CHAPTER 4

PERFORMANCE ANALYSIS OF RESOURCE AGGREGATOR IN GRID OF GRIDS ENVIRONMENT

4.1 INTRODUCTION

There are many heterogeneous grid middlewares established and used for various science applications in the past. Most of these Grids, however, work in isolation and with different utilization levels. In the previous chapter the EGSIA architecture and a mechanism to enable resource sharing amongst heterogeneous grids was discussed and proved with experimental results.

In the present chapter, an analytical model for the proposed system is derived. The purpose of the model is to find an average delay incorporated in various types of resource requests under a dynamic working condition of resource aggregator, and also to estimate the cost involved in serving resource requests. The performance of the resource aggregation system is evaluated and presented the model for semantic matchmaking system based on semantic distance. The various challenges and scheduling mechanisms are discussed in the subsequent sections.

4.2 RESOURCE MANAGEMENT CHALLENGES

Some of the challenges of grids with respect to resource management include the following.
- **Resource monitoring**: Resource monitoring system must be able to provide information about the current status of various resources in a grid and should be able to generate alarms whenever certain important events occur.

- **Resource status updation and communication**: In order to keep up-to-date resource status, continuous monitoring is needed. The increase in the number of status delivery of such monitored observations will consume more bandwidth. It is very much essential to keep track of only relevant changes and communicate instantly.

- **Resource description**: It is the formal process to describe the resources to be shared. The heterogeneity of grid devices, operating systems, services they provide and data they represent needs a standard to describe the resources so that any external grid can utilize the services available with a particular grid.

- **Resource discovery**: The concept of resource sharing in grid computing is realized by service discovery mechanisms which transparently and seamlessly locate available resources/services throughout the grid infrastructure upon request. Since the users want the service to be as faster as possible at a reasonable cost, service discovery becomes a more important issue in grid computing (Krauter et al 2002, Litke et al 2004). Hence proper publishing, indexing, and cataloging of shared resources become very essential. Standards such as GRDL (Grid Resource Definition Language), WSDL (Web Services Definition Language), are available for resource discovery.
- **Resource allocation:** Proper job scheduling and/or resource allocation becomes very crucial in order to provide efficient service to grid users.

### 4.3 SCHEDULING AND RESOURCE AGGREGATION

Job scheduling strategies have been extensively studied in past decade. In the new distributed scenarios, such as grid systems, traditional scheduling techniques have evolved into more complex and sophisticated approaches where other factors, such as the heterogeneity of resources or geographical distribution, have been taken into account.

To reduce the complexity of the resource sharing in these highly dynamic and heterogeneous scenarios, new components and software layers are required. In particular, middleware such as Globus (Foster 1998a) UNICORE (Erwin and Snelling, 2001), or EGEE (Enabling Grids for E-SciencE) gLite (Laure 2006) and higher level services are deployed to perform the job and resource management. In this context, the scheduling task consists of planning the jobs among the resources which are scattered in different centers. The software component responsible for the resource management and scheduling is usually called grid resource broker or meta-scheduler. Efficient resource management will try to satisfy the user requirements and system global performance (Mengistu et al 2007).

Several grid scheduling systems are proposed based on economic issues (Abramson 2002) adaptiveness (Huedo et al 2004), co-allocation strategies (Mohamed and Epema 2008) application-centric approach (Berman and Wolski 1996), advance reservations (Elmroth and Tordsson 2004) and Service Level Agreements (SLA) (Seidel et al 2007). In general, scheduling component will orchestrate all the tasks of finding the appropriate computing resource, finding the software libraries, moving the required data, carry out
the execution and giving back the results of the computation to the user. Thus, the complexity of accessing many heterogeneous systems is reduced to one single interface. Actually, the grid middleware allows interoperability among different distributed systems by translating operations from this interface to local interfaces. Although the original idea of grid computing promised an infrastructure to provide a uniform access to resources across different centers and institutions, the majority of grids have focused on different isolated projects and regional initiatives.

Some examples are TeraGrid in US (Catlett et al 2006) GridX1 in Canada (Agarwal and Ahmed 2007), Naregi in Japan, APACGrid in Australia (Dunning and Nandkumar 2006), Garuda in India (Ram and Ramakrishnan 2006) Grid5000 in France (Bolze and Cappello 2006), DAS-2 in the Netherlands, D-Grid in Germany, e-Science in UK (Hey and Trefethen 2002), and EGEE in Europe (Donno et al 2003). Thus, the need for inter-operability among different grid systems has become necessary for large grid environments. Thus, despite the user having access to many different grid systems with different access mechanisms, he/she has to only deal with one unique interface. In this model, a resource aggregation mechanism to aggregate resources from some of the widely used middleware’s such as Globus (2 and 4), gLite 3 and UNICORE is proposed. A generic interface model which interacts with these middlewares is attempted and that can also support other middlewares in an uniform manner. Further, this interface is integrated with a semantic match making module, which translates the grid resource information into ontology based descriptions. The resource information aggregated by the aggregator is then be used to support semantic description and discovery of grid resources.
4.4 RELATED GRID MIDDLEWARE INTEROPERABILITY MODELS

The proposed analytical model mechanisms and policies are related to previous systems and techniques in several ways. In this chapter the focus is on few models which are given below.

**Resource sharing networks and Delegated match making model**

Iosup et al (2007) proposed a solution to this problem that combines the hierarchical and decentralized approaches for interconnecting grids and they proposed a delegated matchmaking model (DMM). In their model, a hierarchy of grid sites is augmented with peer-to-peer connections between sites under the same administrative control. To operate this architecture, they employed the key concept of delegated matchmaking, which temporarily binds resources from remote sites to the local environment. In order to serve larger and more diverse communities of scientists, the next step in the evolution of grids is to inter-operate several grids into a single computing infrastructure. This raises additional challenges to traditional resource management, such as load management between separate administrative entities.

The drawback of this DMM model is that, the extension of their simulations to more heterogeneous platform is required for various services like accounting for resource and job failures, and to investigate the impact of existing and unmovable load at cluster level.

**Agents based resources sharing architecture**

Kenneth proposed the design and implementation of Sharing Networked Resources with Brokered Leases (Shirako), a system for on-
demand leasing of shared networked resources. This prototype model is based on service oriented architecture for resource providers and consumers to negotiate access to resources over time, arbitrated by brokers. Resource types have attributes that define their performance behavior and degree of isolation. Shirako decouples fundamental leasing mechanisms from resource allocation policies and the details of managing a specific resource or service. It offers an extensible interface for custom resource management policies and new resource types. Experiments with the prototype quantify the costs and scalability of the leasing mechanisms, and the impact of lease terms on fidelity and adaptation. However, resource exchange amongst shirako brokers has not been explored yet.

**Cluster based load sharing Model**

Service level Agreement based coordination mechanism for grid super scheduling was proposed by Ranjan et al (2006). These SLA polices does not have control over the gateways.

Balazinska et al (2004) have proposed a load balancing mechanism for Medusa. It is a stream processing system that allows the migration of stream processing operators from overloaded resources to the other resource with spare capacity. The mechanism of load sharing among the heterogeneous grids devised by Medusa model is considered in the present work. Also, the proposed model differs in exchanging services between heterogeneous middlewares.

**FCFS policy based resource allocation**

The First come first server (FCFS) policy model was proposed by Mualem et al (2001). This allows a job to jump in the queue and execute earlier than jobs that arrived before it, given that enough resources are
available, other waiting jobs are not delayed. If the maximum number of
pivots is set to a large number, the FCFS backfilling algorithm becomes
conservative backfilling. Work on multiple resource partitions and priority
scheduling has shown to reduce job slowdown compared to FCFS by Lawson

A cost estimation model is derived along with semantic match
making system for resource sharing among the heterogeneous grids and their
details are discussed in the subsequent sections.

4.5 PROPOSED RESOURCE SCHEDULING IN
HETEROGENEOUS GRID MIDDLEWARE

In chapter 3, a mechanism of RA was introduced by incorporating
RA as a part of resource scheduling. Since the proposed model is based on
semantic matchmaking algorithm, which is an ontology oriented approach,
performance improvement in terms of time complexity for discovering
resources from four grids considered for experimentation is observed.
Through an empirical study, the performance of the system is analyzed in
terms of delays and costs incurred while operating in such an environment.
The performance analysis of ontology based semantic matchmaking system
shows how the proposed system will reduce latency involved in resource
discovery operations.

The proposed mechanism allows a Grid to reschedule a request to a
underutilized Grid when the cost of serving itself is high. The rescheduling
takes place between Grids that have a pre-established SLA for sharing of
resources. The proposed RA is based on gateway approach which allows
participants to seamlessly allocate resources from heterogeneous Grids. The
metascheduler is incorporated along with the RA. The metascheduler acts as
resource scheduler among the heterogeneous grid environment.
When the Grid user submits the job to RA for number for resources, Gridway metascheduler identifies the necessary Grid resources and selects the suitable Grid based on their SLA and policies of each individual grid environment.

4.6 **ANALYTICAL MODELING**

The analytical model for the RA with metascheduler based system proposed system is explained in detail in the present section. The purpose of the model is to find the average delay incorporated in various types of resource requests under a dynamic working condition of the RA, and also to estimate the cost involved in serving resource requests. The performance of the resource aggregation system has also been evaluated a model for semantic matchmaking system based on ontology is presented.

4.6.1 **Average Delay for Resource Aggregation**

As explained earlier, the RA includes interface drivers for collecting resource information from various middlewares. The RA is considered to be connected to n grids, and each grid is represented as g_i, where i is the index of the grid, which varies from 1 to n. The message transition diagrams shown in Figure 4.1 is used to derive the average resource aggregation delay. It is considered that, after receiving a resource request, the RA initiates various interface drivers connected to it, to obtain the resources available on the respective grids. Locally aggregated resources are sent to the RA for final resource aggregation and to create resource catalog.

The delay in resource aggregation is defined as the time taken for the RA to aggregate requested resources from various grids. The average delay, T_{avg}, is defined as the sum of the delays over a type of resource (k), requested in a unit time.
\[ T_{\text{avg}} = \sum_{i=1}^{k} \lambda_i T_i^{(j)} \]  

(4.1)

where \( \lambda_t \) is the arrival rate of requests for resource type-\( t \), and \( T_i^{(j)} \) is the delay per resource type-\( i \), with \( j \), the number of grids operating at that instance of time \( t \) (\( j \leq n \)). The Table 4.1 shows set of time parameters used in analysing delay in generation of resource catalog. The, \( T_i^{j} \) can be expressed as

\[ T_i^{j} = cT_x \]  

(4.2)

where \( c \) is the coefficient of \( T_x \), denotes the number of such time parameters required for type \( i \) resource request. If the number of hops between the RA, and the selected grid is \( N_h \), then for various types of resource requests, the resource aggregation delay for resource request \( i \) is given by.

\[ T_i^{j} = \begin{cases} T_{\text{rf}} + N_h (T_{\text{np}} + T_{\text{rq}}) + T_{\text{ma}} + N_h (T_{\text{np}} + T_{\text{rst}}) & \text{Resource i not cataloged} \\ 0 & \text{Resource i cataloged} \end{cases} \]  

(4.3)

There is no request analysis time if the resource is already cataloged at the RA level, this is the major time saving in the proposed system. The arrival rate of requests for resource type-\( i \), i.e., \( \lambda_i \) is given by

\[ \lambda_i = \lambda_u P_i \]  

(4.4)

This arrival is considered as a Poisson process with average rate \( \lambda_u \), with the PDF of the transactions inter-arrival time as

\[ f_\lambda(t) = \lambda_u e^{-\lambda_u t} \]  

(4.5)
The \( P_i \) is the probability of occurrence of type-i request. By considering a particular time interval \((t, t + \Delta t)\), the number of type-i requests appearing in this interval is given by, \( I(t, t + \Delta t) \). The \( P_i \) is given by,

\[
P_i = \int_0^\infty P[I(t, t + \Delta t) = 1] \int_0^\infty \lambda_i \Delta t e^{-\lambda_i \Delta t} \, d\lambda_i \Delta t \quad (4.6)
\]

![Figure 4.1 The resource aggregation message transition diagram](image)

**Table 4.1 Time parameters**

<table>
<thead>
<tr>
<th>Symbols</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{rf} )</td>
<td>Request formulation time by the RA</td>
</tr>
<tr>
<td>( T_{rqp} )</td>
<td>Request propagation time on selected grid</td>
</tr>
<tr>
<td>( T_{rqt} )</td>
<td>Request transmission time on selected grid</td>
</tr>
<tr>
<td>( T_{rqa} )</td>
<td>Request analysis time by the selected grid</td>
</tr>
<tr>
<td>( T_{rsp} )</td>
<td>Response propagation time on selected grid</td>
</tr>
<tr>
<td>( T_{rst} )</td>
<td>Response transmission time on selected grid</td>
</tr>
</tbody>
</table>
4.6.2 Cost of Serving a Resource Request by a Grid

The resource providers use a pricing function per a resource request unit, which is given by

\[ \text{Price} = C + (C \times L) \]  \hspace{1cm} (4.7)

where \( C \) is the fixed cost of a unit of resource at the provider, load \( L \) is obtained from the policy in use by the resource provider grid; load is the estimate when the policy supports forecasts or the actual load of both running and waiting jobs in the queue. A resource unit corresponds to one resource per second (i.e. a second of a CPU). Although straightforward, this pricing function has two components that capture namely the fixed cost of resources and the variable price caused by the demand as shown in Figure 4.2 (Surana et al 2006).

For each grid \( g_i \), the allocation of its resources to user communities over a unit of time represents a cost. The real valued cost function of the participating \( g_i \) is represented by \( \text{cost}_i(L) \), where \( 0 \leq L \leq 1 \) is the current load determined by the number of resource units in use in grid \( g_i \). Therefore, the cost given by \( \text{cost}_i(L) \) depends on the number of resources allocated for the requests. Although each grid could have its own cost function, it is assumed that, the participating grids utilize a quadratic cost function. The use of a quadratic function allows us to specify contracts with price ranges. The cost function \( \text{cost}_i(L) \) is given by,

\[ \text{cost}_i(L) = L_{\text{units}} \times \left( pc + \left( pc \times (\beta L)^2 \right) \right) \]  \hspace{1cm} (4.8)

where \( L_{\text{units}} \) is the number of units in use at load \( L \), \( \beta \) is a minimal constant value that determines how steep the cost curve is as the load approaches 1 and
\[ p_c = \sum_{k=1}^{n} \left( c_{p_i} \left( \frac{r_{u_i}}{\sum_{k=1}^{n} r_{u_j}} \right) \right) \]  

(4.9)

where \( n \) is the number of \( g_i \)'s, \( c_{p_i} \) is the price of a resource unit at resource provider \( g_i \), and \( r_{u_i} \) is the number of resource units contributed by provider \( g_i \) until a given time horizon, which could be a period of contract between the RA and the respective \( g_i \).

Figure 4.2 The cost estimation model

4.7 PERFORMANCE EVALUATION OF RESOURCE AGGREGATION SYSTEM

The metrics that are used for performance evaluation is Average Weighted Response Time (AWRT) of resource requests. The AWRT measures how long average users wait to have their resource request
completed (de Assuncao and Buyya 2009). A small AWRT indicates that average users do not wait long for their requests to complete. The AWRT, demonstrates whether the response time of user requests is improved through aggregating g_i's during resource discovery process, which is given by

$$\text{AWRT} = \frac{\sum_{i \in k} T_{\text{req}} \times m_i \times (T_{\text{req}}^{(m_i)} - S_j)}{\sum_{i \in k} T_{\text{req}} \times m_i}$$

(4.10)

where \(s_i\) is request submission time for resource of type-i, \(m_i\) is the number of grids available at that instance of time, \(T_{\text{req}}\) is the resource request analysis time, and \(T_{\text{req}}^{(m_i)}\) is the time of discovering resource of type-i, when \(m_i\) number of grids are operational. The resource consumption \((T_{\text{req}} \times m_i)\) of each request for resource of type-i is used as the weight for normalization.

### 4.8 MODELING OF SEMANTIC MATCHMAKING SYSTEM USED

The semantic matchmaking system based on the ontology works on the principle of finding out semantic distance between any two concepts (Broens et al 2004, Shu et al 2007, Xing et al 2006, Bellur et al 2008). Let \(C_i\) be a concept, where \(i = 1 \ldots n\). \(D(C_i)\) is the domain of concept \(C_i\), a domain can have any number of such concepts, and all the concepts are grid resource descriptions. This model assumes that the semantic matchmaking happening between concepts belonging to the same domain.
Figure 4.3 An ontology tree of concept of resource

The distance d between two resource concepts is computed using a real function Rf, which maps the Cartesian product of domains into a positive real number R⁺.

\[ d = Rf \left( D(C_i) \times D(C_i) \right) \rightarrow R^+ \]  \hspace{1cm} (4.11)

The semantic distance of two resource concepts is the sum of the "subsumption" distance (ds), i.e., some resource concept is subsumed by some other resource concept, and "definition" distance (dd) which matches the semantics of two definitions, therefore

\[ d(C_1,C_2) = ds(C_1,C_2) + dd(C_1,C_2) \]  \hspace{1cm} (4.12)

where, ds is the distance between two resource concepts within a hierarchy, while the dd is the difference between the semantic description of resource two concepts, which is calculated by the matchmaking algorithm. The dd(C1,C2) will be ignored, if a resource concept C2 is derived from resource concept C1, because C2 inherits all the properties of C1 due to inheritance. The Table 4.2 illustrates computation of conceptual distance d, by considering a sample resource description ontology tree given in the Figure 4.3.
Table 4.2 Distance between concepts

<table>
<thead>
<tr>
<th>Instance</th>
<th>Result</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>d(C11,C11)</td>
<td>0</td>
<td>Cyclic, distance between a resource concept to itself</td>
</tr>
<tr>
<td>d(C1,C11)</td>
<td>ds(C1,C11)</td>
<td>Since dd(C1,C11) = 0, because C111 is offspring of C1</td>
</tr>
<tr>
<td>d(C111,C1)</td>
<td>ds(C111,C1) + dd(C111,C1) = 2 +dd(C111,C1)</td>
<td>The distance of hierarchy of resource concepts C111 and C1 is 2</td>
</tr>
<tr>
<td>d(C111,C12)</td>
<td>ds(C111,C12) + dd(C111,C12) = 1 +dd(C111,C12)</td>
<td>The distance of hierarchy of resource concepts C111 and C12 is 1</td>
</tr>
<tr>
<td>d(C11,C12)</td>
<td>ds(C11,C12) + dd(C11,C12) =dd(C11,C12)</td>
<td>where ds(C11,C12) = 0 because C11 and C12 are at the same level</td>
</tr>
</tbody>
</table>

4.9 SIMULATION AND RESULTS

This section presents a detailed discussion on environment used for conducting experiments, to analyze the performance of the resource aggregator. The section highlights simulation environment and discusses the various results obtained.

4.9.1 Simulation Environment

In four different clusters, the middleware Globus 2, Globus 4, UNICORE and gLite 3 has been installed. The Globus cluster contains two nodes each to perform job execution and they use PBS as their local resource managers. Similarly, in gLite cluster one node is configured as Computing Element (CE) and the other as Worker Node (WN). The UNICORE cluster is built with four nodes each installed with UNICORE 6. The CE is configured as site BDII. A simple CA has been used to set up Grid Security Infrastructure in all the four resources.
Further, there is a machine named F as shown in Figure 4.4 where the resource aggregator and the central repository resides. There is a XML file called as resource registration file in which hostname and middleware of every resource is present. This hostname can be site BDII or Top level BDII of gLite, and they can either be local GRIS or site GIIS of Globus. In case of UNICORE, it is the hostname of the machine. A parser module present in the aggregator parses the XML file and invoke appropriate driver to "pull" resource information from respective resources.

A user interface is provided for requesting a resource which is deployed onto the tomcat container in machine F. Currently, there is a provision to request the required operating system and its release. It can further be extensible to include the memory required to execute the job and the number of nodes. The resource aggregator developed in the present work is extensible to extract resource information from other middleware. In such cases, appropriate driver shall be written and plugged in with the aggregator.

![Figure 4.4 Experimental Setup](image-url)
4.9.2 Results and Discussion

An experiment has been conducted with four Globus 2 resources, four Globus 4 resources, four gLite computing elements, and four UNICORE resources to measure the time taken for aggregating resource information. The Figure 4.5 demonstrates utilization of Grids under three workload conditions. The load conditions considered are {LOW, NORMAL, HIGH}. The workload distribution model is proposed based on the number of resource requests generated by heterogeneous users (Trived et al 2006). Four Grids are operated under various interface drivers, demonstrating the cost involved in serving the requests. Quadratic cost functions are estimated and used for each Grid. The Figure 4.6 shows arrival pattern of four types of resource requests from user portals. Many times, request arrival depends on the previous arrival of other request, and few times they are independent.

![Grid Utilization](image)

Figure 4.5 Grid Utilization under Various Workloads
The average time taken for contacting and collecting information from a Globus 2.4 resource is 3.25 milliseconds and Globus 4.x resource is 2.75 milliseconds. Similarly for a gLite computing element the average time taken is 1.75 milliseconds, and for a UNICORE resource it is 3.25 milliseconds. With the experimental setup, the time taken for aggregating resource information and storing it in the database is 11 milliseconds. The time complexity measured while aggregating resource information is given in Figure 4.7. The four resource characteristics considered for experimentation are R1:Storage, R2:Memory, R3:Video Memory, and R4:CPU Cycles.

In this modeling, minimal numbers of resources for each middleware are considered. In case of the UNICORE 6.0 middleware which is a beta version, only static information has been retrieved, as no provision for retrieving dynamic information of a UNICORE resource is seen. To contact the Globus resources (GT2 and GT4), the anonymous option is used to skip the credential checking at the resource site. This option is used as the
main objective is to aggregate the resource information, but establishing security features is outside the scope of the present work.

Figure 4.7 Time complexity for Resource Aggregation

Following equations demonstrate the Average Weighted Response Time (AWRT) with and without resource aggregator environments.

\[
AWRT_{K\text{(withoutRA)}} = \frac{j \times r_k \times p_j \times m_j \times (c_j - s_j)}{j \times r_k \times p_j \times m_j} \quad (4.13)
\]

\[
AWRT_{K\text{(withRA)}} = \frac{i \times r_k \times t_{qa} \times m_i \times (t_{mi} - s_i)}{i \times r_k \times t_{qa} \times m_i} \quad (4.14)
\]

The equation 4.13, indicates the AWRT of Resource requests without the use of resource aggregator. The response time function depends on the process execution time. The equation 4.14, represents the AWRT of Resource requests with the resource aggregator. Here the response function depends on the request analysis time of the selected grid. The result has been depicted in Figure 4.8. By comparing 4.13 and 4.14, it is inferred that \( t_{qa} < P_i \). Hence the request analysis time is less than the execution time of the request. Further it is found that, \( (t_{mi} - s_i) < (c_j - s_j) \). Since the time for discovering
resource of type i is less than the time required for the completion of service of that particular request. The Table 4.3, shows comparisons for sample test run on various types of grids and Figure 4.8 shows the plot of the AWRT comparison

<table>
<thead>
<tr>
<th>Type of Grid</th>
<th>With Resource aggregator</th>
<th>Without Resource aggregator</th>
</tr>
</thead>
<tbody>
<tr>
<td>GT2</td>
<td>3.25</td>
<td>3.75</td>
</tr>
<tr>
<td>GT4</td>
<td>2.5</td>
<td>3.25</td>
</tr>
<tr>
<td>gLite</td>
<td>1.75</td>
<td>3.1</td>
</tr>
<tr>
<td>UNICORE</td>
<td>3.25</td>
<td>4.12</td>
</tr>
</tbody>
</table>

Figure 4.8 AWRT Comparison

4.10 SUMMARY

The motivation behind the present work is to develop an effective resource aggregation technique, which improves the performance of operation
of islands of Grids. This chapter proposes a mechanism of resource aggregation by incorporating resource aggregator as a part of resource scheduling. Since the proposed model is based on semantic matchmaking algorithm, which is an ontology oriented approach, there is a considerable performance improvement in terms of time complexity for resource discovery from four grids. Through the empirical study conducted, the performance of the system is analyzed in terms of delays and costs incurred while operating such an environment. Further, it is also presented that the performance of ontology based semantic matchmaking system which showed reduced latency involved in resource discovery operations. The proposed system operates on a message exchanges based protocol, hence there is a very good opportunity for implementing this scheme using intelligent agents. This would bring an automatic and highly sophisticated environment to operate islands of Grids.