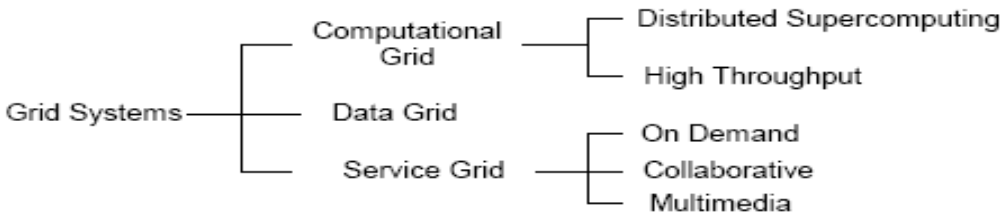


Figure 2.2 Types of Grid [17]

Computational Grid: Computational Grids have higher aggregate computational capacity available for single applications than the capacity of any constituent machine in the system. Depending on how this capacity is utilized, these systems can be further subdivided into *distributed supercomputing* and *high throughput* categories: A *distributed supercomputing Grid* executes the application in parallel on multiple machines to reduce the completion time of a job. A *high throughput Grid* increases the completion rate of a stream of jobs and is well suited for 'parameter sweep' type applications such as Monte Carlo simulations [18].

Data Grid: This *category* is for systems that provide an infrastructure for synthesizing new information from data repositories such as digital libraries or data warehouses that are distributed in a wide area network. Typical applications for these systems include special purpose data mining that correlates information from multiple different data sources. The data Grid initiatives, European DataGrid Project [19] and Globus [20], are working on developing large-scale data organization, catalog, management, and access technologies.

Service Grid: This category is for systems that provide services that are not provided by any single machine. This category is further subdivided as *on-demand*, *collaborative*, and

multimedia Grid systems: A *collaborative Grid* connects users and applications into collaborative workgroup. An *on-demand Grid* category dynamically aggregates different resources to provide new services. A *multimedia Grid* provides an infrastructure for real-time multimedia applications. This requires supporting QoS across multiple different machines whereas a multimedia application on a single dedicated machine may be deployed without QoS [21].

Most ongoing research activities developing Grid systems fall into one of the above categories. Development of truly general-purpose Grid systems that can support multiple or all of these categories remains a hard problem.

2.1.3 Grid Topology

There are three topologies of Grid namely, IntraGrid, InterGrid and ExtraGrid. The simplest of these is the IntraGrid [5], which is comprised merely of a basic set of Grid services within a single organization. The complexity of the Grid design is proportionate to the number of organizations that the Grid is designed to support, and the geographical parameters and constraints. As more organizations join the Grid, the non-functional or operational requirements for security, directory services, availability, and performance become more complex. As more organizations require access to Grid resources, the requirements for increased application layer security, directory services integration, higher availability, and capacity become more complicated. The topological view is shown in Figure 2.3.

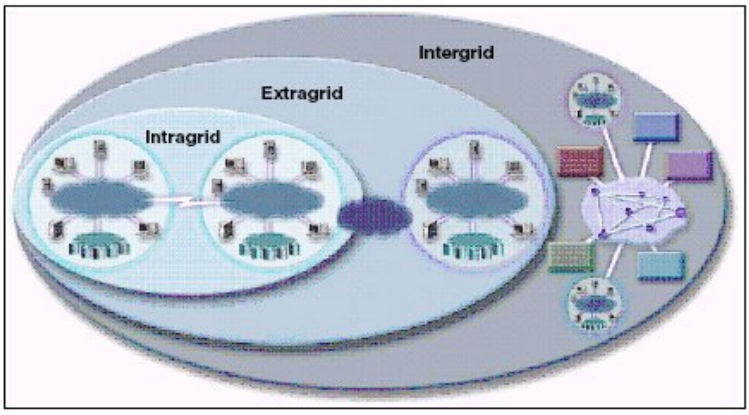


Figure 2.3 topological views of Grids [5]

IntraGrid: A typical IntraGrid [5] topology, as illustrated in Figure 2.4 below, exists within a single organization, providing a basic set of Grid services. The single organization could be made up of a number of computers that share a common security domain, and share data internally on a private network. The primary characteristics of an IntraGrid are a single security provider, bandwidth on the private network is high and always available, and there is a single environment within a single network. Within an IntraGrid, it is easier to design and operate computational and data Grids.

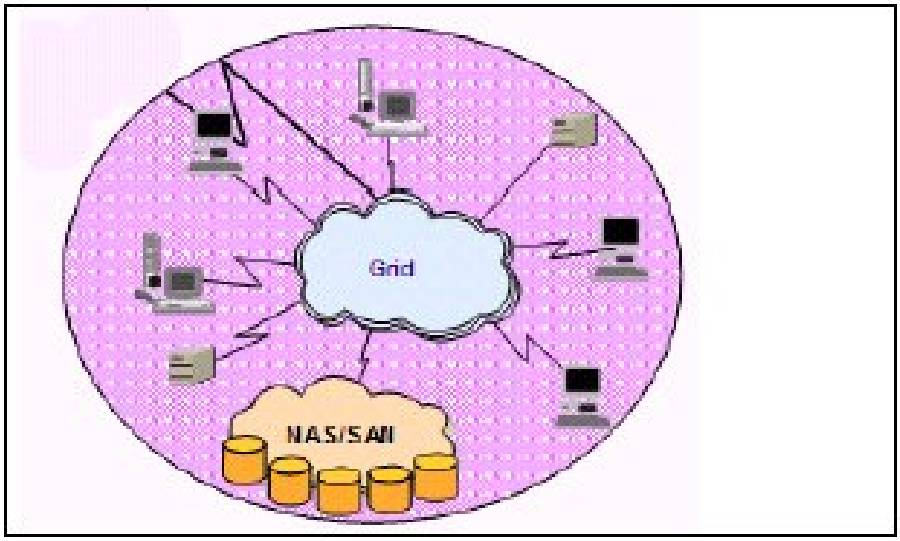


Figure 2.4 IntraGrid [5]

ExtraGrid: Based on a single organization, the ExtraGrid [5] expands on the concept by bringing together two or more IntraGrids. An ExtraGrid, as illustrated in the Figure 2.5 given below, typically involves more than one security provider, and the level of management complexity increases. The primary characteristics of an ExtraGrid are dispersed security, multiple organizations, and remote/WAN connectivity. Within an ExtraGrid, the resources become more dynamic and the Grid needs to be more reactive to failed resources and failed components.

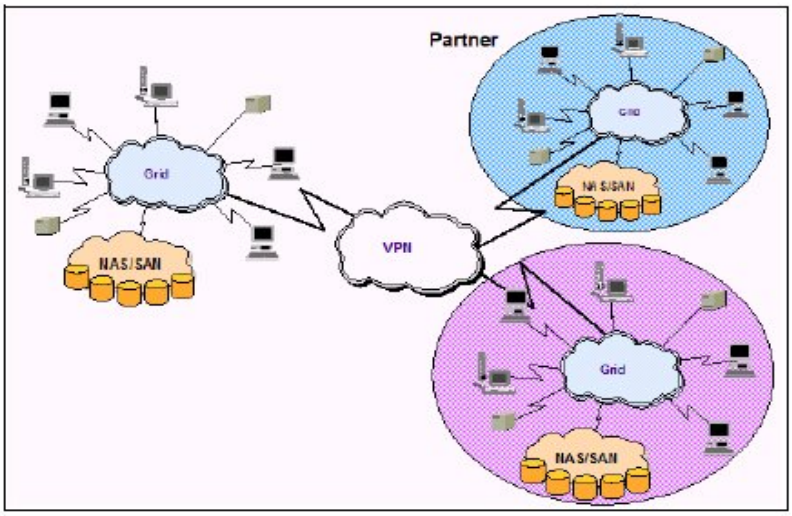


Figure 2.5 ExtraGrid [5]

InterGrid: An InterGrid [5] requires the dynamic integration of applications, resources, and services with patterns, customers, and any other authorized organizations that will obtain access to the Grid via the internet/WAN. An InterGrid topology, as illustrated in Figure 2.6, is primarily used by engineering firms, life science industries, manufacturers, and by businesses in the financial industry. The primary characteristics of an InterGrid include dispersed security, multiple organizations, and remote/WAN connectivity. The data in an InterGrid is global public data, and applications, both vertical and horizontal, must be modified for a global audience.

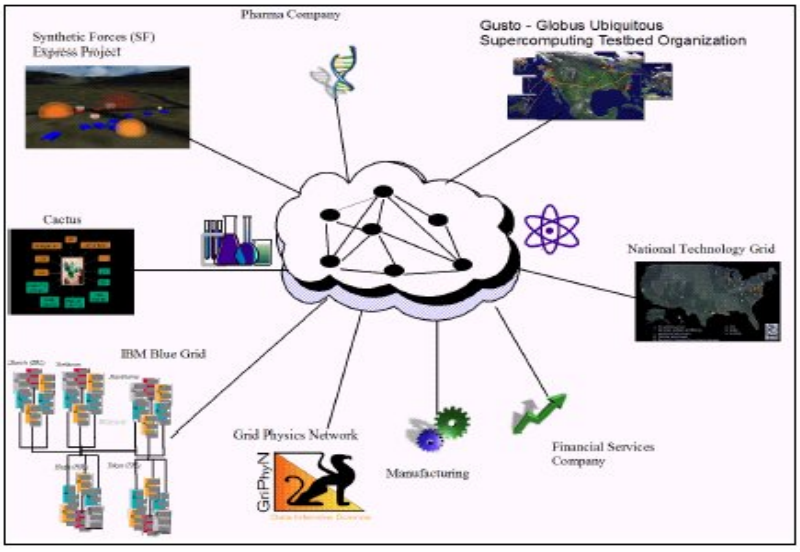


Figure 2.6 InterGrid [5]

Above discussion provide us with a good knowledge of basic concepts of Grid Computing, its characteristics, types and topology. Next Grid Resource Management will be discussed

2.1.4 Grid Resource Management

Grid Resource Management is defined as the process of identifying requirements, matching resources to applications, allocating those resources, and scheduling and monitoring Grid resources over time in order to run Grid applications as efficiently as possible. Resource discovery is the first phase of resource management. Scheduling is the second and Resource monitoring is the next step. Scheduling process directs the job to appropriate resource and monitoring process monitors the resources. Based upon the resource information provided by the resource monitor, the resources which will be heavily loaded will act as server of task and the resources which are Lightly Loaded will act as receiver of task. Task will be migrated from heavily loaded node to lightly loaded node. The ability to monitor these Grid resources and services needed by applications is essential for providing seamless access to them on the Grid.

2.2 Grid Resource Monitoring

Monitoring systems, in the broadest sense, are tools that report some set of measurements to higher-level services [22].

In the Grid environment, the resources may belong to different institutions, have different usage policies and pose different requirements on acceptable requests. Moreover, Grid resources are potentially very large in number and variety; individual resources are not centrally controlled, and they can enter and leave the Grid systems at any time. One of the fundamental operations needed to support location-independent computing is Resource Monitoring; it is the act of collecting information concerning the characteristics and status of resources of interest.

The goal of Grid monitoring is to measure and publish the state of resources at a particular point in time. To be effective, monitoring must be "end-to-end", meaning that all components in an environment must be monitored. This includes software (e.g. applications, services, processes and operating systems), host hardware (e.g. CPUs, disks, memory and sensors) and networks (e.g. routers, switches, bandwidth and latency). Monitoring data is needed to understand performance, identify problems and to tune a system for better overall performance. Fault detection and recovery mechanisms need the monitoring data to help determine if parts of an environment are not functioning correctly, and whether to restart a component or redirect service requests elsewhere. A service that can forecast performance might use monitoring data as input for a prediction model, which could in turn be used by a scheduler to determine which components to use [26].

2.2.1 Grid Monitoring Requirements

A set of general requirements for monitoring systems that are considered important follows; these may vary considerably depending on the use cases [23] that need to be supported by a specific system.

Scalability: Monitoring systems have to cope efficiently with a growing number of resources, events and users. This scalability can be achieved as a result of good performance and low intrusiveness. The former guarantees that a monitoring system will achieve the needed throughput within an acceptable response time in a variety of load scenarios. The latter refers to the intrusiveness imposed by the monitoring system to the entities being monitored. Intrusiveness is typically measured as a function of host (processor, memory, I/O) and network load (bandwidth) generated by the collection, processing and distribution of events. It can be seen that a monitoring system with moderate performance will be insufficient in heavy load scenarios (many resources and/or users), whereas a system inflicting non-trivial intrusiveness degrades the capacity of the monitored resources [24].

Extensibility: A monitoring system must be extensible with respect to the supported resources and hence the events generated by the latter. To this end, desirable features include an extensible and self-describing event encoding method (i.e., data format); an event schema service which allows controlled and dynamic extensions/modifications; a producer-consumer protocol that can accommodate new event types [24] [25].

Portability: The Grid infrastructure is heterogeneous so portability of a monitoring system, and particularly that of the sensors, is of major importance; otherwise a system is unable to monitor specific types of resources, and hence support their visibility on the Grid. The concept of portability also applies to the generated events, meaning that any encapsulated measurements must be platform independent [24] [25].

Security: Certain scenarios may require a monitoring service to support security services such as access control, single or mutual authentication of parties, and secure transport of monitoring information [24].

Fault tolerance: The system should be robust to infrastructure failures like node and network failures of various types. As systems scale in the number of nodes, failures become both inevitable and commonplace. The system should be able to localize such failures so that the system continues to operate and delivers useful service in the presence of failures and the effect is minimized [25].

Interoperability: it is a property referring to the ability of diverse systems and organizations to work together (inter-operate). As the Grid environment is heterogeneous and full of diverse resources and system so interoperability is must for a Grid monitoring system.

2.2.2 Grid Monitoring Process

Grid Monitoring Process is a distributed activity. Distributed systems are composed of independent components. For example, a Grid infrastructure is composed of servers, desktops, and the network connecting the different components. The different stages of Grid monitoring are data collection, data processing, data distribution, and data presentation.

***Data Collection*:** The first stage in Grid monitoring is to gather or collect data from different components through *sensors*. The sensors gather specific information like the bandwidth usage, CPU and memory usage, health of the node, and so on. There are two ways in which the data can be gathered by the sensors namely *active* and *passive monitoring*. In the former case, the sensors actually introduce load or messages in the distributed system to gather the information. Passive monitoring, on the other hand, gather the information by looking at the link, CPU utilization, memory, without actually introducing any message or load in the system. Once the data have been collected, the sensors report the data in a specific format or structure to the other layers. The gathered data can be static in nature like the network topology, configurations of machines, or dynamic like the utilization of the system, system load, network available bandwidth, and so on. Moreover, the data can be collected in a time-based or in an event based manner. In the former case, data is collected in a periodic manner, while in the latter, data is collected only when a specified event takes place [25].

Data Processing: This step involves getting information out of collected data. This step is generally application-specific and may take place during any stage of the monitoring process. Typical examples include filtering according to some predefined criteria, or summarizing a group of events. This step is essential as sensors may gather different types of data from the components. However, an application may not be concerned about all the gathered data and may only be concerned about certain information, or in some cases aggregated information. Due to the hierarchical nature of many distributed monitoring systems, there is a need to process and filter data at every step of the hierarchy [25].

Data Transmission: This step involves the transmission of collected and processed data to the different entities interested. Transmission involves sending the data in a format understood by other parties over a transmission medium for example the network. Depending on the nature of the data, different authentication, encryption, and integrity mechanisms can be applied to secure the transmitted data [24] [25].

Data Presentation: It involves some further processing so that the overwhelming number of received events will be provided in a series of abstractions in order to enable an end-user to draw conclusions about the operation of the monitored system. A

presentation, typically provided by a GUI application making use of visualization techniques, may either use a real-time *stream* of events or a recorded *trace* usually retrieved from an archive. However, in the context of Grids, we generalize the last stage as *consumption* since the users of the monitoring information are not necessarily humans and therefore visualization may not be involved [24].

2.2.3 Grid Monitoring Architecture

The Global Grid Forum [28], Grid Monitoring Architecture Working Group [29] has defined a Grid Monitoring Architecture (GMA) as a recommendation for Grid monitoring system. The GMA [23] is a producer/consumer-based architecture that is driven by a standard and extensible set of events. Figure 2.7 shows the basic GMA.

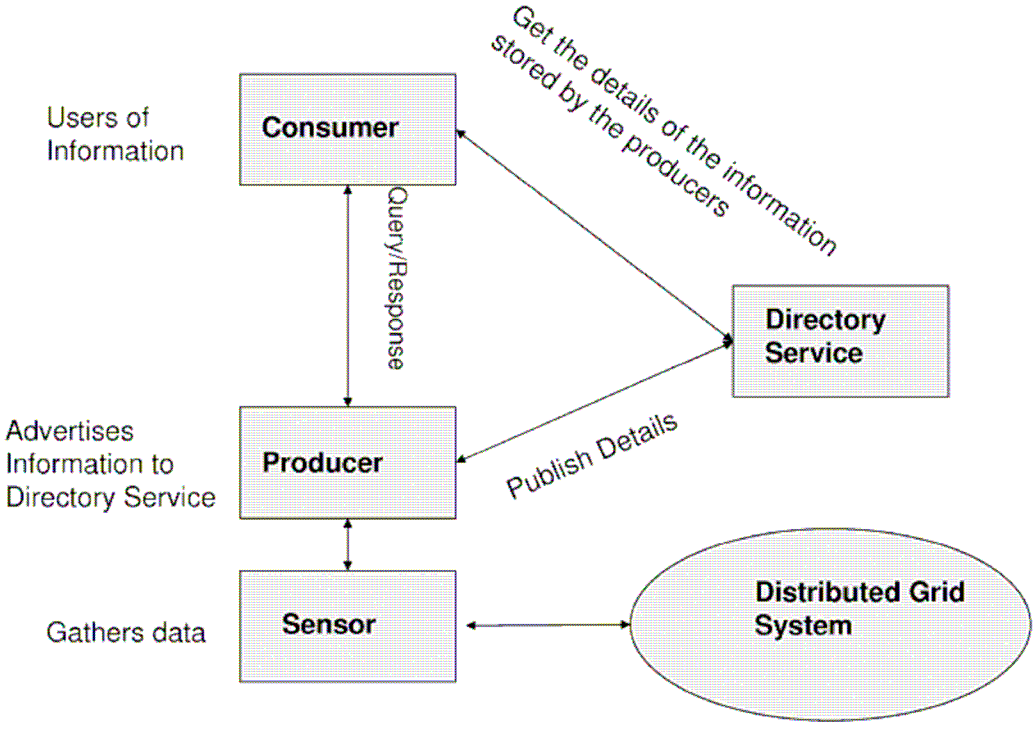


Figure 2.7 Grid Monitoring Architecture (GMA) overview [25]

The different components of the GMA are sensors, producers, consumers, and the directory service.

Sensors: Sensors are the source of the monitoring data and are responsible for gathering data from the distributed Grid system. They are also responsible for generating events from the measured data. The architecture implicitly assumes that sensors are the logical block in the overall GMA structure [23].

Producers: A producer is a software component that sends monitoring data (events) to a consumer. Producers register themselves to the directory service and provide information about the type and structure of information they want to be made available to the Grid. The producers are also some form of sensor managers by making available the amount and type of data that would be useful to the consumers [25] [26].

Producers can deliver events in a stream or as a single response per request. In streaming mode, a virtual connection is established between the producer and consumer and events can be delivered along this connection until an explicit action is taken to terminate it. In query mode, the event is delivered as part of the reply to a consumer-initiated query, or as part of the request in a producer initiated query [26].

Directory Service: The Directory Service supports the publication and discovery of producers, consumers and monitoring data (events). It is responsible for publishing the event type and the corresponding producers. Consumers can contact the service to find out the type of information available and locate the producers who can provide the information [25] [26].

Consumers: Any program that receives monitoring data (events) from a producer can be a consumer. These are the users of the monitoring data. The consumers query the directory service and get the information about the producers. Once this information is obtained, the consumers contact the producers directly to get the information [25].

The GMA was devised as a framework for monitoring Grid systems. However, it goes beyond that and provides a nice framework for combining monitoring and information systems. The model is not constrained by any protocol or data models. Therefore, implementers are free to choose their own data models for querying over the monitoring system.

2.2.4 Challenges in Grid Resource Monitoring

Traditionally resource monitoring systems work under the assumptions that they have complete control on the resource and thus can implement the mechanism and policies related to effective use of that resource. But in Grid, resources are distributed across different geographical locations resulting in heterogeneity so Grid monitoring is characterized by significant requirements including, among others, scalable support for both pull and push data delivery models applied over vast amounts of current and past monitoring data that may be distributed across organizations. In addition, a monitoring system's data format has to balance between extensibility and self-description on one hand and compactness on the other. The former is required to accommodate the ever expanding types of monitored resources, whereas the latter is a prerequisite for non-intrusive and scalable behavior. The problem is further complicated by the continuous evolution of Grid middleware and the lack of consensus regarding data representation, protocols and semantics, leading to *ad hoc* solutions of limited interoperability [24].

The Grid monitoring challenges in general can be listed as follows [30]:

- No single point of observation
- No central point of monitoring information
- Diverse Hardware and Software Systems
- Different policies and decision making mechanisms
- Network monitoring is very important
- Larger monitoring data sets
- Security

2.3 Resource Monitoring in Existing Grid Middleware

Grid is an infrastructure that involves the integrated and collaborative use of computers, networks, databases and scientific instruments owned and managed by multiple organizations. Grid applications often involve large amounts of data and/or computing

resources that require secure resource sharing across organizational boundaries. This makes Grid application management and deployment a complex undertaking. Grid middleware provide users with seamless computing ability and uniform access to resources in the heterogeneous Grid environment. Several software toolkits and systems have been developed, most of which are results of academic research projects, all over the world [31]. This section will focus on three of these middleware— Globus toolkit 4, Alchemi, and Condor. It discusses the resource monitoring in these Grid Middleware in detail.

2.3.1 Globus Toolkit 4

The Globus Toolkit 4 is an open source software toolkit used for building Grid systems and applications. It is being developed by the Globus Alliance and many others all over the world. It is packaged as a set of components that can be used either independently or together to develop applications. The toolkit includes software and libraries for resource monitoring, resource management, data management, communication, fault detection, security and portability [32] [33]. GT4 makes heavy use of *Web services* mechanisms to define its interfaces and structure its components. Web services provide flexible, extensible, and widely adopted XML-based mechanisms for describing, discovering, and invoking network services. GT4 defines Web services interfaces to most (not yet all) of its major components, thus allowing the use of standardized Web services mechanisms to describe GT4 service interfaces and to invoke GT4 service operations. GT4 implements Web services interfaces for management of computational elements and activities executing on those elements (Grid Resource Allocation and Management service, or GRAM), to instrumentation (Grid TeleControl Protocol, or GTCP), and for managing data transfers (Reliable File Transfer service, or RFT). It also provides GridFTP, a data transfer service for which no Web service interface has yet been defined. These various components enable reliable, secure, and managed interactions with individual resources, and thus provide a basis for building larger infrastructures [33]. Our focus here is on resource monitoring in Globus toolkit 4.

MDS4

The Monitoring and Discovery System (MDS4) [34] of the Globus Toolkit 4 is a suite of components for monitoring and discovering Grid resources and services. MDS4 is compliant with WSRF and WS-Notification specifications. The Globus Toolkit 4's Monitoring and Discovery System (MDS4) implements a standard Web Services interface to a variety of local monitoring tools and other information sources.

MDS4 is a "protocol hourglass," defining standard protocols for information access and delivery and standard schemas for information representation. Below the neck of the hourglass, MDS4 interfaces to different local information sources, translating their diverse schemas into appropriate XML schema (based on standards such as the GLUE schema whenever possible). Above the neck of the hourglass, various tools and applications can be constructed that take advantage of the uniform Web Services query, subscription, and notification interfaces to those information source that MDS4 implements [35] [36].

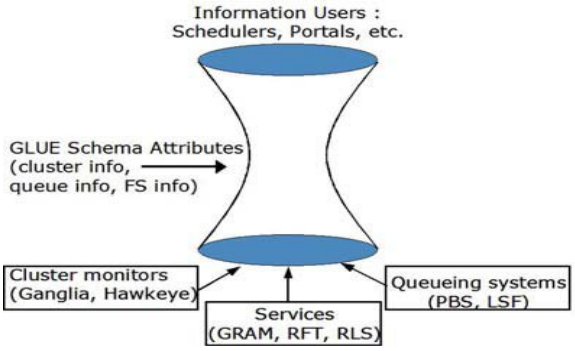


Figure 2.8 The MDS4 hourglass provides a uniform query, subscription and notification interface to a wide variety of information sources, web services, and other monitoring tools [35].

Both monitoring and discovery applications require the ability to collect information from multiple, perhaps distributed, information sources. To meet this need, MDS4 provides so-called *aggregator services* that collect recent state information from registered *information sources*; and browser-based interfaces, command line tools, and Web service interfaces that allow users to query and access the collected information. MDS4 provides three different aggregator services with different interfaces and behaviors

(although all built on a common framework): *MDS-Index*, which supports Xpath queries on the latest values obtained from the information sources; *MDS-Trigger*, which performs user-specified actions (such as send email or generate a log-file entry) whenever collected information matches user determined criteria; and *MDS-Archiver*, which stores information source values in a persistent database that a client can then query for historical information [33].

MDS4 makes heavy use of XML and Web service interfaces to simplify the tasks of registering information sources and locating and accessing information of interest. In particular, all information collected by aggregator services is maintained as XML, and can be queried via Xpath queries (as well as other Web services mechanisms).

MDS provides access to static and dynamic information of resources. Basically, it contains the following components [3] as shown in figure 2.9:

- Information Providers
- Grid Resource Information Service(GRIS)
- Grid Index Information Service (GIIS)
- MDS Client

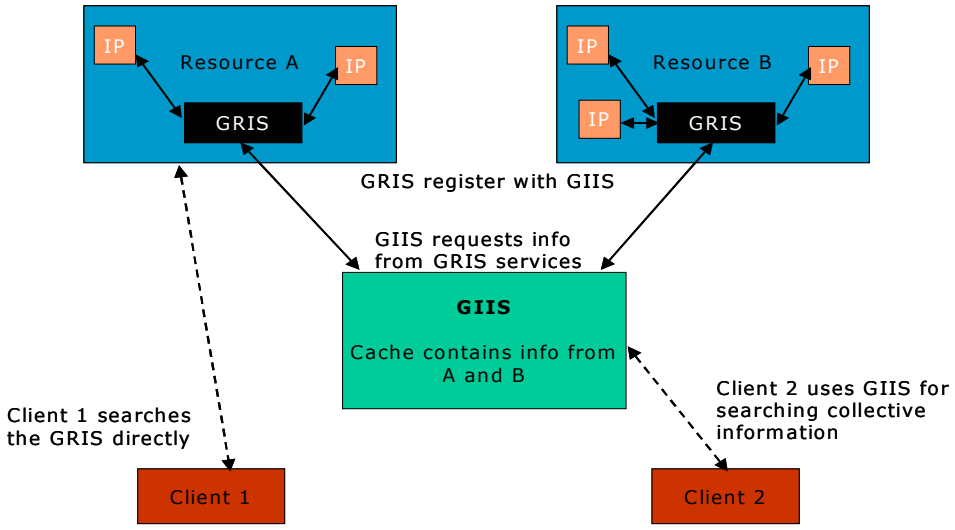


Figure 2.9 MDS overview [35]

The resource information is obtained by the information provider and it is passed to GRIS. GRIS registers its local information with the GIIS, which also registers with another GIIS, and so on. MDS clients can get the resource information directly from GRIS (for local resources) and/or a GIIS (for Grid-wide resources).

Grid Resource Information Service (GRIS)

GRIS is the repository of local resource information derived from information providers. GRIS is able to register its information with a GIIS, but GRIS itself does not receive registration requests. The local information maintained by GRIS is updated when requested, and cached for a period of time known as the time-to-live (TTL). If no request for the information is received by GRIS, the information will time out and be deleted. If a later request for the information is received, GRIS will call the relevant information provider(s) to retrieve the latest information [3] [33].

Grid Index Information Service (GIIS)

GIIS is the repository that contains indexes of resource information registered by the GRIS and other GIISs. It can be seen as a Grid wide information server. GIIS has a hierarchical mechanism, like DNS, and each GIIS has its own name. This means client users can specify the name of a GIIS node to search for information [3] [30].

Information providers

The information providers translate the properties and status of local resources to the format defined in the schema and configuration files. In order to add your own resource to be used by MDS, you must create specific information providers to translate the properties and status to GRIS.

MDS client

The MDS client is based on the LDAP client command, **ldapsearch**. A search for a resource information that you want in your Grid environment is initially performed by the MDS client [3].

WebMDS is a web-based interface to WSRF resource property information that can be used as a user-friendly front-end to the index service. *It* provides a simple XSLT-transform based visual interface to the data [35].

Hierarchical MDS

The MDS hierarchy mechanism is similar to the one used in DNS. GRIS and GIIS, at lower layers of the hierarchy, register with the GIIS at upper layers. Clients can query the GIIS for any information about resources that build a Grid environment [30].

2.3.2 Alchemi.Net

Though scientific computing facilities have been heavy users of Unix-class OSes, the vast majority of computing infrastructure within enterprises is still based on Microsoft Windows. Alchemi was developed to address the need within enterprises for a desktop Grid solution that utilizes the unused computational capacity represented by the vast number of PCs and workstation running Windows within an organization [39] [40]. Alchemi [37] [38], a .NET-based Grid computing framework is an open source software framework that allows you to painlessly aggregate the computing power of networked machines into a virtual supercomputer and to develop applications to run on the Grid. It also provides an object-oriented programming model along with web service interfaces that enable its services to be accessed from any programming environment that supports SOAP-XML abstraction [39]. It has been designed with the primary goal of being easy to use without sacrificing power and flexibility [37].

Alchemi includes:

- The runtime machinery (Windows executables) to construct computational Grids.
- A .NET API and tools to develop .NET Grid applications and Grid-enable legacy applications.

There are four types of distributed components (nodes) involved in the construction of Alchemi Grids and execution of Grid applications [40]:

- Manager
- Executor
- User
- Cross-Platform Manager.

As far as resource monitoring in Alchemi is concerned, Heartbeat is available in Alchemi that informs the Manager about the resource characteristics and status.

2.3.3 Condor

Condor [41] [42] is developed by the Condor Team at the University of Wisconsin-Madison (UW-Madison), and was first installed as a production system in the UW-Madison Computer Sciences department. Condor is a high-throughput computing environment that can manage a large collection of diversely owned machines and networks; Condor is a specialized job and *resource management system* (RMS) [43] for compute intensive jobs. Like other full-featured systems, Condor provides a job management mechanism, scheduling policy, priority scheme, resource monitoring, and resource management [44]. Users submit their jobs to Condor, and Condor subsequently chooses when and where to run them based upon a policy, monitors their progress, and ultimately informs the user upon completion. Condor uses Hawkeye [45] to monitor characteristics and status of resources.

Hawkeye

Hawkeye [45] is another distributed monitoring system, coming from the University of Wisconsin Madison. It provides mechanisms for monitoring distributed collections of computers by gathering computer-based resource information. Hawkeye's design goals include retrieval of host resource information in a consistent and extensible manner and the ability to automatically execute tasks in response to observed conditions on the monitored hosts. Resource information, it is packaged as a self-contained system. Data

collected by Hawkeye is made available for applications and users to manage monitored resources [25] [26].

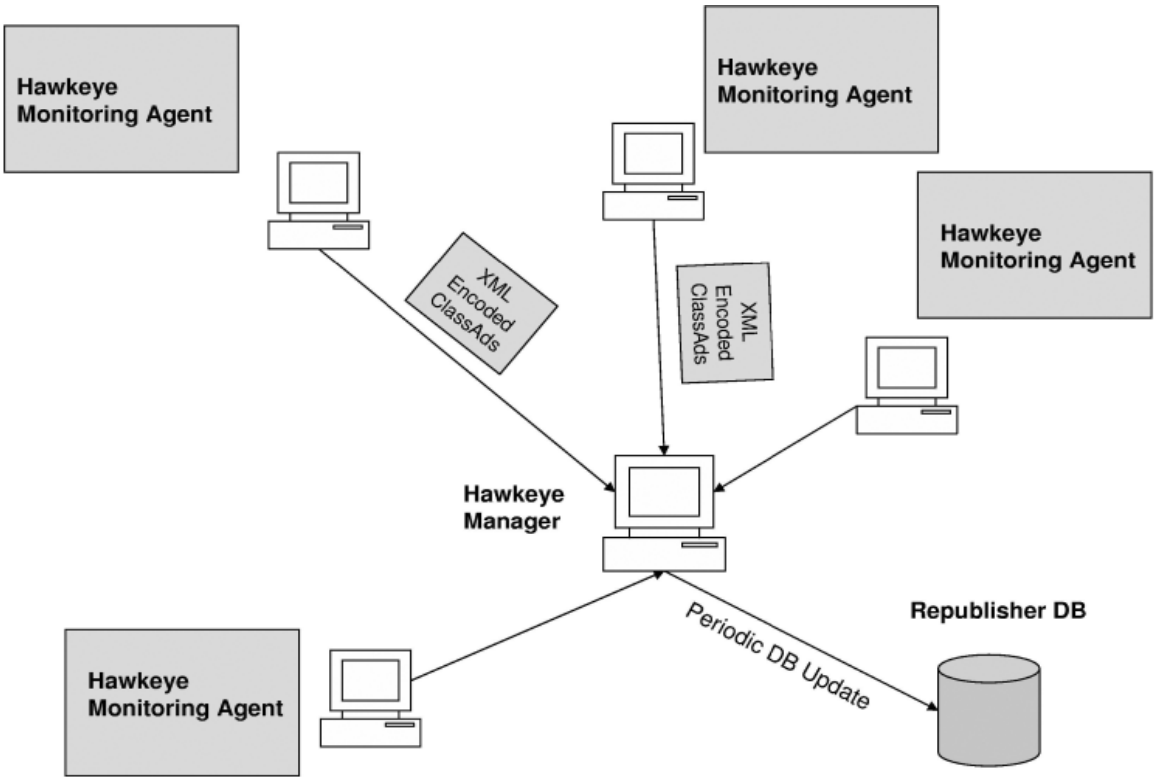


Figure 2.10 Overview of the Hawkeye monitoring system [25]

Each node that is to be monitored hosts a Hawkeye monitoring agent (HMA) which periodically calculates the different resource information metrics that are measured and return them in the form of Condor ClassAd attribute–value pairs. The monitoring agent is responsible for consolidating information from multiple modules into a single ClassAd that describes the combined status of a local resource. Periodically, the HMA pushes this combined information to a remote Hawkeye Manager (HM). The manager stores the information periodically in a round robin database. The information can be queried through simple interfaces [22] [25] [26].

2.4 Other Resource Monitoring Tools

In the previous section resource monitoring systems available in various Grid middleware have been discussed. This section discusses the two important Grid Resource monitoring tools that are used commercially in various Grid applications.

2.4.1 Ganglia

Ganglia [46] is an open-source project that grew out of the **University of California, Berkeley.** Ganglia is a distributed monitoring system for high performance computing systems such as clusters and the Grid.

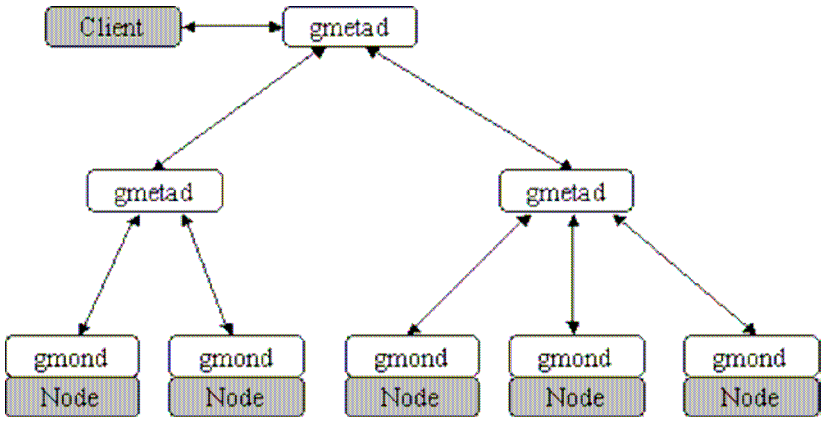


Figure 2.11 the architecture of Ganglia [26]

It is based on a hierarchical design targeted at federations of clusters. Ganglia use XML for data representation, XDR [48] for data transport, and RRDtool [49] for data storage and visualization. Ganglia attempts to minimize per-node system overheads by using specialized data structures and algorithms [46] [47]. Figure 2.11 shows the architecture of Ganglia.

Gmond: The Ganglia Monitoring Daemon (gmond) is a multi-threaded daemon, which runs on each cluster node to be monitored; it has four main responsibilities:

• monitor changes in host state;

• multicast relevant changes;

• listen to the state of all other Ganglia nodes via a multicast channel;

• answer requests for an XML description of the cluster state.

Each daemon transmits information in two different ways: multicasting host state in XDR format or sending XML over a TCP connection [25] [26].

Gmetad: Ganglia Meta Daemons (gmetad) are used to provide a federated view. At each node in the tree, a *gmetad* periodically polls a collection of child data sources, parses the collected XML, saves all numeric, volatile metrics to round-robin databases, and exports the aggregated XML over a TCP socket to clients. Data sources may be either *gmond* daemons, representing specific clusters or other *gmetad* daemons, representing sets of clusters. Data sources use source IP addresses for access control and can be specified using multiple IP addresses for failure. The latter capability is natural for aggregating data from clusters since each *gmond* daemon contains the entire state of its cluster [25] [26].

PHP Webfront End: Ganglia PHP Web User Interface provides a view of the gathered information via real-time dynamic Web pages. Most importantly, it displays Ganglia data in a meaningful way for system administrators and computer users. Although the Web UI to Ganglia started as a simple HTML view of the XML tree, it has evolved into a system that keeps a colorful history of all collected data. The UI depends on the existence of the *gmetad*, which provides it with data from several Ganglia sources. Specifically, the Web UI will open the local port 8651 (by default) and expects to receive a Ganglia XML tree. The Web pages themselves are dynamic; any change to the Ganglia data appears immediately on the site. This behavior leads to a very responsive site, but requires that the full XML tree be parsed on every page access. Therefore, the Ganglia UI should run on a dedicated machine if it presents a large amount of data. The UI is written in the PHP

scripting language, and uses graphs generated by *gmetad* to display historical information [26].

2.4.2 R-GMA

The Relational Grid Monitoring Architecture (R-GMA) [30] monitoring system is an implementation of the Grid Monitoring Architecture (GMA) [25] defined within the Global Grid Forum (GGF). It is based on the relational data model and Java servlet technologies. Its main use is notification of events that is, a user can subscribe to a flow of data with specific properties directly from a data source. For example, a user can subscribe to a load-data data stream and create a new producer/consumer pairing to allow notification when the load reaches some maximum or minimum [22] [50].
Figure 2.12 shows the high level overview of the RGMA architecture.

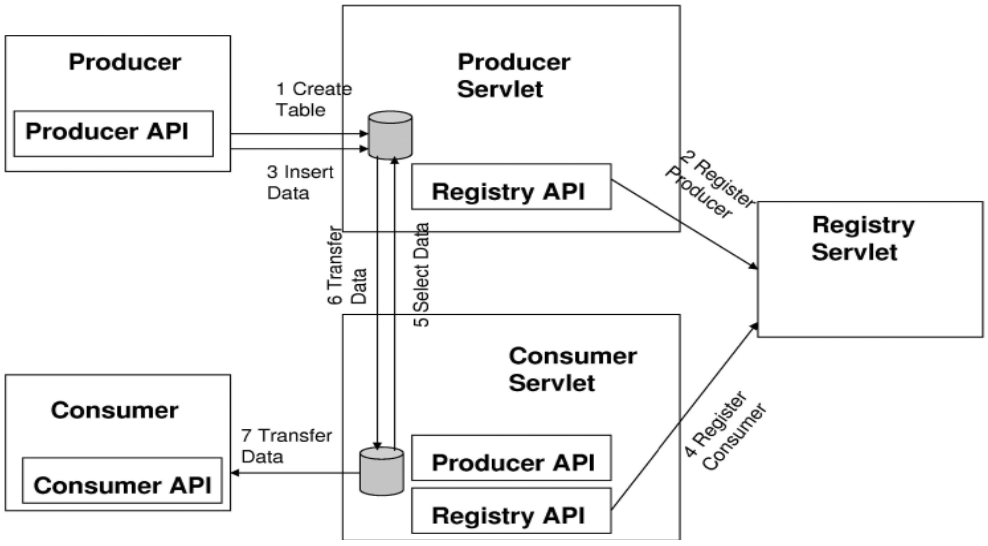


Figure 2.12 RGMA Architecture [25]

Currently, a standard R-GMA deployment includes information gathering by interfacing to the MDS information providers, including basic host data, queue information, and some network data. R-GMA uses the SQL mapping of the GLUE schema, and users can add additional information providers by simply defining table entries and collectors [22].

The R-GMA consists of a registry, or directory service, and a set of producers and producer servlets that advertise tables and definitions of the data held in the RDBMS in the Registry. The RDBMS holds the information for all the producers, like, the registered table name, the predicate etc. Users or other agents can contact a consumer servlet that issues SQL queries against a set of supported tables, to find suitable producers and then contact the producers directly to query for the data. R-GMA producers can be quired using a command line tool, a Java graphical display tool, or the R-GMA browser [22] [51].

In addition to the Resource Monitoring Tools discussed above there is lot many Resource Monitoring Tools like Netlogger, Monlisa, Jinni, Inca, HadleSystem GT project are available for Grid environment.

2.5 Gap Analysis

Above comparison shows that no Resource Monitoring system is perfect. Each is having certain limitations; So there is need to develop a monitoring system for Grid environment that overcome the limitations of existing resource monitoring systems.

2.6 Problem Formulation

Grid computing enables sharing, selection and aggregation of large collections of geographically and organizationally distributed heterogeneous resources for solving large-scale data and compute intensive problems. Resources are dynamic in nature, resource availability is not sure at any point in time and it makes resource monitoring more important in case of Grid environment. To improve the global throughput of these environments, effective and efficient Resource Monitoring approach is fundamentally important. At present, there exist two main approaches in Grid resource monitoring: MDS (Monitoring and Directory Service) and GMA (Grid Monitoring Architecture). MDS is used to find, describe and monitor resources, services etc. It is composed of GRIS (Grid Resource Information Service), GIIS (Grid Index Information Service), GRIP (Grid Information Protocol), and GRRP (Grid Registration Protocol). The MDS

has some disadvantages. GRIS cannot provide comprehensive description of resource information. LDAP cannot support large amounts of data objects effectively, and it cannot support update and complex query, too. The layered and centralized mechanism in MDS brings on a rapid growth of data traffic in monitoring services provided by top nodes, and results in a longer response time of monitoring information. There are only three components in the GMA: producer, consumer, and directory services. The basic unit of monitoring data in GMA is an event. The GMA supports subscription and the request/response model. The monitoring data in GMA can be migrated directly from producer to consumer. The problems in GMA are: There is a lack of metadata definition of monitoring information; when a new resource is added, new suitable producer and consumer must be developed. The GMA don't support the third-party monitoring services and QoS (Quality of Services).

The focus of our study is to consider factors which can be used as characteristics for decision making to initiate Resource Monitoring. Based upon above findings, Agent-Based Resource Monitoring is the solution for efficient monitoring in Grid environment.

This thesis work analyzes the existing Resource Monitoring Systems and tries to find out performance bottlenecks in it. As a result the following things have been addressed in this thesis:

• Study of existing Resource Monitoring systems for a Grid environment

• An Agent-Based Resource Monitoring Approach has been proposed, designed and implemented the Grid environment.

Chapter 3

Proposed Agent-Based Grid Resource Monitoring Approach

This chapter is going to discuss in detail the Proposed Agent-Based Grid Resource Monitoring System, its Architecture and components.

3.1` Resource Monitor Architecture

Agent based Resource Monitoring Architecture identifies the basic components of the system. The monitor architecture defines the purpose and functions of its components, while indicating how these components interact with one another. The component layers are listed below:

- Client Layer
- Server Layer
- Resource Monitoring Layer

Each layer shares the behavior of the underlying component layers. The following describes the core features of each of these component layers, starting from the bottom of the stack and moving upward.

3.2 Components of Agent based Resource Monitoring System

Agent based monitoring System consists of the following basic components that co ordinate together to perform Grid resource monitoring.

Next chapter will discuss implementation details for the proposed Agent Based Grid Resource Monitoring system and experimental results.

Chapter 4

Implementation Details and Experimental Results

This chapter describes the implementation details including setting up of the Grid environment using various middleware, installation and integration of proposed Agent Based Grid Resource Monitoring System and Experimental Results.

4.1 Implementation Details

This Section discusses the implementation of proposed Agent based Grid Resource Monitoring System in detail. Monitoring System is based upon client server architecture and is implemented in java using Netbeans IDE 1.6.0. Java language offers several features that facilitate with easiness the development and deployment of a software environment for Grid computing. Java's network based features are very useful to develop Grid application. Java runtime systems are available for most hardware platforms and operating systems. Because of the heterogeneity of the hardware and of operating systems employed by Internet users, it is crucial that a platform for large-scale Grid computing be available for a large variety of different computer systems. Consequently, a Java- based platform potentially allows every computer in the Internet to be exploited for distributed, large- scale computations, while at the same time the maintenance costs for the platform are minimal. Apart from its portability and compatibility, language safety and a sophisticated security model with flexible access control are further frequently cited advantages of Java.

4.1.1 Installing and Integrating Agent based Grid Resource Monitoring System

This section discusses the installation and integration of Agent Based Grid Resource Monitoring System with Globus toolkit 4 on Fedora 6 platform.

1. Installation of RRDtools

The RRDtools package is required on every client machine running the GMETAD since it supports the functions of the GMETAD by providing a way to efficiently store the data collected. Install Client RRDtools on each client machine running Globus toolkit 4 on fedora 6 platform:

- Download the developer package "rrdtool-devel-1.0.38-1.i386.rpm" from the downloads section of the RRDtools web page. i.e.: (http://www.rrdtool.com/download.html)
- Install the developer package in the location decided by the system administrator.
- Download the regular release "rrdtool-1.0.38-1.i386.rpm" from the RRDtools web page http://www.rrdtool.com/download.html
- Install the regular package in the location decided by the system administrator.

This completes the RRD tool installation. On completion, please make sure that the following two files are present in appropriate locations

• librrd.a in /usr/lib/librrd,a

• rrd.h in /usr/include/rrd.h

These two files are important for data storage and their absence will result in no data being registered in the main machine.

2. Installation of Client agent

Install Client agent Gmond on each client machine running Globus toolkit 4 on fedora 6 platform:

- Download the package " ganglia-gmetad-3.0.7-1.i386.rpm" from

  http://ganglia.sourceforge.net/downloads.php
- Install the package in the location decided by the system administrator.
- Start the service gmond

[root@goel ~] # service gmond start

1. Integration of Client agent with Grid middleware

Monitoring System needs to be integrated with Grid middleware, Globus toolkit 4 here in order to get the resource info. To integrate client agent with Globus go to the directory: Globus_LOCATION$/etc/globus_wsrf_mds_usefulrp and change the entry in file gluerp.xml.

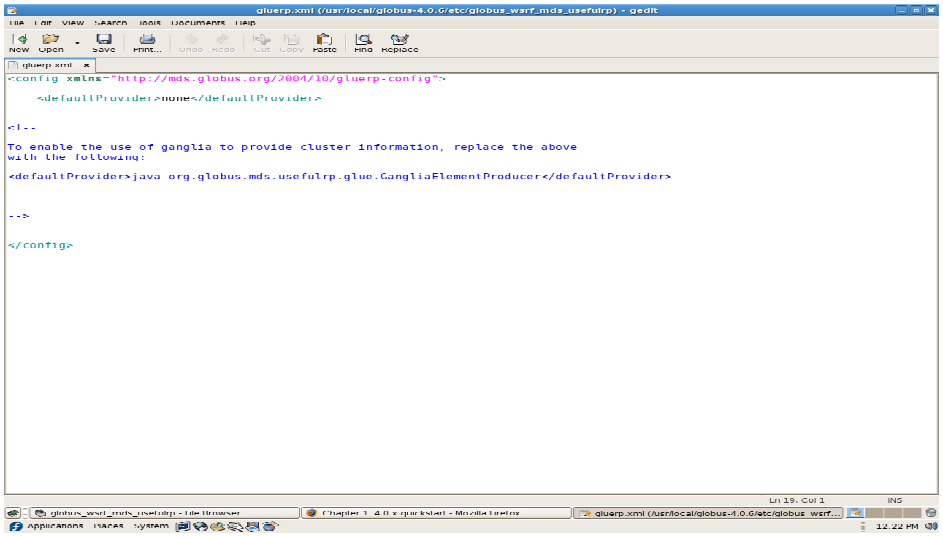


Figure 4.1 gluerp.xml

2. Running Server Agent

Run Server Agent updatefile on server machine running globus toolkit 4 on fedora 6 platform.

[root@goel ~] # gcc -o updatefile.out updatefile.c

[root@goel ~] # ./updatefile.out

3. Running Resource Monitoring

Run resource monitoring module, A-RMS on the machine running on window platform where central database repository is created. A-RMS should run on window platform as database repository is in MS access which is not sported by fedora.

4.2 Experimental Results

This section discusses the experimental results obtained when proposed Agent-Based Resource Monitoring System Monitors the Grid environment.

The very first screen that appear to Grid administrator is the administrator interface to Agent-based Resource Monitoring System as shown in Figure 4.2

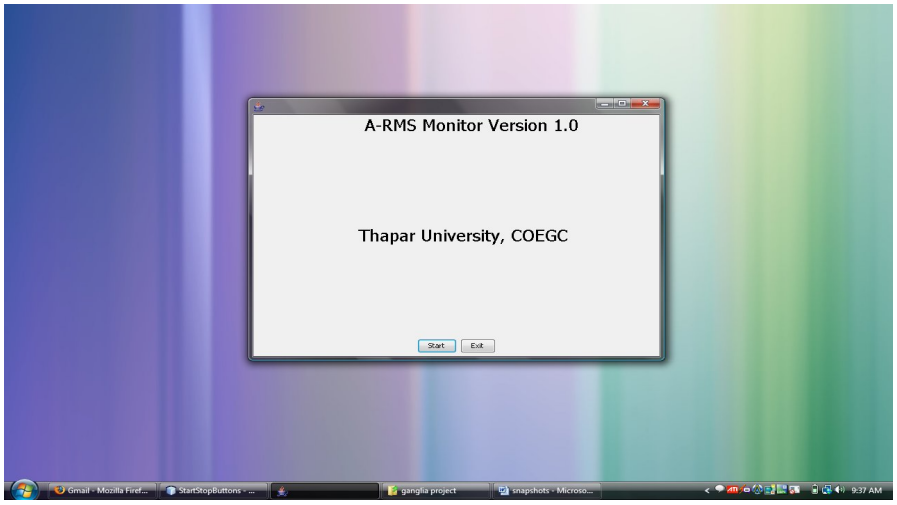


Figure 4.2 Administartor Interface

As shown in the Figure 4.2, there are 2 radio buttons Start and Exit for selection of action to be performed by the Grid administrator. By clicking the start button Resource Monitoring is started. Whenever start button is pressed, stop button appears in place of start button. Resource monitoring can be stopped at any moment by pressing stop button. Exit button is to exit the Agent-Based Resource Monitoring System.

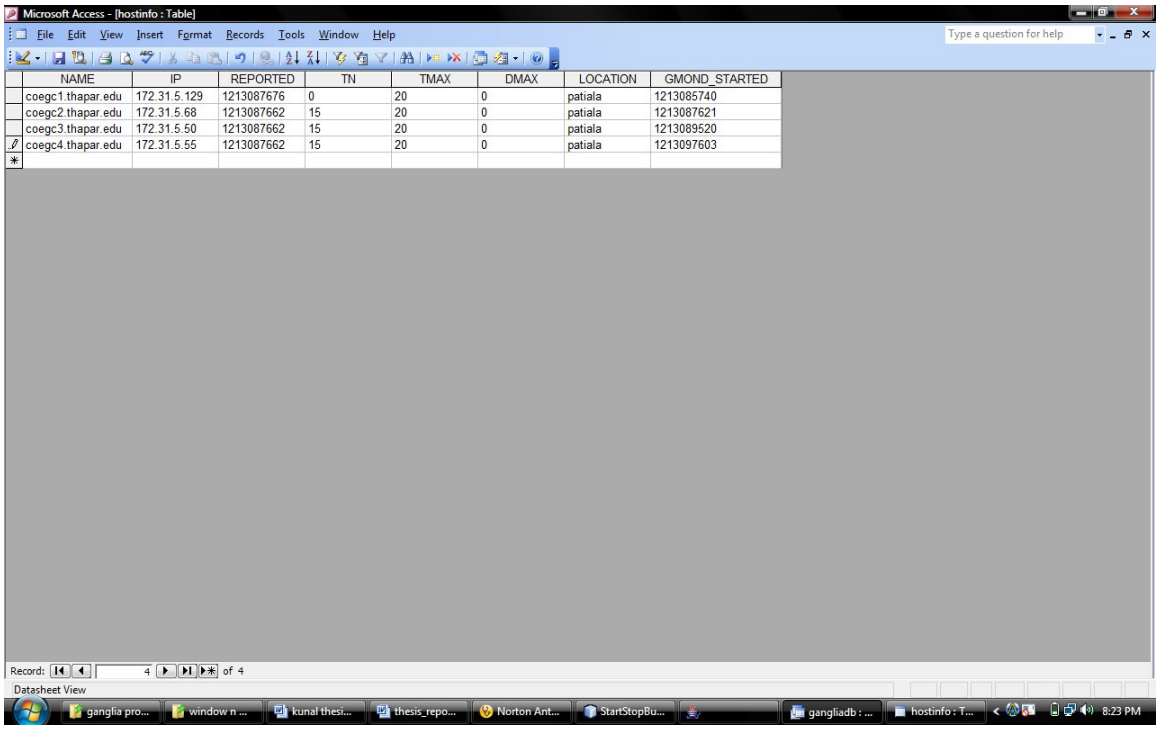


Figure 4.3 Hostinfo

Figure 4.3 shows the Hostinfo, a part of central database repository, it stores the name and ipaddress of the nodes connected to Grid environment. When start button is pressed Hostinfo is updated periodically and whenever a node is added to Grid environment it's name and ipaddress is put into the Hostinfo. Other part of central resource repository is shown in the Figure 4.4.

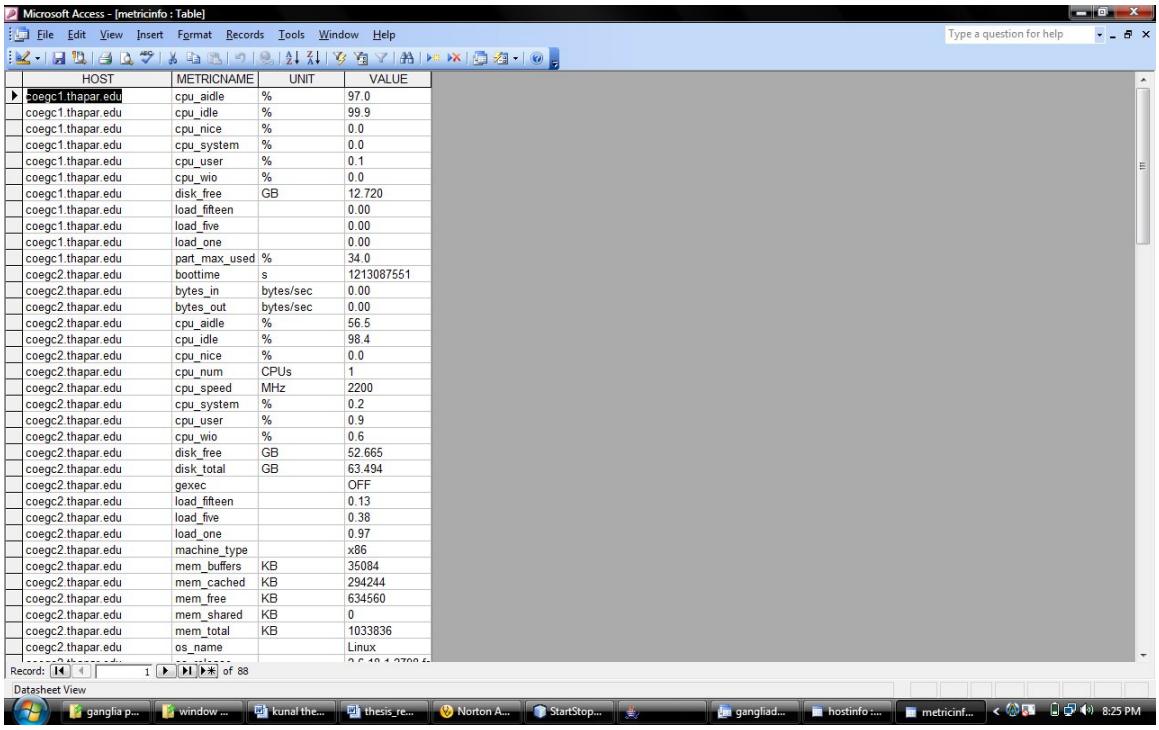


Figure 4.4 Metricinfo

Metricinfo is the most important part of central database repository. It stores the detailed information about different performance parameters of each node in Grid environment. When start button is pressed Metric info is also updated periodically along with Hostinfo and whenever a new node is added to Grid it's parameters are also updated in Metricinfo.

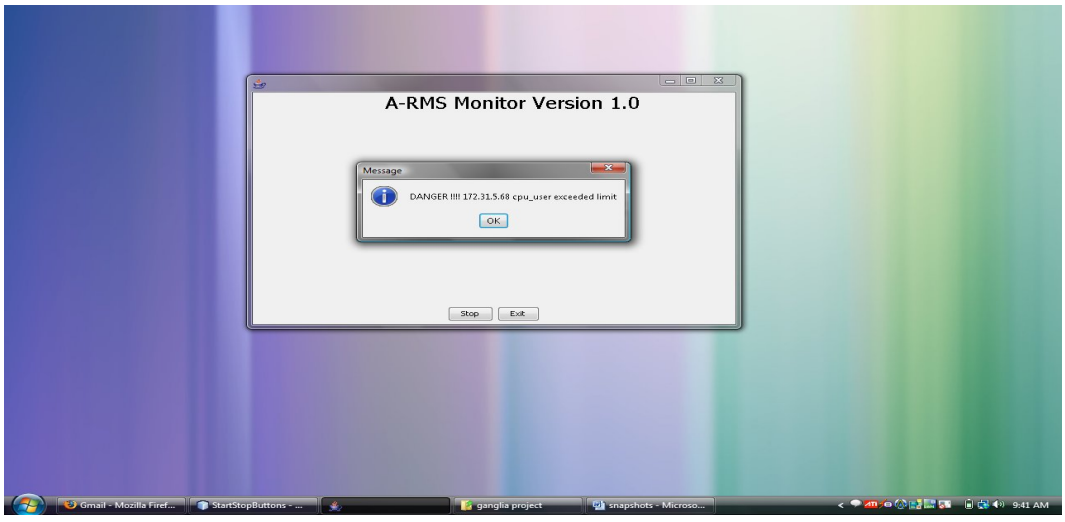


Figure 4.5 cpu_user exceeded

Figure 4.5 shows one of the node 172.31.5.68 performance parameter cpu_user exceeding its normal limit. This information is provided by the resource monitoring module to Grid administrator. It is due to the excessive use of this node, some heavy job is run on this node. In order to bring the cpu_user of node 172.31.5.68 to normal value Job running on this node need to be migrated to another node, this decision will be taken by the Grid administrator.

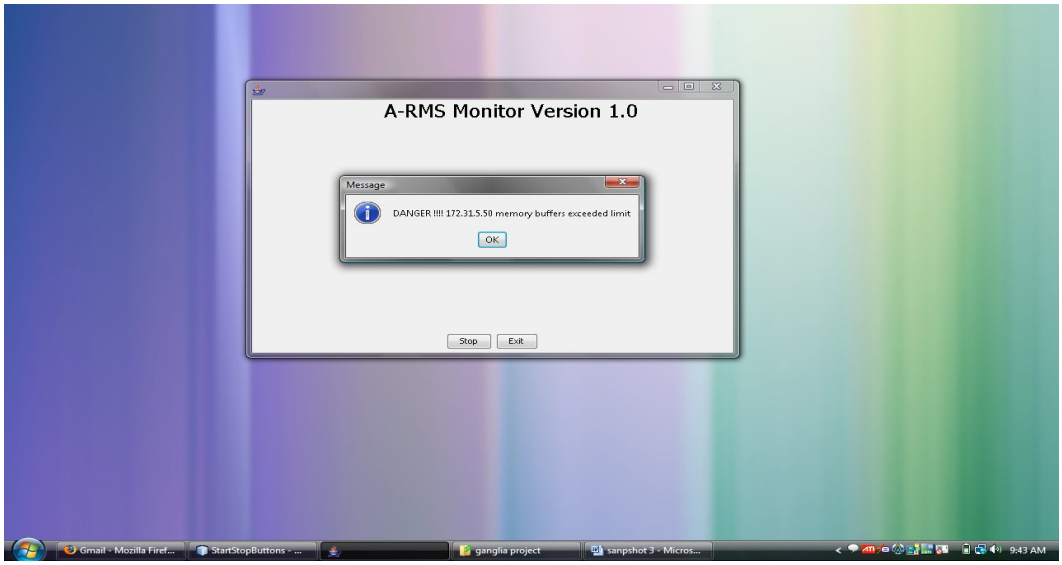


Figure 4.6 memory buffers exceeded

Figure 4.6 shows another node 172.31.5.50 performance parameter memory buffers exceeding its normal limit. It's due to over usage of memory at this node. Again the problem can be solved by migrating job to some other node. Action will be taken by Grid administrator.

This chapter has discussed the implementation details including setting up of the Grid environment using various middleware, installation and integration of proposed Agent-Based Resource Monitoring System and Experimental Results. Next Chapter concludes the whole thesis and proposes future work that can be done to enhance the monitoring system designed.

Chapter 5

Conclusions and Future Scope of Work

This chapter discusses the conclusions of the work presented in this thesis. The chapter ends with a discussion of the future direction which this work will take.

5.1 Conclusions

The work presented in this Thesis provides an insight into the world of Grid Computing, the current state of the art in Grid Resource Monitoring along with pros and cons of each approach.

Through this thesis, we have described multiple aspects of Grid Computing and introduced numerous concepts which illustrate its broad capabilities. Grid Computing is definitely a promising tendency to solve high demanding applications and all kinds of problems. Objective of the Grid environment is to achieve high performance computing by optimal usage of geographically distributed and heterogeneous resources.

But Grid application performance remains a challenge in dynamic Grid environment. There are a number of factors, which can affect the Grid application performance like Resource Monitoring, heterogeneity of resources and resource sharing in the Grid environment, Resource Discovery. Resources can be added to Grid and can be withdrawn from Grid at any moment. This characteristic of Grid makes resource monitoring one of the critical features of Grid infrastructure.

In this thesis we have focused on Resource Monitoring and tried to present the impacts of Resource Monitoring on Grid application performance and finally proposed an efficient Agent based Resource Monitoring approach for Grid environment. Agent Based Resource Monitoring system is then implemented and successfully installed and integrated with Globus toolkit 4 in test Grid environment. It overcomes the problems found in existing Resource Monitoring approaches, important features of proposed Agent based Resource Monitoring System are:

- It is very flexible. New resources can be added easily. Just Client Agent (gmond) need to be installed and run on added client machine.
- It has layered and distributed mechanism and hence fast response time of monitoring information.
- It can support large amount of data objects, update and other complex queries.
- It supports third party monitoring service and QoS.
- It provides information about the resource status to Grid administrator in an easy to read format (pop up messages).

5.2 Future Scope of Work

Proposed Agent Bases Grid Resource Monitoring System is installed and integrated with Globus toolkit 4 on fedora 6 platform. As it is developed in java so it can be installed on any OS platform like Window, Solaris but problem is in integrating it with the different Grid middleware like Alchemi.Net, Condor, Sun N1 GE6. In future work can be done to integrate the proposed monitoring system with these Grid Middleware so that a complete heterogeneous, distributed, dynamic Grid environment will be monitored using this Agent Based Grid Resource Monitoring System.

References

[1] A.S. Rana, PhD Thesis, "A Globally Distributed Grid Monitoring System to Facilitate High Performance Computing at DE/SAM-GRID (Design, Development, Implementation and Development of a Prototype)", The University of Texas, Arlington, December 2002.

[2] Ian Foster, Carl Kesselman, and Steve Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of High Performance Computing Applications, 15 (3), pages:200-222, 2001. http://www.globus.org/research/papers/anatomy.pdf

[3] C. Kesselman. I. Foster, " Computational grids", in The Grid 2: Blueprint for a New Computing Infrastructure, Elsevier, San Francisco, ISBN no: 1-55860-933-4, 2004.

[4] Ian Foster, "What is the Grid? A Three Point Checklist," Argonne National Laboratory & University of Chicago.

[5] Ferreira, L., Bieberstein, N., Berstis, V., Armstrong, J., "Introduction To Grid Computing With Globus", IBM Redbook, IBM Corp http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf

[6] Fran Berman, Geoffrey Fox, Anthony J.G. Heg, "Grid Computing: Making the Global Infrastructure a Reality", Wiley Series in communications, networking and distributed systems, ISBN no: 0-470-85319-0, 2003.

[7] Introduction to grid computing. http://www.lip.pt/computing/projects/EGEE/public_gridintro_en.htm

[8] Grid information available at http://www-03.ibm.com/grid/

[9] Luciano Tarricone and Alexandra Esposito, "Grid Computing for Electromagnetics", Artech House, Boston, London, ISBN No:1-58053-777-4, 2004.

[10] History of Grid available at http://gridcafe.web.cern.ch/gridcafe/whatisgrid/dream/powergrid.html

[11] Miguel L. Bote-Lorenzo, Yannis A. Dimitriadis, and Eduardo G´omez-S´anchez, "Grid Characteristics and Uses: a Grid Definition", School of Telecommunications Engineering, University of Valladolid Camino Viejo del Cementerio s/n, 47011 Valladolid, Spain {migbot, yannis, edugom}@tel.uva.es

[12] Zoltan Juhasz, Petwr Kacsuk, Dieter Kranzlmuller, "Distributed and Parallel Systems", Springer, eBook ISBN No:0-387-23096-3, 2005.

[13] Ferreira, L., Bieberstein, N., Berstis, V., Armstrong, J., "Introduction To Grid Computing with Globus", IBM Redbook, IBM Corp http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf

[14] Rajkumar Buyya and S Venugopal, "A Gentle Introduction to Grid Computing and Technologies", http://www.buyya.com/papers/GridIntroCSI2005.pdf.

[15] I Foster, C Kesselman, J M Nick and S Tuecke, "Grid services for distributed system integration," Computer, vol. 35, no. 6, pp. 37–46, June 2002. http://www.globus.org/alliance/publications/papers/ieee-cs-2.pdf

[16] J. Joseph, M. Ernest, C. Fellenstein, "Evolution of grid computing architecture and grid adoption models", IBM Systems Journal, Volume 43, Issue 4 (2004), pages: 624-645.

[17] Klaus Krauter, Rajkumar Buyya and Muthucumaru Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," SOFTWARE—PRACTICE AND EXPERIENCE, Softw. Pract. Exper. 2002; 32:135–164 (DOI: 10.1002/spe.432)

[18] R Buyya, D Abramson, J Giddy, "Nimrod/G: An architecture for a resource management and scheduling system in a global computational Grid", Proceedings of the International Conference on High Performance Computing in Asia–Pacific Region (HPC Asia 2000), 2000.

[19] W Hoscheck, J Jaen-Martinez, A Samar, H Stockinger, K Stockinger, "Data management in an international data Grid project", Proceedings 1st IEEE/ACM International Workshop on Grid Computing (Grid 2000), December 2000, pages:77–90.

[20] A Chervenak, I Foster, C Kesselman, C Salisbury, S Tuecke, "The data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets", Journal of Network and Computer Applications 23(3), pages:187–200.

[21] K Nahrstedt, H Chu, S Narayan, "QoS-aware resource management for distributed multimedia", Journal of High-Speed Networking (Special Issue on Multimedia Networking) December 2000; 7(3–4), pages:227–255.

[22] Jennifer M. Schopf and Ben Clifford, "Monitoring Clusters and Grids", Cluster World Magazine, May 2004.

[23] Z. Balaton, P. Kacsuk, N. Podhorszki, F. Vajda, "Use Cases and the Proposed Grid Monitoring Architecture", Tech. Rep. LDS-1/2001, Computer and Automation Research Institute of the Hungarian Academy of Sciences, 2001, available at http://www.lpds.sztaki.hu/publications/reports/lpds-1-2001.pdf, accessed on 13th July, 2006.

[24] Serafeim Zanikolas, Rizos Sakellariou, "Taxonomy of grid monitoring systems", Future Generation Computer Systems, 21(1), January 2005, pages:163-188.

[25] Aniraban Chakarabarti, "Grid Computing Security", Springer, ISBN No: 978-3-540-44492-3, 2007.

[26] Maozhen li, Mark Baker, "The Grid Core Technologies", John Wiley and Sons Ltd. ISBN No: 13 978-0-470-09417-4, 2005.

[27] Jarek Nabrzyski, Jennifer M. Schopf, Jan We¸Glarz, "Grid Resource Management State of the Art and Future Trends", Kluwer Academic Publishers Boston/Dordrecht/London, ISBN: 1-4020-7575-8, 2003.

[28] Global Grid Forum. http://www.gridforum.org

[29] Global Grid Forum, Grid Monitoring Architecture Working Group. http://www.didc.lbl.gov/GGF-PERF/GMA-WG/

[30] Stratos Efstathiadis, "Introduction to Grid Monitoring.ppt",BNL, ITD – STAR PPDG.

[31] Parvin Asadzadeh, Rajkumar Buyya, Chun Ling Kei, Deepa Nayar, and Srikumar Venugopal, "Global Grids and Software Toolkits: A Study of Four Grid

Middleware Technologies", Technical Report, GRIDS-TR-2004-4, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, July 1, 2004.

[32] Globus Toolkit 4. http://www.globus.org/toolkit/about.html

[33] A Globus Primer 4 available at http://www.globus.org/toolkit/docs/4.0/key/

[34] MDS Details available at http://www.globus.org/toolkit/mds

[35] MDS4: The GT4 Monitoring and Discovery System. Contact email: jms@mcs.anl.gov http://www.globus.org/toolkit/docs/4.0/info

[36] MDS4 and Project Deployments. July 28, 2005 Contact: mds-internal@mcs.anl.gov

[37] Alchemi.Net details are available at http://alchemi.net/

[38] Alchemi.Net Manual available at http://www.gridbus.org/~alchemi/files/1.0.beta/docs/AlchemiManualv.1.0.htm

[39] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, Srikumar Venugopal, "A .NET-based Grid Computing Framework and its Integration into Global Grids", Technical Report, GRIDS-TR-2003-*, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, December 2003.

[40] Akshay Luther, Rajkumar Buyya, Rajiv Ranjan, and Srikumar Venugopal, "Alchemi: A .NET-based Grid Computing Framework and its Integration into Global Grids", International Conference on Internet Computing, pages 269-278, 2005.

[41] Condor Manual available at http://www.cs.wisc.edu/condor/manual/v6.5/

[42] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Condor and the Grid", in Fran Berman, Anthony J. G. Hey, Geoffrey Fox, in Grid Computing: Making the Global Infrastructure A Reality, John Wiley, 2003, ISBN:0-470-85319-0.

[43] F Ferstl, "Job and resource management systems", in R Buyya(ed.), High Performance Cluster Computing: Architectures and Systems. Vol. 1. Upper Saddle River NJ: Prentice Hall PTR, 1999.

[44] T Tannenbaum, D Wright, K Miller, and M Livny, "Condor – a distributed job Scheduler" in T Sterling(ed.), Beowulf Cluster Computing with Linux, Cambridge, MA; MIT Press, 2001.

[45] Hawkeye Team. Hawkeye, Monitoring and Management Tool for Distributed Systems, 2004. http://www.cs.wisc.edu/condor/hawkeye

[46] Information about Ganglia is available at http://ganglia.info/

[47] Ganglia distributed monitoring and execution system, March 2004, http://ganglia.sourceforge.net/.

[48] RFC 1014 – XDR: External Data Representation Standard, June 1987, http://www.faqs.org/rfcs/rfc1014.html.

[49] RRDTool, June 2004.

http://www.caida.org/tools/utilities/rrdtool/.
[50] R-GMA details are available at
http://www.r-gma.org
[51] Mark Baker and Garry Smith, "GridRM: A Resource Monitoring Architecture for the Grid", Lecture Notes in Computer Science, Springer, Volume 2536/2002, ISBN:978-3-540-00133-1, pages:268-273.
[52] J. Cao, D.J. Kerbyson and G.R. Nudd, "Use of agent-based service discovery for resource management in metacomputing environment", in: Proceedings of 7th International Euro-Par Conference, LNCS 2150, Manchester, UK, 2001, pp. 882–886.

Appendix A

Installation of GT4

GT4 is downloadable from http://www.globus.org/toolkit/downloads/4.0.6/

In our lab we installed GT4 on Fedora Core 6 so we downloaded gt4.0.6-all-source-installer.tar.gz from the above mentioned site.

1. Before Installation

- Create a user named 'globus' in the local machine. If installing in the head node of the cluster and username is shared between nodes, create a common user to all the nodes.
- Create a directory 'globus-4.0.6' and set its access rights so that 'globus' user has the full control over it:
  [root@goel ~] # mkdir /usr/local/globus-4.0.6
  [root@goel ~] # chown globus:globus /usr/local/globus-4.0.6
- Set the basic environment variable GLOBUS_LOCATION to /usr/local/globus-4.0.6

2. Prerequisites

The following table provides the details of the packages that need to be installed before the installation of GT4.

Direct Downloads *	Website	Compulsory Requirement	Description
GNU tar,sed,Make,gcc	http://www.gnu.org	Yes	Basic tools
Ant 1.5.1+	http://ant.apache.org/	Yes	platform independent build tool
J2SE 1.4.2+	http://java.sun.com/j2se	Yes	java standard edition. this is not directly downloadable
zlib 1.1.4+	http://www.gzip.org	Yes	compression libs required by GPT
sudo	http://www.courtesan.com/sudo	Yes	a tool that provides normal users to execute super user level commands, used by GRAM
JDBC compliant database	http://www.postgresql.org	Yes	recommended is postgresql 7.1+ for RFT, CAS etc
IODBC	http://www.iodbc.org	-	for RLS(Replica Location Service)
Tomcat	http://jakarta.apache.org/tomcat/	-	optional for runtime needs
Junit	http://www.junit.org	-	Needed by ant for unit testing.
Mail server	-	-	Needed for sending and receiving certificates.

* **Direct Downloads:** These are downloads that can be attained directly by clicking the link.

3. Building the Toolkit

- Build the Globus toolkit as globus user

```
[root@goel ~ ] # su globus
[globus@goel ~ ] $ cd /home/globus
[globus@goel ~ ] $ tar xzf gt4.0.6-all-source-installer.tar.gz
[globus@goel ~ ] $ cd gt4.0.6-all-source-installer
[globus@goel ~ ] $ ./configure
[globus@goel ~ ] $ make
[globus@goel ~ ] $ make install
```

4. Setting up Security on first machine

- Create the SimpleCA
  [globus@goel ~] $ $GLOBUS_LOCATION/setup/globus/setup-simple-ca
  globus_simple_ca_xxxxxxxx_setup-0.18.tar.gz is created in the directoy /home/globus/.globus/
- Make my machine trust that new CA by running the following commands as root
  [globus@goel ~] $ su root
  [root@goel~]$$GLOBUS_LOCATION/setup/globus_simple_ca_xxxxxxxx_setup/setup-gsi –default
  Notice that the hash value xxxxxxxx matches the hash value of my SimpleCA.
- Get a hostcert for the machine
  [root@goel ~] # source $GLOBUS_LOCATION/etc/globus-user-env.sh
  [root@goel ~] # Grid-cert-request –host 'hostname'
  Note: we can contact the Simple CA at kunal@goel
- Sign the certificate using SimpleCA as globus
  [root@goel ~] $ su globus
  [root@goel ~] $ Grid-ca-sign –in /etc/Grid-security/hostcert_request.pem –out /home/globus/hostsigned.pem
  We get the new signed certificate 01.pem at /home/globus/.globus/simpleCA/newcerts
- Copy signed certificate into /etc
  [root@goel ~] $ cp ~globus/hostsigned.pem /etc/Grid-security/hostcert.pem
- Create host certificate/key owned bt the globus

  [root@goel ~] $ /etc/Grid-security

  [root@goel ~] $ cp hostcert.pem containercert.pem

  [root@goel ~] $ cp hostkey.pem containerkey.pem

  [root@goel ~] $ chown globus : globus container*.pem
- Get a usercert for kunal
  [kunal@goel ~] $ source $GLOBUS_LOCATION/etc/globus-usr-env.sh
  [kunal@goel ~] $ Grid-cert-request
  Note: contact globus simpleCA at kunal@goel
- Get the certificate request to globus user so it can be signed, then send the signed cert back to kunal
  [kunal@goel ~] $ cat /home/kunal/.globus/usercert_request.pem | mail globus@goel
  [kunal@goel ~] $ su globus
  [globus@goel ~] $ Grid-ca-sign –in request.pem –out signed.pem
  [globus@goel ~] $ cat signed.pem | mail kunal@goel
- kunal copies the cert to the proper location
  [globus@goel ~] $ su kunal
  [kunal@goel ~] $ cp signed.pem ~/.globus/usercert.pem

- Create Grid-mapfile as root for authorization
  [root@goel ~] # cd /etc/Grid-security
  [root@goel ~] # vim /etc/Grid-security/Grid-mapfile
  [root@goel ~] # cat /etc/Grid-security/Grid-mapfile
  " / O=Grid/ OU=simpleCA-goel.thapar.edu/ OU=thapar.edu/ CN=kunal" kunal

5. Set up GridFTP

Secure credentials are in place now, it's time to start a service

- Create Gridftp service
  [root@goel ~] $ cd /etc/Grid-security
  [root@goel ~] $ vim /etc/xinetd.d/Gridftp
  [root@goel ~] $ cat /etc/xinetd.d/Gridftp

```
service gsiftp
{
instances               = 100
socket_type             = stream
wait                    = no
user                    = root
env                     += GLOBUS_LOCATION=/usr/local/globus-4.0.6
env                     += LD_LIBRARY_PATH=/usr/local/globus-4.0.6/lib



server                  = /usr/local/globus-4.0.6/sbin/globus-Gridftp-
server
server_args             = -i
log_on_success          += DURATION
nice                    = 10
disable                 = no
}
```

[root@goel ~] $ vim /etc/services
[root@goel ~] $ tail /etc/services

```
vboxd           20012/udp
binkp           24554/tcp                       # binkp fidonet
protocol
asp             27374/tcp                       # Address Search
Protocol
asp             27374/udp
dircproxy       57000/tcp                       # Detachable IRC Proxy
tfido           60177/tcp                       # fidonet EMSI over
telnet
fido            60179/tcp                       # fidonet EMSI over TCP

# Local services
gsiftp          2811/tcp
```

[root@goel ~] $ /etc/init.d/xinetd reload
[root@goel ~] $ netstat –an | grep 2811

- Gridftp server is waiting for a request, so run a client and transfer a file
  [root@goel ~] $ cd /
  [root@goel ~] $ su kunal

[kunal@goel ~] $ Grid-proxy-init –verify –debug
[kunal@goel ~] $ globus-url-copy gsiftp://kunal.thapar.edu/etc/group file:///tmp/bacon.test.copy
[kunal@goel ~] $ diff /tmp/bacon.test.copy /etc/group

6. Starting the web service container

Setup an /etc/init.d entry for the webservices container

- Create start-stop script as globus user

[root@goel ~] $ su globus
[globus@goel ~] $ vim $GLOBUS_LOCATION/start-stop
[globus@goel ~] $ vim $GLOBUS_LOCATION/start-stop

```
#! /bin/sh
set -e
export GLOBUS_LOCATION=/usr/local/globus-4.0.6
export JAVA_HOME=/usr/java/j2sdk1.5.0_15/
export ANT_HOME=/usr/local/apache-ant-1.7.0

export GLOBUS_OPTIONS="-Xms256M -Xmx512M"

. $GLOBUS_LOCATION/etc/globus-user-env.sh

cd $GLOBUS_LOCATION
case "$1" in
    start)
        $GLOBUS_LOCATION/sbin/globus-start-container-detached -p 8443
        ;;
    stop)
        $GLOBUS_LOCATION/sbin/globus-stop-container-detached
        ;;
    *)
        echo "Usage: globus {start|stop}" >&2
        exit 1
       ;;
esac
exit 0
```

[globus@goel ~] $ chmod +x$GLOBUS_LOCATION/start-stop

- Create an /etc/init.d script to call the globus user's start-stop script

[globus@goel ~] $ su root
[root@goel ~] $ vim /etc/init.d/globus-4.0.6
[root@goel ~] $ cat /etc/init.d/globus-4.0.6

```
#!/bin/sh -e
case "$1" in
  start)
    su - globus /usr/local/globus-4.0.6/start-stop start
    ;;
  stop)
    su - globus /usr/local/globus-4.0.6/start-stop stop
    ;;
  restart)
    $0 stop
    sleep 1
```

```
    $0 start
    ;;
  *)
    printf "Usage: $0 {start|stop|restart}\n" >&2
    exit 1
    ;;
esac
exit 0
```

[root@goel ~] $ chmod +x /etc/init.d/globus-4.0.6

- Start the globus container
  [root@goel ~] $ /etc/init.d/globus-4.0.6 start
  After execution of above command we get RFT warnings. These are expected right now as we havn't setup our database yet.
  127.31.5.50 is my ip address but we get "127.0.0.1". it need to be fixed! Edit

  $GLOBUS_LOCATION/etc/globus_wsrf_core/server-config.wsdd and client-server-config.wsdd, add a line reading <parameter name="logicalHost" value="*172.31.5.50*" /> under the <globalConfiguration> section. For instance:

```
<globalConfiguration>
   <parameter name="logicalHost" value="172.31.5.50" />
```

- Run a sample clients/services to interact with the container
  [root@goel ~] $ su kunal
  [kunal@goel~]$ counter-client –s https://goel.thapar.edu : 8443/wsrf/services/ConnterService

7. Configuring RFT

Configuring PostgreSQL needs the following steps:

- Configure the system to allow TCP/IP connections to postgres, as well as add a trust entry for our current host. First , set "listen_addresses = '*'" in the postgres configuration file.

  [root@goel ~] # vim /var/lib/pgsql/data/postmaster.conf

  [root@goel ~] #grep POSTMASTER /var/lib/pgsql/data/postmaster.conf

  POSTMASTER_OPTIONS="-i"

  Note: -i option is needed to accept TCP/IP connections to database.

  [root@goel ~] # vim /var/lib/pgsql/data/pg_hba.conf

  [root@goel ~] # grep rftDatabase /etc/pgsql/data/pg_hba.conf

  host rftDatabase "globus" "140.221.8.31" 255.255.255.255 md5

  [root@goel ~] # /etc/init.d/postgresql restart

  [root@goel ~] # su postgres –c "createuser –P globus"

- globus user create the rftDatabase

[globus@goel ~] $ createdb rftDatabase

- Load the rft schema

  [globus@goel ~] $ psql -d rftDatabase -f $GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema.sql

- Change the password to globus user's password in jindi-config.xml

  [globus@goel ~] $ vim $GLOBUS_LOCATION/etc/globus_wsrf_rft/jindi-config.xml

  [globus@goel ~] $ grep -C 3 password $GLOBUS_LOCATION/etc/globus_wsrf_rft/jindi-config.xml

```
</parameter>
            <parameter>
                <name>
                password
                </name>
                <value>
                *****
```

- Restart the container to load the new RFT configuration

  [root@goel ~] # /etc/init.d/globus-4.0.6 restart

  [root@goel ~] # head /usr/local/globus-4.0.6/var/container.log

  Great, we got rid of the warning.

- Try an RFT transfer to make sure the service is really working

  [root@goel ~] # su kunal
  [kunal@goel ~] $ cp /usr/local/globus-4.0.1/share/globus_wsrf_rft_test/transfer.xfr /tmp/rft.xfr
  [kunal@goel ~] $ vim /tmp/rft.xfr
  [kunal@goel ~] $ cat /tmp/rft.xfr

```
true
16000
16000
false
1
true
1
null
null
false
10
gsiftp://goel.thapar.edu:2811/etc/group
gsiftp://goel.thapar.edu:2811/tmp/rftTest_Done.tmp
```

  [kunal@goel ~] $ rft –h goel.thapar.edu –f /tmp/rft.xfr
  [kunal@goel ~] $ diff /etc/group /tmp/rftTest_Done.tmp

  RFT did its job, starting up a reliable transfer and notifying us of the status and results.

8. Setting up WS GRAM

[root@goel ~] # visudo
[root@goel ~] # cat /etc/sudoers

```
globus ALL=(bacon) NOPASSWD: /usr/local/globus-4.0.1/libexec/globus-
Gridmap-and-execute
-g /etc/Grid-security/Grid-mapfile /usr/local/globus-
4.0.1/libexec/globus-job-manager-script.pl *
globus  ALL=(bacon) NOPASSWD: /usr/local/globus-4.0.1/libexec/globus-
Gridmap-and-execute
-g /etc/Grid-security/Grid-mapfile /usr/local/globus-
4.0.1/libexec/globus-gram-local-proxy-tool *
```

9. Run the Sample Job

[kunal@goel ~] $ globusrun-ws –submit –c /bin/true

Success. Now we've got a working GRAM installation.

Appendix B

Setting up Computational Grid: Alchemi.NET

Steps for Installing Computational Grid: Alchemi.NET
These are the steps for installing various Alchemi.NET Components. For every component we are giving its installation, configuration and operations are defined.

1. Alchemi.NET Manager Installation

The Alchemi.NET Manager can be installed in two modes
- As a normal Windows desktop application
- As a windows service. (Supported only on Windows NT/2000/XP/2003)

To install the Manager as a windows application, use the Manager Setup installer. For service mode installation use the Manager Service Setup. The configuration steps are the same for both modes. In case of the service-mode, the "Alchemi.NET Manager Service" installed and configured to run automatically on Windows start-up. After installation, the standard Windows service control Manager can be used to control the service. Alternatively the Alchemi.NET Manager Service Controller program can be used. The Manager Service controller is a graphical interface, which is exactly similar to the normal Manager application.
Install the Manager via the Manager installer. Use the sa password noted previously to install the database during the installation.
Configuration & Operation
The Manager can be run from the desktop or Start -> Programs -> Alchemi.NET -> Manager ->
Alchemi.NET Manager. The database configuration settings used during installation automatically appear when the Manager is first started.
Click the "Start" button to start the Manager.
When closed, the Manager is minimized to the system tray.

2. Cross Platform Manager Installation

Install the XPManager web service via the Cross Platform Manager installer.
Configuration
If the XPManager is installed on a different machine that the Manager, or if the default port of the Manager is changed, the web service's configuration must be modified. The XPManager is configured via he ASP.NET Web.config file located in the installation directory (wwwroot\Alchemi.NET\CrossPlatformManager by default):

```
<appSettings>
<add key="ManagerUri" value="tcp://localhost:9000/Alchemi.NET_Node" />
</appSettings>
```

Operation
The XPManager web service URL is of the format
http://[host_name]/[installation_path]
The default is therefore

http://[host_name]/Alchemi.NET/CrossPlatformManager
The web service interfaces with the Manager. The Manager must therefore be running and started for the web service to work.

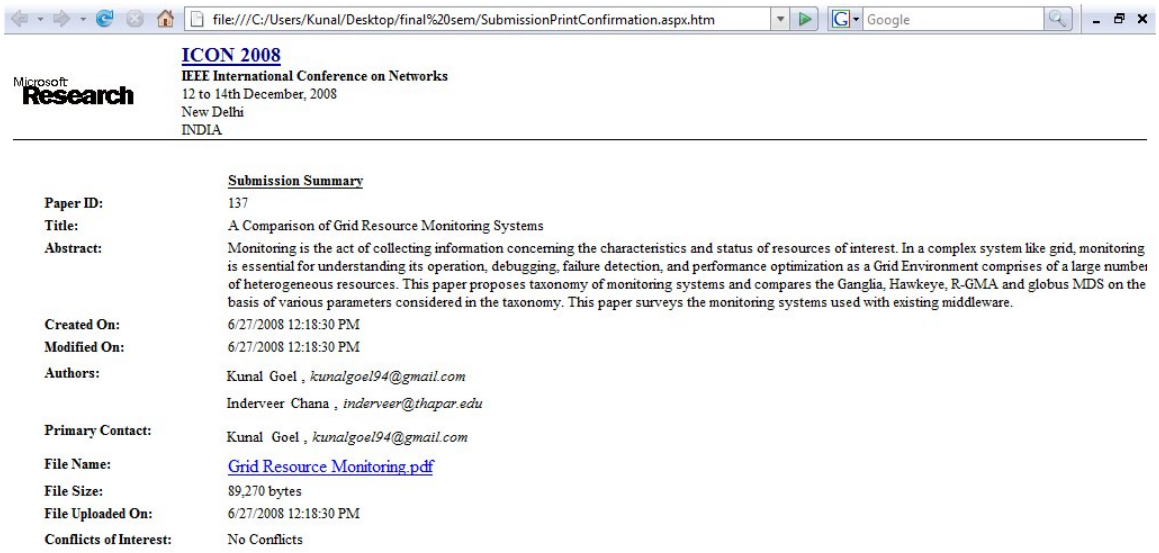
3. Executor Installation

The Alchemi.NET Executor can be installed in two modes
- As a normal Windows desktop application
- As a windows service. (Supported only on Windows NT/2000/XP/2003)

To install the executor as a windows application, use the Executor Setup installer. For service mode installation uses the Executor Service Setup. The configuration steps are the same for both modes. In case of the service-mode, the "Alchemi.NET Executor Service" installed and configured to run automatically on Windows start-up. After installation, the standard Windows service control Manager can be used to control the service. Alternatively the Alchemi.NET ExecutorServiceController program can be used. The Executor service controller is a graphical interface, which looks very similar to the normal Executor application.
Install the Executor via the Executor installer and follow the on-screen instructions.
Configuration & Operation
The Executor can be run from the desktop or Start -> Programs -> Alchemi.NET -> Executor -> Alchemi.NET Executor. The Executor is configured from the application itself. You need to configure 2 aspects of the Executor:
The host and port of the Manager connect to Dedicated / non-dedicated execution. A non-dedicated Executor executes Grid threads on a voluntary basis (it requests threads to execute from the Manager), while a dedicated Executor is always executing Grid threads (it is directly provided Grid threads to execute by the Manager). A non-dedicated executor works behind firewalls and Click the "Connect" button to connect the Executor to the Manager.

Papers Communicated / Accepted / Published

Kunal Goel, Inderveer Chana, "A Comparison of Grid Resource Monitoring Systems" communicated in IEEE International Conference on Networks, ICON 2008, at India Habitat Centre, New Delhi from 12 to 14 December, 2008**(Communicated)**.



file:///C:/Users/Kunal/Desktop/final%20sem/SubmissionPrintConfirmation.aspx.htm

Microsoft Research

ICON 2008
IEEE International Conference on Networks
12 to 14th December, 2008
New Delhi
INDIA

Submission Summary

Paper ID:	137
Title:	A Comparison of Grid Resource Monitoring Systems
Abstract:	Monitoring is the act of collecting information concerning the characteristics and status of resources of interest. In a complex system like grid, monitoring is essential for understanding its operation, debugging, failure detection, and performance optimization as a Grid Environment comprises of a large numbe of heterogeneous resources. This paper proposes taxonomy of monitoring systems and compares the Ganglia, Hawkeye, R-GMA and globus MDS on the basis of various parameters considered in the taxonomy. This paper surveys the monitoring systems used with existing middleware.
Created On:	6/27/2008 12:18:30 PM
Modified On:	6/27/2008 12:18:30 PM
Authors:	Kunal Goel , *kunalgoel94@gmail.com* Inderveer Chana , *inderveer@thapar.edu*
Primary Contact:	Kunal Goel , *kunalgoel94@gmail.com*
File Name:	Grid Resource Monitoring.pdf
File Size:	89,270 bytes
File Uploaded On:	6/27/2008 12:18:30 PM
Conflicts of Interest:	No Conflicts