# 私立東海大學

# 資訊工程與科學研究所

## 碩士論文

指導教授：楊朝棟 博士

在多重網格計算環境上具監控服務的資源經紀人之實作

**Design and Implementation of a Resource Broker with Monitoring Service on Multiple Computational Grid Environments**

研 究 生：胡文仁

中華民國九十八年六月

# 摘要

網格計算被廣泛的應用，透過不同的虛擬組織整合了許多分散式的資源來達到高效能的計算，有別以往的單一網格環境，所可利用的資源受限於其虛擬組織的規模大小而有限，在本論文當中，我們提出一個多重的網格新架構，於本研究當中於多重網格環境之上層，建置一個資源代理人的平台，整合了來自於不同虛擬組織的單一網格環境，形成一個更寵大的計算資源，打破不同虛擬組織之間的限制，讓所屬於不同網格當中的計算資源可以被更有效的利用，避免不必要的資源浪費。另外，提出一個跨網格環境的資源配置策略，讓資源代理人在分配工作置整個多重網格環境時，達到較佳的資源分配，避免受到網路壅塞的關係而降低效能。

**關鍵字:** 網格計算、多重網格、資源代理人、跨網格資源配置

# Abstract

Grid computing is now in widespread use, which integrates geographical computing resources across multiple virtual organizations to achieve high performance computing. A single grid often not provide a huge resource, because virtual organizations have no the adequate of computing resources restriction on the scale of organizations. In this thesis, we present a new grid architecture, which integrates multiple computational grids, named Multi-Grid, from different virtual organizations. We build a resource broker on multiple grid environments, which integrates a number of single grids from different virtual organizations without the limited of organizations. We can efficiently use the multiple grid resource avoid waste of resource. In addition, we proposed a Multi Grid Resource Selection Strategy for the resource broker to select the better allocation of resource before submitting job avoid network congestion caused of decrease of performance.

**Keywords:** Grid computing、Multi-Grid、Resource Broker、MGRSS

# Acknowledgements

I have been indebted in the preparation of this thesis to my supervisor, Dr. Chao-Tung Yang of Tunghai University, whose patience and kindness, as well as his academic experience, have been invaluable to me. No matter the capability in the open discussions, or the preciseness in thesis writing, Professor Yang gave me a deep influence and inspiration. I would also like to thank Professor Yueh-Min Huang, Professor Win-Tsung Lo, Professor Wen-Chung Shih and Professor Fang-Rong Hsu for their valuable comments and advice given while serving on my reading committee.

I am indebted to my many classmates of lab for providing a stimulating and fun environment in which to learn and grow. They gave me opportunities to gain more knowledge and shared their knowledge with me. My research would not have been completed without the help from them.

Last, I must be grateful beyond place to my parents, who at an early age instilled a love of learning and through the years gave me all of their support and encouragement to pursue it.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

## 1.1 Motivation

Grid technology plays a major role in tackling large-scale problems by integrating distributed resources to provide users with a supercomputer liked capacity for data sharing and computation [5, 20, 25]. Participating sites may be physically distributed, heterogeneous, and governed by different administrative domains. Many grid related studies and projects has been proposed for solving large scale scientific problems such as earthquake simulation, atmosphere and ocean simulation, high energy and nuclear physics [47-48], astronomy [48], bioinformatics [29, 46], and medical image processing [28]. There are more and more proposed grid projects such as Globus, Condor, LEGION, Grid PP, EGEE, P-Grid, DutchGrid, ESnet, and Grid Bus [32-40].

Even though Grid computing encounters a conceptual framework, the Globus Toolkit® (GT) that enables Grid computing was realized. The GT, is an open source project developed by the Globus Alliance® for building Grids, not only provides users with an implementation of the necessary services of a middleware to build Grid infrastructures, but also able to be employed to implement immense applications on Grid infrastructures. However, the GT lacks some features, such as a queuing system, a friendly interface, and a proper resource broking manager to respectively process, accept and broke jobs or workloads submitted by users. In addition, a monitoring mechanism that can monitor the status of user jobs is also expected.

In the Grid environment, applications make use of shared Grid resources to improve performance. The target function usually depends on many parameters, e.g., the scheduling strategies, the configurations of machines and links, the workloads in a Grid,

the degree of data replication, etc. In this work, we examine how those parameters may affect performance. We choose an application's overall response time as an object function and focus on dynamically scheduling independent tasks. We define the job, scheduler, and performance model of a Grid site and conduct experiments on Tiger Grid platform. We use the Ganglia [21, 22, 30] and NWS [23, 45] tools to monitor resource status and network-related information [18], respectively. Understanding influence of each parameter is not only crucial for an application to achieve good performance, but would also help to develop effective schedule heuristics and design high quality Grids.

In Taiwan, some research institutes has devoted to build grid platforms for academic research use such as Tiger Grid [27] and Medical Grid [28] that integrate available computing resources in some universities and high schools. They respective use their grid resource, but some of virtual organizations have no the adequate of computing resources restriction on the scale of organizations. For this reason, we construct a multi-grid resource broker [26] integrate those single grid environment as a multiple grid environment so that resources can be used more effectively.

## 1.2 Contributions

In this thesis, we propose a multi-grid architecture to solve the above problem. When a new grid joins in the multi-grid environment by the proposed system, the user can use the entire multi-grid resources, and get the more computing resource. This approach could help these virtual organizations to possess more computing resources without any extra overhead.

We also provide a user-friendly web portal which integrates with the resource broker [7, 11, 14], the cross grid service, the monitoring service, and the multi-grid resource selection strategy, called "Multi-Grid Resource Broker." The main function of

the resource broker is to match the available resources to the user's requirement. The resource broker helps users to select suitable resources according to user preferences and job characteristics. In our architecture, we chose the Globus Toolkit to be the middleware of grids. Although Globus provides a monitor tool, called MDS, which is not capable of providing the rich set of all the requisite information. Thus we used another monitor tool "Ganglia" [21, 22, 30] on the multi-grid system. Ganglia is a scalable open source distributed system for monitoring status of nodes in wide-area systems based on clusters.

We also propose a multi-grid resource selection strategy, called "MGRSS". The strategy helps user select the better performance of machine in order to shorten the execution time of programs, and furthermore we adjusted each grid user has quota of resource. The experimental result shows that MGRSS exhibits a better performance than other strategies

Overall, we construct a multi-grid platform, which integrates three grid systems including the Tiger Grid, the Medical Grid, the Bio Grid, and GCA Grid, and furthermore, design and implement a Multi-Grid Resource Broker with multi-grid resource selection strategy.

## 1.3 Thesis Organization

The rest of this thesis is as follows. Chapter 2 describes a background review of Grid computing, middleware, the related techniques used, and the related work. Chapter 3 shows the system design and implement, includes Resource Broker, Software Stack Diagram Cross Grid Information Server, and Cross Grid Authentication Service, Cross Grid Information Service, and Web portal. Chapter 4 shows the flowchart and algorithm of multi-grid resource selection strategy. Chapter 5 presents experimental

results of comparing the performance of different strategies. Finally, conclusions and future works are discussed in Chapter 6.

# Chapter 2

# Background Review

## 2.1 Grid Computing and Middleware

Grid computing encounters distributed heterogeneous resources, including different platforms, hardware/software, computer architecture, and computer languages, which are geographically distributed and governed by different Administrative Domains over a network using open standards to solve large-scale computational problems. As more Grids are deployed worldwide, the number of multi-institutional collaborations is rapidly growing. However, to realize Grid computing is full potential, it is expected that Grid participants are able to use one another's resources.

Functionally, Grids can be classified as computational Grid and data Grid. The computational Grid is the beacon to scientists for solving large-scale problems like gene comparison, high-energy physics, earthquake simulation, and weather prediction, etc. The subject of this work is the resource management and allocation for a Grid system that is primarily intended to support computationally expensive tasks like simulations and optimizations on a Grid.

### 2.1.1 Globus

The Globus Toolkit (GT) [1, 32], is an open source project developed by the Globus Alliance for building Grids, not only provides users with an implementation of the necessary services of a middleware to build Grid infrastructures, but also able to be employed to implement immense applications on Grid infrastructures. It is important to

note, that the GT, however, offers only the fundamental software and technologies necessary for setting up a Grid.

The GT contains five broad areas includes below:

- Security: Grid Security Infrastructure (GSI) and Community Authorization Service (CAS)
- Execution Management: Web Service Grid Resource Allocation and Management (WS GRAM)
- Data Management: GridFTP and Global Access to Secondary Storage (GASS)
- Information Services: Monitoring and Discovery System (MDS)
- Common Runtime: eXtensible IO (XIO)

WS GRAM is a set of Web services designed to handle requesting and using remote system resources by providing a single common protocol and API, and it also supports a uniform and flexible interface to local job scheduling systems. The Grid Security Infrastructure (GSI) provides mutual authentication of both user and remote resources using GSI (Grid-wide) PKI-based identities. WS GRAM provides a simple authorization mechanism based on GSI identities and a mechanism to map GSI identities to local user accounts.

GridFTP is a high-performance, secure, reliable data transfer protocol based upon the FTP protocol and optimized for high-bandwidth wide-area networks. The current GridFTP uses GSI security on both control (command) and data channels. Other features include striping, third-party transfers between two servers, partial file transfers, reliability/restart, large file support, reusable data channels, parallel transfers, TCP buffer size control, and command pipelining.

### 2.1.2 EGEE

The EGEE project [36] aims to provide researchers in academia and industry with access to major computing resources, independent of their geographic location. The EGEE project will also focus on attracting a wide range of new users to the Grid.

The project will primarily concentrate on three core areas:

- The first area is to build a consistent, robust and secure Grid network that will attract additional computing resources.
- The second area is to continuously improve and maintain the middleware in order to deliver a reliable service to users.
- The third area is to attract new users from industry as well as science and ensure they receive the high standard of training and support they need.

The EGEE Grid will be built on the EU Research Network GÉANT and exploit Grid expertise generated by many EU, national and international Grid projects to date.

Funded by the European Commission, the EGEE project community has been divided into 12 partner federations, consisting of over 70 contractors and over 30 non-contacting participants covering a wide-range of both scientific and industrial applications.

### 2.1.3 CrossGrid

The CrossGrid project [52] is divided into four work packages which deal with the technical aspects of the project, and one work package dealing with management, dissemination, and exploitation:

- WP1 CrossGrid Application Development,
- WP2 Grid Application Programming Environments develops, integrates and tests tools that facilitate the development and tuning of parallel distributed, interactive applications on the Grid.

- WP3 New Grid Services and Tools develops the new, generic CG services and software infrastructure to support the Grid users, applications and tools as defined in the work packages WP1 and WP2.
- WP4 International Tested Organization collects all of the developments from the work packages WP1-3 and integrates them into successive software releases. It also gathers and transmits all feedback from the end-to-end application experiments back to the developers, thereby linking development, testing, and user experience.
- WP5 Project Management ensures the professional management of the project and active dissemination and exploitation of its results.

The local WP web sites deliver not only the general descriptions of their tasks but also the current status of all tasks within the given WP.

All the technical tasks are coupled and they form one logical structure, with applications at one end and the infrastructure at the other, and with intermediate programming environment and software tools in-between. The whole project is factorized into a number of individual tasks, each of which has one or a few institutions attached. The technical and managerial responsibilities for the four work packages lie primarily with their leaders, who are the top experts in their fields, and who originate from the principal contractors of the Consortium. It is up to these managers to prepare the work plan for these work packages and to report periodically on the progress (deliverables). All these tasks have to be synchronized by the central management.

## 2.2 System Development Tools

### 2.2.1 Java CoG Kit

The Java CoG (Commodity of Grid) Kit [2, 31] provides access to Grid services through the Java higher-level framework. Components providing client and limited

server side capabilities are included. The Java CoG Kit provides a framework for utilizing the many Globus services as part of the Globus metacomputing toolkit. Many of the classes are provided as pure Java implementations. Thus, writing client-side applets without installing the Globus toolkit is possible.

Java CoG (Commodity of Grid) Kit combines Java technology with Grid Computing to develop advanced Grid Services and accessibility to basic Globus resources. It allows easier and more rapid application development by encouraging collaborative code reuse and avoiding duplication of effort among problem-solving environments, science portals, Grid middleware, and collaborative pilots.

The Java-based Application uses the Java CoG kit to connect to the Grid system. Key characteristics include: GridProxyInit, a Java class for submitting pass phrases to Grid to extend certificate expiration dates, GridConfigureDialog, which uses the UITool in the CoG Kit to enable users to configure process numbers and host names of Grid servers, and GridJob, which creates GramJob instances. This class represents a simple gram job and allows for submitting jobs to a gatekeeper, canceling them, sending signal commands, and registering and unregistering from callbacks. GetRSL, RSL (Resource Specification Language) provides a common interchange language to describe resources.

### 2.2.2 Parallel Application Development Tool

MPI [4] is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementers, and users. MPICH [42-43] is a freely available, portable implementation of MPI. MPICH-G2 [44] is a Grid-enabled implementation of the MPI v1.1 standard. That is, it uses services provided by the Globus Toolkit (e.g., job startup, security). MPICH-G2 enables coupling of multiple machines, potentially with different architectures, to run MPI applications. MPICH-G2

automatically converts data in messages sent between machines of different architectures, and supports multi-protocol communication by automatically selecting TCP for inter-machine messaging and, where available, vendor-supplied MPI for intra-machine messaging. Existing parallel programs written for MPI can be executed over the Globus infrastructure after just recompilation.

### 2.2.3 Machine Information Provider

The Ganglia [21, 22, 30] is an open source project grew out of the University of California, Berkeley's Millennium initiative. The Ganglia is a scalable distributed system for monitoring status of nodes (processor collections) in wide-area systems based on Grid or clusters. It adopts a hierarchical; tree-like communication structure among its components in order to accommodate information from large arbitrary collections of multiple Grid or clusters. The information collected by the Ganglia monitor includes hardware and system information, such as processor type, CPU load, memory usage, disk usage, operating system information, and other static/dynamic scheduler-specific details.

### 2.2.4 Network Information Provider

The NWS (Network Weather Service) [12, 45] is a distributed system that detects network status by periodically monitoring and dynamically forecasting over a given time interval. The service operates a distributed set of performance sensors (network monitors, CPU monitors, etc.) from which it gathers system condition information. It then uses numerical models to generate forecasts of what the conditions will be for a given time period. The system includes sensors for end-to-end TCP/IP performance (bandwidth and latency), available CPU percentage, and available non-paged memory. The sensor interface, however, allows new internal sensors to be configured into the system.

### 2.2.5 Performance Benchmarking

The HPL [51] is a software package of solving a dense linear system in double precision arithmetic, it can be thought of a portable implementation of the High Performance Computing Linpack Benchmark for distributed-memory computers. The HPL provides timing and testing programs to quantify the precision of the acquired solution and the time it starts computation. The best performance reachable by this software depends on a large variation of factors. Nonetheless, with some restriction from the interconnection network, the algorithm and its attached scalable implementation in the sense that their parallel efficiency is maintained constant relate to the memory usage of each processor.

### 2.2.6 JRobin

JRobin [49] is a 100% pure java implementation of RRDTool's functionality. It follows the same logic and uses the same data sources, archive types and definitions as RRDTool does. JRobin supports all standard operations on Round Robin Database (RRD) files: CREATE, UPDATE, FETCH, LAST, DUMP, XPORT and GRAPH. JRobin's API is made for those who are familiar with RRDTool's concepts and logic, but prefer to work with pure java. If you provide the same data to RRDTool and JRobin, you will get exactly the same results and graphs. JRobin is made from the scratch and it uses very limited portions of RRDTool's original source code. JRobin does not use native functions and libraries, has no Runtime.exec() calls and does not require RRDTool to be present. JRobin is distributed as a software library (jar files) and comes with full java source code

### 2.2.7 JFreeChart

JFreeChart [50] is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature

set includes: a consistent and well-documented API, supporting a wide range of chart types; a flexible design that is easy to extend, and targets both server-side and client-side applications; support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG); JFreeChart is "open source" or, more specifically, free software. It is distributed under the terms of the GNU Lesser General Public Licence (LGPL), which permits use in proprietary applications.

## 2.3 Related Works

In [24], authors present an integrated Grid information system [3] which is designed and implemented using existing information systems. As a result, the proposed system is able to monitor the basic information on each computer, the network state, computing state of a cluster, the nodes and queues of a cluster, and more using a Globus toolkit, NWS, Condor, and LSF. In addition, since the integrated Grid information system allows for the discovery and monitoring of all Grid resources, the usability of the system can be proliferated.

In [19], authors propose our enhanced multi-site resource selection algorithm—CGRS algorithm, based on the distributed computational grid model and the grid scheduling model. The CGRS algorithm integrates a new density-based grid resource-clustering algorithm into the decoupled scheduling approach of the GrADS and decreases its time complexity. Also, we establish a performance model and mapping strategy for the synchronous iterative applications and demonstrate the correctness and effectiveness of CGRS algorithm in our simulation

In [36], authors present a tool able to check if a given grid service works as expected for a given user or set of users on the different resources available on a grid. Our solution deals with the grid services as single components that should produce an

expected output to a pre-defined input, what is quite similar to unit testing. Our tool, called Service Availability Monitoring or SAM, is being currently used to monitor some of the largest (maybe the largest) production grids available today.

In [52], the CrossGrid project addresses realistic problems in medicine, environmental protection, flood prediction, and physics analysis and is oriented towards specific end-users: Medical doctors, who could obtain new tools to help them to obtain correct diagnoses and to guide them during operations; industries, that could be advised on the best timing for some critical operations involving risk of pollution; flood crisis teams, that could predict the risk of a flood on the basis of historical records and actual hydrological and meteorological data; physicists, who could optimize the analysis of massive volumes of data distributed across countries and continents. Corresponding applications will be based on Grid technology and could be complex and difficult to use: the CrossGrid project aims at developing several tools that will make the Grid more friendly for average users. Portals for specific applications will be designed, that should allow for easy connection to the Grid, create a customized work environment, and provide users with all necessary information to get their job done.

# Chapter 3

# System Design and Implementation

## 3.1 Resource Broker

This study implements a resource broker with the cross grid information service and cross grid authentication service on the multi-grid computational environment. The resource broker discovers and evaluates multi-grid resources, and makes the decision of job submission depending on the job requirement. The system architecture of the resource broker is shown in Figure 3-1. Users could easily make use of our resource broker through a common Grid portal [6, 7, 13-17]. The primary task of Resource Broker is to compare requests of users and resource information provided by Information Service. After choosing the appropriate job assignment scheme, Grid resources are assigned and the Scheduler is responsible to submit the job. The results are collected and returned to Resource Broker. Then, Resource Broker records results of execution in the database of Information Center through the Agent of Information Service. The user can query the results from Grid portal.

Our resource broker architecture include the four layers, web portal, resource brokering subsystem, multi-grid manager center, and multi-grid resource. The multi-grid resource consists of many single grid environments. When a new single grid joins our multi-grid environment, the grid administrator should register at the web portal and provide the personal file and the related information of the grid, such as the CA package, the grid-mapfile of Globus, and the machine lists. The multi-grid system deploys the environment according to the registered information, and then assigns a resource broker account to a single grid user. This account could access the resource supported by the resource broker on the multi-grid environment, Figure 3-1 shows the concept. The multi-grid manager center has cross grid authentication service and cross

grid information service, that is important part on entire multi-grid environment, which control the access of grid and the information gather, the detail described in following section.

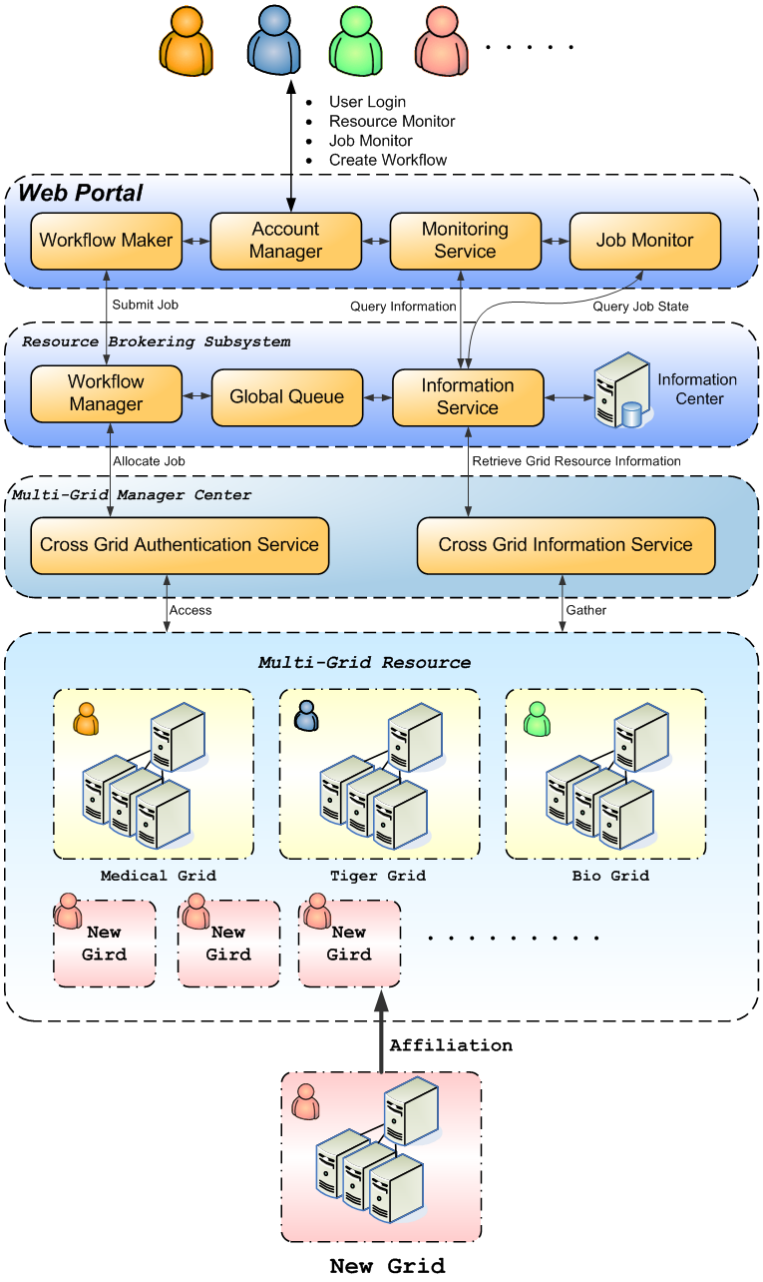


**Figure 3-1.** Resource Broker system architecture

## 3.2 Software Stack Diagram

The system software stack includes three layers constructed using a bottom-up methodology. The layers are described below:

- Bottom Layer: principally consists of Nodes, as shown in Figure 3-2. The layer contains two main blocks, the Information Provider, which uses Ganglia to gather machine information on Nodes, such as number of processors/cores, processor loading, total/free memory, and disk usage, and NWS, which gathers essential network information such as bandwidth and latency. The second block contains Grid Middleware, used to join Grid Nodes together, and the MPICH-G2 required for running parallel applications on the Grid.
- Middle Layer: contains grids as shown in Figure 3-3. Each grid consists of several nodes located in the same place or connected to the same switch/hub. All nodes in a site are connected to each other and to the Internet. Moreover, grids are usually built as clusters with each node having a real IP.
- Top Layer: contains the Resource Broker, the Monitoring Service, and the multi-grid manager, as shown in Figure 3-3. The Resource Broker coordinates Grid resources, dispatches jobs to resources, and monitors job statuses. One contribution of this work is proposing a resource allocation scheme that can handle user jobs requiring more Grid resources than one dedicated cluster can supply. To make strategic decisions in dispatching jobs the Re-source Broker needs fresh information on the Grid from the Monitoring Service, which also provides a web front-end for users to observe job progress. Monitoring Service, which also provides a web front-end for users to observe job progress. The task of multi-grid manager

is integrating the grids into the multi-grid environment, which include the cross grid authentication service and cross grid information service.

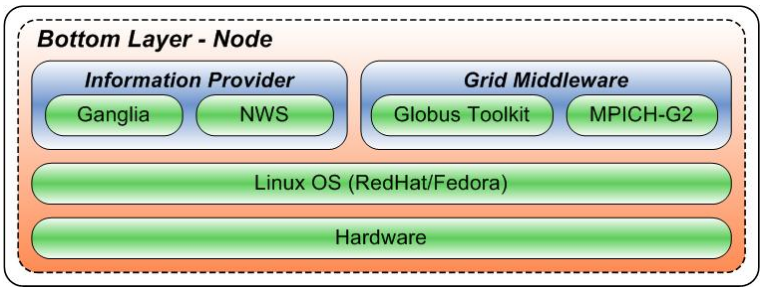


**Figure 3-2.** The software stack of all nodes

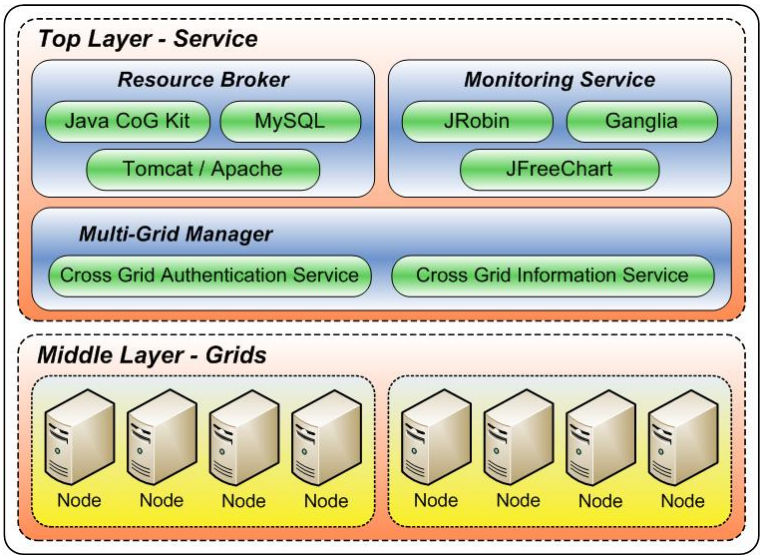


**Figure 3-3.** The software stack of all sites and the service

## 3.3 Cross Grid Authentication Service

The Globus Toolkit authentication is issued by the certificate of GSI. Each user and service is certificated to identify and authenticate the trust between users or services [32]. If two parties have the certificate and both of them trust the CAs, then these two parties could trust each other. This is known as the mutual authentication. A subject name, which identifies the person or object that the certificate represents. The cross

grid authentication service manages the several certificates and the subject of grid, and we know these messages from the registered information via web portal. All of nodes on multi-grid environment should setup the cross grid service tool, which written in the shell script program, show in

Figure **3-4**. The cross grid service tool contain some procedure to regular automatic update with multi-grid manager center, such as the IP and domain of host list, the certificates, and the subject. Any single grid user on our multi-grid environment makes use of the multi-grid resource through the tool to update with multi-grid manager center. The primary task of cross grid information is to gather the IDL file, which contains the several attribute of host on grid, the detail describe in next section.

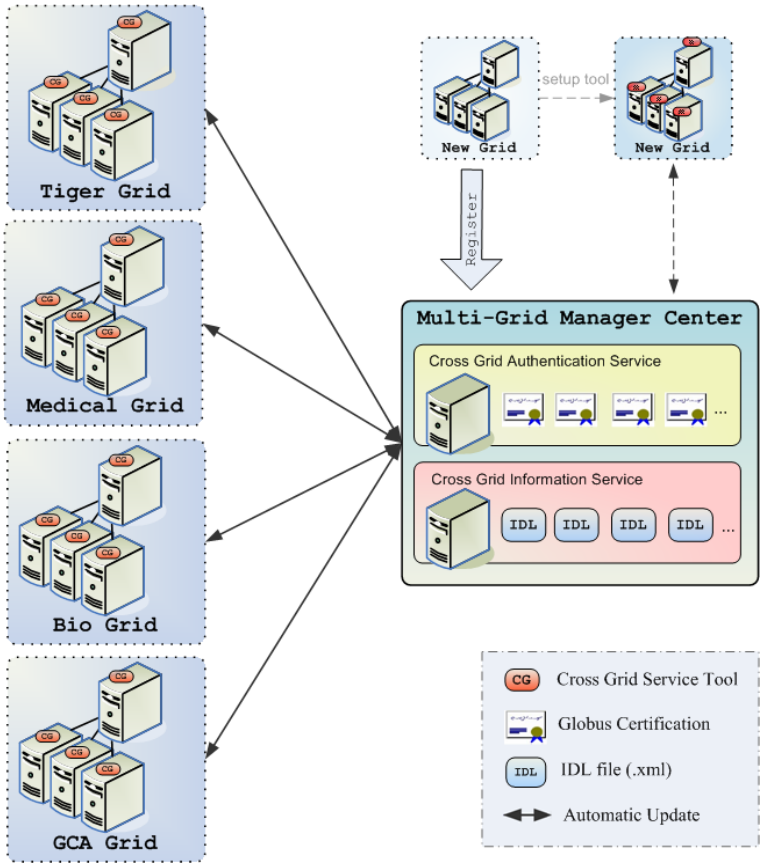


**Figure 3-4.** Multi-Grid Manager Center

## 3.4 Cross Grid Information Service

The monitor tool is the important component in a grid environment [8-12]. A monitor tool could help grid administrator to manage and monitor the machines. On the other hand, the middleware could gather the resource information to find the suitable resource by the monitor tool. There are common monitor tools, such as MDS, Ganglia, Cacti, and Condor [41, 43, 44]. On a multi-grid environment, grids use different monitor tools with different information formats. Therefore, we define a new information format to translate different formats. The proposed format is called IDL (Information Describing Language), as shown in Figure 3-5. It is responsible to exchange and translate resource information among grids, and to perform the post-transfer filtering to ensure that only necessary information is passed to clients, end users, and software components.


**Figure 3-5.** Information Describing Language.

The CGIS consists of three layers: the Core Service Layer, the Translator Layer, and the Resource Layer, as shown in Figure 3-6. The Core Service Layer contains the

Agent, Filter, Getter and Setter, and Gather. These components are installed in every grid environment for information gathering and maintenance. The Translator Layer supports a variety of format conversion monitor tool, such as Ganglia, Cacti, and MDS. The information in different monitor tools is transformed into the IDL format. The Resource Layer describes resource information from different grid environments. In this study, it includes Tiger Grid, Medical Grid, and Bio Grid.

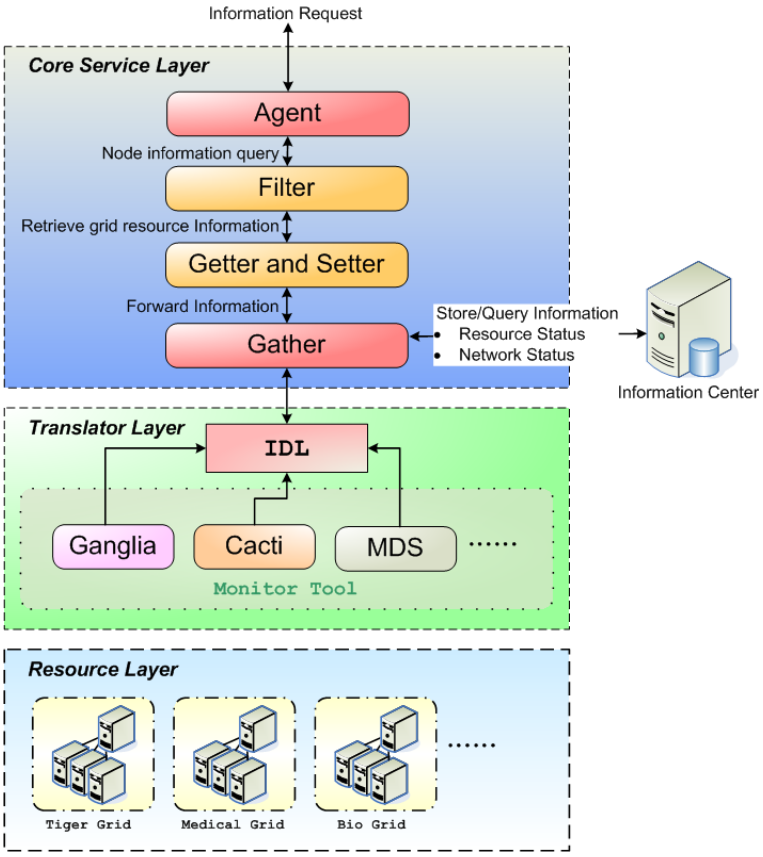


**Figure 3-6.** Cross Grid Information Service architecture

## 3.5 Web Portal

We provides a registration page let a new single grid join our multi-grid environment, the register service will automatic send mail to the multi-grid administrator, administrator adjustment and configuration the setting of multi-grid environment when

receive the registration page. Later, notify the new single grid download the cross grid service tool from multi-grid server achieve the step of registration, as shown in Figure 3-7.

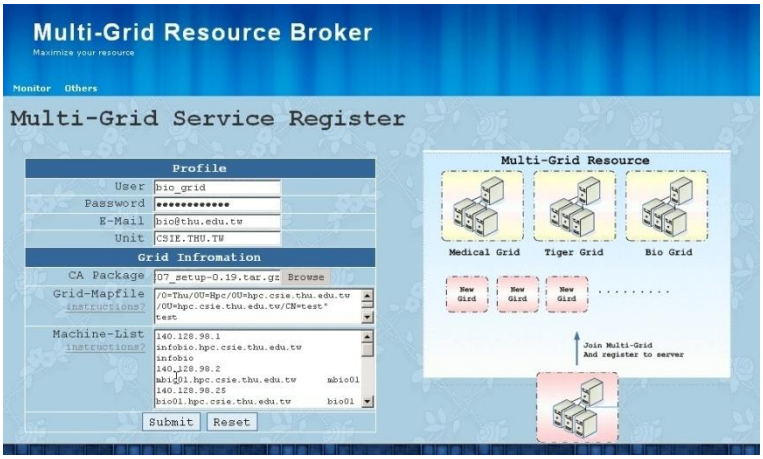


**Figure 3-7.** Registration page of Multi-Grid Resource Broker

We use IDL XML format to exchange the information of multi-grid resource. The collected information not only utilized when submitting job, but also show the status of grids. The Monitor Service could monitor the resources supported by the multi-grid resource broker. We provide the interface to observe the status of HostInfo, Daemons, and Ganglia. Hosts Info show the status include the number of CPU, CPU speed, the average of load in one minute, the average of load in five minute, the size of free memory, the size of free swap , as shown in Figure 3-8. Daemon Status is another service to monitor the services status of machines by the NMAP, which parse the list of host by IDL file and scan the fix port through the list, then get the status whether the service to be on, as shown in Figure 3-9. Ganglia only collect the status of grid resources. In this study, we rewrite the Ganglia codes and modify the setting to support the multiple grids. Our Ganglia page has the multiple level architecture, that show

Tiger Grid, Medical Grid, Bio Grid, and GCA Grid in the web portal, as shown in Figure 3-10.

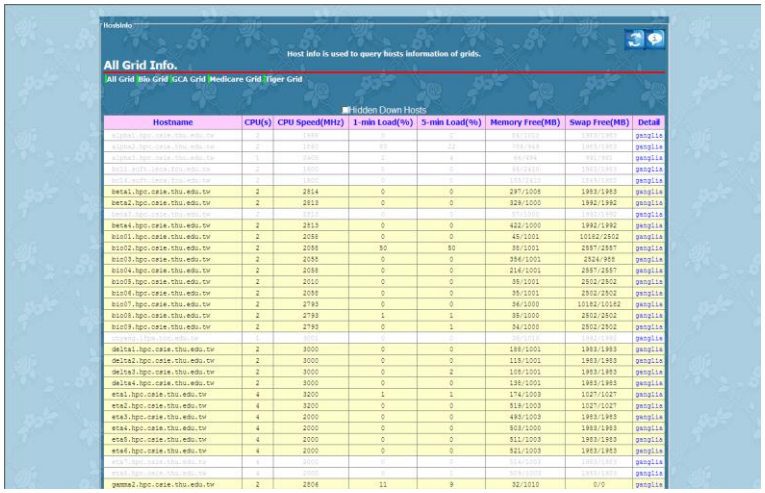


**Figure 3-8.** Hosts Info

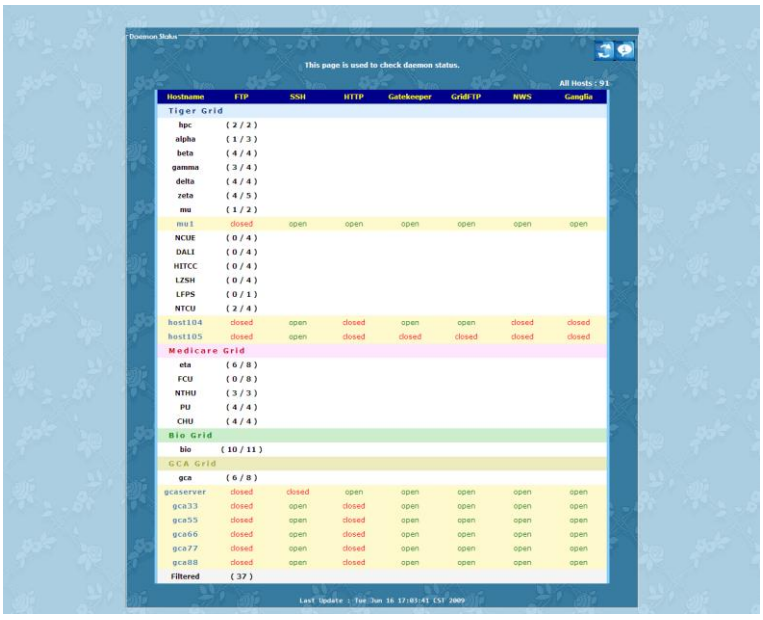


**Figure 3-9.** Daemon Status


**Figure 3-10.** Ganglia

The Tiger Grid have the network measure service, we use the NWS achieve the point to point network bandwidth measure. But the NWS tool lacks the database to conserve the statistics, so we stored the statistics into round-robin database every five minutes, and then draw the graph through JRobin, as shown in Figure 3-11. JRobin is a 100% pure java implementation of RRDTool's functionality. RRD stands for Round Robin Database, and it is what it sounds like. There are a fixed number of records in the database and once the last record has been written in the database the next update goes to the first record, and around and around it goes. In this way, your databases will never grow out of control. The only downside to this is that, obviously, you've got to know how much data you'll want to look at historically ahead of time so that when you

generate your database you have enough data. You may want a day's worth of info, or even months. The main functions are listed in the following:

1. Define and create an initial round-robin database in the server, the database must need setting the database name, start time, timestamp, data sources and one or more round robin archives.

```
public void createRrd(String clusters[],String rrdFile,long startTime)
throws RrdException, IOException{
    /* Define RRD */
    RrdDef rrdDef = new RrdDef(+rrdFile+".rrd", startTime-1, 300);

    /* Define the data source */
    for (int i=0;i<clusters.length;i++)
        rrdDef.addDatasource(clusters[i], "GAUGE", 300, Double.NaN,
Double.NaN);

    /* Define the round robin archives */
    rrdDef.addArchive("AVERAGE", 0.5, 1, 288);  //300 sec
    rrdDef.addArchive("AVERAGE", 0.5, 7, 288);  //35 min
    rrdDef.addArchive("AVERAGE", 0.5, 30, 288); //2.5 hr
    rrdDef.addArchive("MAX", 0.5, 1, 288);
    rrdDef.addArchive("MAX", 0.5, 7, 288);
    rrdDef.addArchive("MAX", 0.5, 30, 288);

    /* Create a RRD file */
    RrdDb rrdDb = new RrdDb(rrdDef);
    rrdDb.close();
  }
```

2. Measure the network bandwidth every five minutes; moreover store the statistics in the round robin databases.

```
public void update(String[] args) throws IOException, RrdException{
    /* Open the data source */
    RrdDb rrdDb = new RrdDb(args[1]+".rrd");
    Sample sample = rrdDb.createSample();
```

```
    try{
        sample.setTime(currentTime);    /* update time */
        for (int i=2;i<args.length;i+=2)
            sample.setValue(args[i],Long.parseLong(args[i+1]));
        sample.update();
        rrdDb.close();
    }catch(Exception e){
        if (e.toString().contains("NumberFormat"))
            System.out.println("");
        else
            System.out.println("Exception: "+e);
    }
}
```

3. We want to get the network diagram, so draw the graph from the data source of round robin database and appropriate setting the attribute of graph, such as diagram name, the diagram size, and the comment.

```
public void drawer(String cluster,String clusters[],String timeType,long
startTime,long endTime) throws RrdException, IOException{
    String rrdFile=cluster+".rrd";
    String pngFile=timeType+"/"+cluster+".png";
    String dataformat="";

    /* define graph */
    RrdGraphDef gDef = new RrdGraphDef();
    gDef.setTimePeriod(startTime-600, endTime);
    gDef.setDefaultFont(new Font("Courier New", Font.PLAIN, 12));
    gDef.setTitleFont(new Font("Courier New", Font.BOLD, 16));
    gDef.setTitle("Network Flow: "+cluster+"     last "+timeType);
    gDef.setVerticalLabel("Mbit per second");
    gDef.comment("\n          Now          Average        Max\n");

    /* draw  */
    for (int i=0;i<clusters.length;i++){
        gDef.datasource(clusters[i], rrdFile, clusters[i], "AVERAGE");

        String space1="",space2="";
```

```
            for (int j=0;j<5-cluster.length();j++)
                space1+=" ";
            for (int j=0;j<5-clusters[i].length();j++)
                space2+=" ";
            String line=space1+cluster+" <--> "+space2+clusters[i];

            /* draw the lines */
            gDef.line(clusters[i], Color.decode(colors[i*2]), line);
            gDef.comment("\t");

            /* print the statistics */
            dataformat="@8.2 Mb";   /* Alignment */
            gDef.gprint(clusters[i], "LAST",dataformat);
            gDef.comment("\t");
            gDef.gprint(clusters[i], "AVERAGE",dataformat);
            gDef.comment("\t");
            gDef.gprint(clusters[i], "MAX",dataformat);
            gDef.comment("\n");
        }

        /* comment */
        gDef.comment("\nLatest Update: "+new SimpleDateFormat("yyyy/MM/dd
HH:mm").format(new Date(new Timestamp(endTime*1000).getTime()))+"@r");

        RrdGraph graph = new RrdGraph(gDef);
        graph.saveAsPNG(pngFile, 380, 100); /* diagram size */

        /* print the time of latest update */
        if (timeType.equals("hour")){
            try {
                BufferedWriter out = new BufferedWriter(new
FileWriter("networkinfo/bandwidth/draw_time"));
                out.write(String.valueOf(endTime));
                out.close();
            } catch (IOException e) {
            }
        }
    }
```


**Figure 3-11.** NWS Information

The function pages have a web page of help manual in our web portal, which describe the capability of page and the function of hyperlinks, and we hope any user can use the web page easily without too much trouble on multi-grid environment, as shown in Figure 3-12.


**Figure 3-12.** Help Page

The user could execute a parallel program via the resource broker and choose the required for the execution of the host after the use submit the job, as shown in Figure 3-13 and Figure 3-14, and get the result form the Job Monitor Service, which show the many detail messages in execute time, such as scheduling log, machine list, result message, running log, debug message, and turnaround time, as shown in Figure 3-15.


**Figure 3-13.** Job Monitor

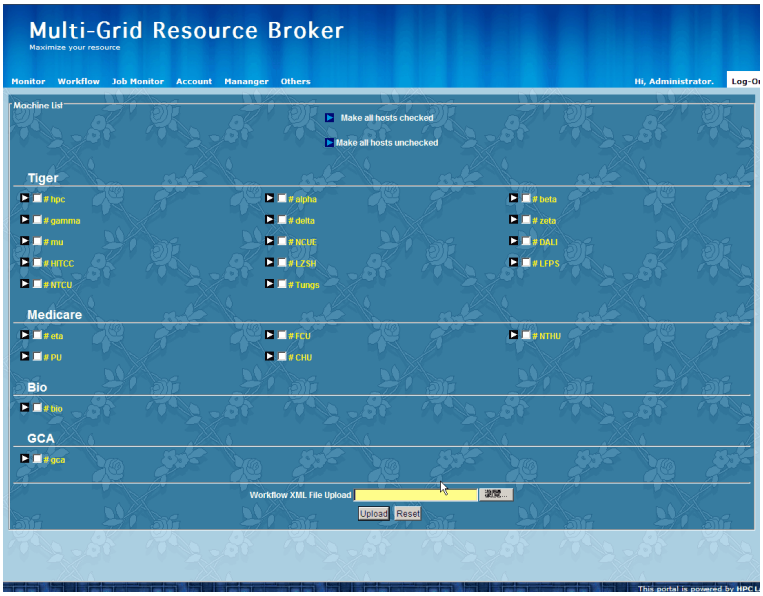


**Figure 3-14.** Resource Selection

We enhance the graphical display of monitor function, uses JFreechart tool draw more meaningful graphics, include Job Monitor, Login Info, and Utilization of Resource, as shown in Figure 3-15, Figure 3-16 and Figure 3-17. JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications.

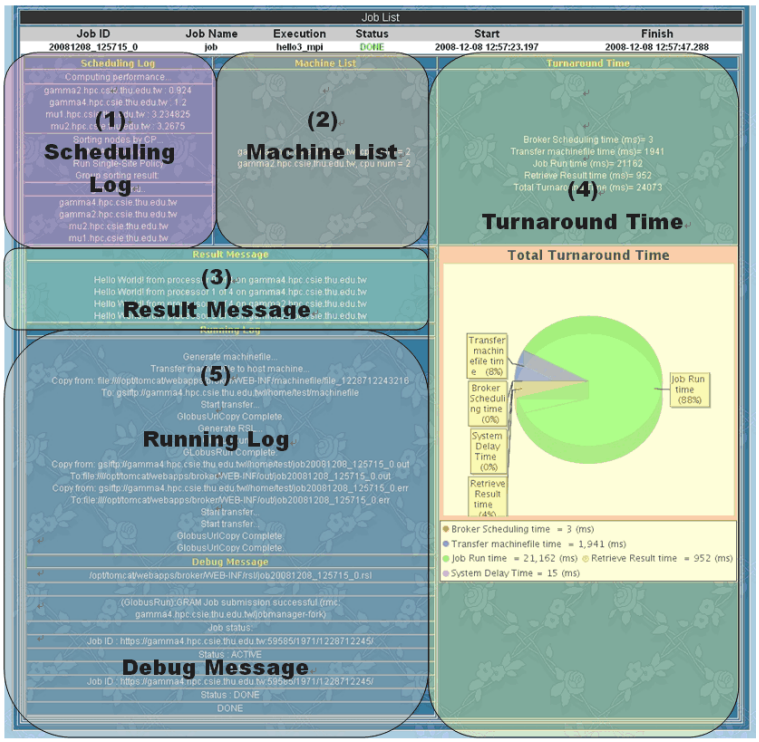


**Figure 3-15.** Job Monitor

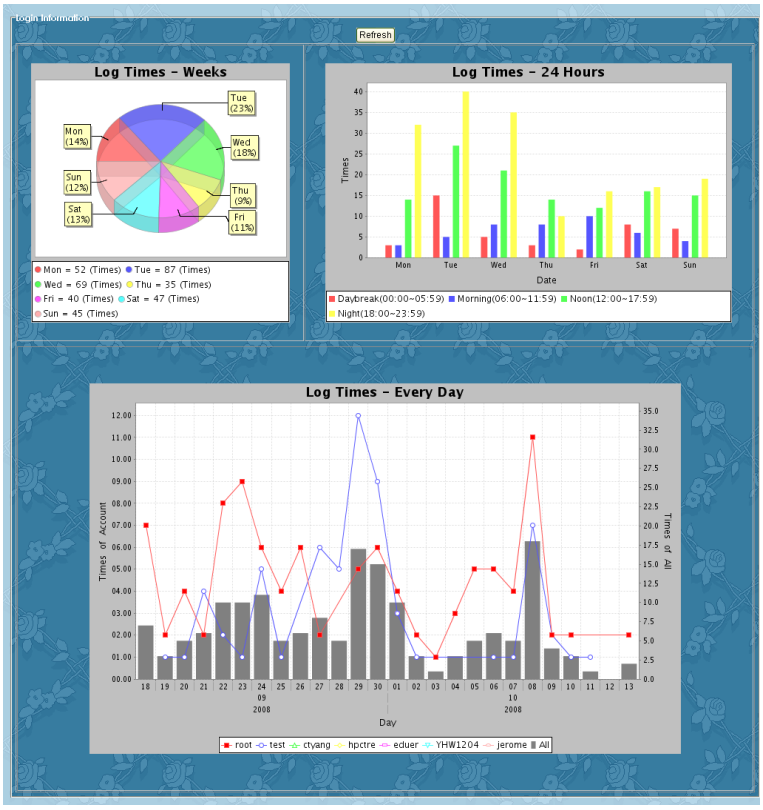


**Figure 3-16.** Login Information

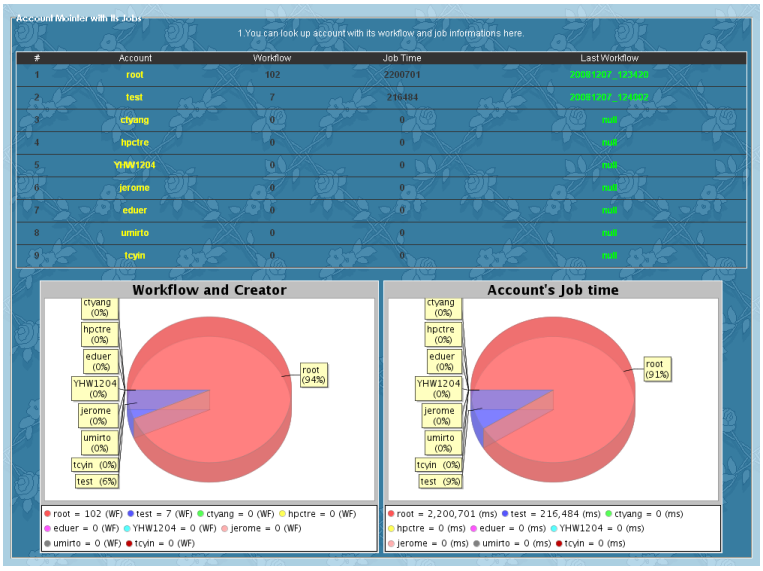


**Figure 3-17.** Utilization of Resources

# Chapter 4

# Multi-Grid Resource Selection Strategy

## 4.1 Parameters and MGRSS Algorithm

In this section, the parameters used in the algorithm are listed and explained in the following:

- $job_i$: The $i^{th}$ job dequeued from the job queue, where $i$ = 1~$n$. The job contains some information such as job name, program name, program arguments, input data, and number of processors to run on. The program is usually a parallel program written by MPI library and compiled by MPICH-G2 that is a Grid-enabled implementation of the MPI standard. The Resource Broker will allocate resources according to the information provided by the job.
- $NP_{mGrid}$ : The total number of processors on the multiple grid
- $NP_{lGrid}$ : The total number of processors on the local grid
- $NP_{mf}$ : The total number of processors with idle status on the multiple grid
- $NP_{lf}$ : The total number of processors with idle status in the local grid
- $NP_{max}$ : The maximum processors on the multiple grid
- $NP_{valid}$ : The available processors on the multiple grid
- $NP_{req}$: The processors used for executing $job_i$, if the Resource Broker dispatches the $job_i$ successfully, the resources distributed over several Nodes or Sites used for $job_i$ will be locked for a while until the $job_i$ finishes.
- $S_{max}$ : The higher scores of the machines
- $NP_{valid}$ : How many number of processor user can apply.

$$NP_{vaild} = \alpha \cdot \frac{NP_{mGrid}}{NP_{\max}} \cdot NP_{lGrid} \quad (1)$$

- $Load_i$: It is a sum of the $Load_{1min}$, $Load_{5min}$, and $Load_{15min}$ of $Node_i$, the expression of $Load_i$ is shown as follows:

$$Load_i = \frac{load_{1min} \times 3 + load_{5min} \times 2 + load_{15min}}{6} \quad (2)$$

Where $Load_{1min}$ is the average of load in one minute, $Load_{15min}$ is the average of load in five minute, and $Load_{15min}$ is the average of load in fifteen minute. For each parameter has different proportion value, the time is closer, the value is higher.

- $PE_i$ : The computing performance efficiency of host, the expression is shown as follows:

$$PE_i = HPL_i \times \frac{100 - Load_i}{100} \quad (3)$$

Where $HPL_i$ is the benchmarking value of host obtained by the approach of benchmarking.

- $Flow_i$ : The average network flow of host, the expression is shown as follows:

$$Flow_i = \frac{\sum_{k=1}^{N}(Byte\_In_k + Byte\_Out_k)}{N * 1024^2} \quad (4)$$

Where *Byte_in* and *Byte_out* get from the IDL, estimated the network flow influence through these two parameters. We also considered that same domain of other machines will affect the flow of the entire site, where *N* is the number of nodes in same domain. Therefore we use the average of all nodes in same domain. Finally, the $Flow_i$ mean Megabyte per second after divided by square of 1024.

- $TP_i$ : Total performance power, the expression is shown as follows:

$$TP_i = PE_i - \beta \cdot Flow_i \quad (5)$$

Where $\beta$ is the effect ratio used to regulate the percentage of *PE* and *Flow*.

The main policy of MRGSS, a single grid select resource itself when that possesses sufficient resources, otherwise selects the multiple grid resource. The number of available resources is variation, which according to their own resources of local grid. The procedure of MGRSS is listed as follows:

```
1  MGRSS(job_i) {
2      NP_mGrid = getNumFromIDL();
3      NP_lGrid = getNumFromIDL();
4      NP_mf = getFreeHostFromIDL();
5      NP_lf = getFreeHostFromIDL();
6      NP_max = getMaxNumber( NP_mGrid );
7      NP_valid = getValidNumberProcess();
8      if (NP_req,i < NP_lGird) then              /* Single Grid */
9          if (NP_req,i < NP_lf)
10             S_max = getMaxRankCandidate(NP_lf);
11             submitJob(job_i, S_max);
12         else
13             enqueue (job_i);
14     else
15         if (NP_req,i < NP_valid) then          /* Multi Grid */
16             if (NP_req,i < NP_mf)
17                 S_max = getMaxRankCandidate(NP_mf);
18                 submitJob(job_i, S_max);
19             else
20                 enqueue (job_i);
21         else
22             discard job_i;
23     }
```

## 4.2 MGRSS Flowchart

Submit job is the most important part of Resource Broker. Among the many resources, chooses the appropriate resource to do assignment is a very important topic, beacuse the fine resource selection strategy can shorten the execution time of job, and upgrade the

overall performance on Multi-Grid environment. For above purposes, we provide a Multi-Grid Resource Selection Strategy, called MGRSS, as shown in Figure 4-1.

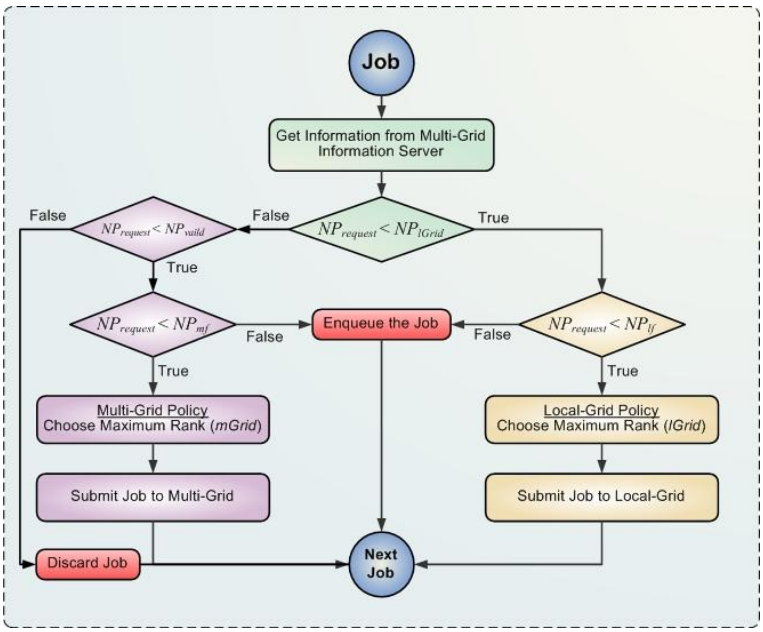


**Figure 4-1.** Multi-Grid resource selection strategy flowchart

First a job dequeued from the job queue for scheduling. Second, it gets the latest information from an IDL file of multi-grid information server determining if the number of processors of request is smaller than the total number of processors of Multi-Grid. If false, discard job. If true, determining use the single grid or multi-grid strategy by the processors of request. For single grid strategy, the processors of request should be smaller than the free processors of local grid; otherwise, enqueue the job and waiting for five minutes. When the waiting time is exceed the five minutes, resubmit the job. If the processors of request still smaller than the free processors of local grid, we adopt the multiple grid strategy. For multiple grid strategy, if the processors of request larger than the valid processors of multiple grid, then discard job. Otherwise if the processors of request smaller than free processors of multiple grid, then we adopt the multiple grid strategy.

# Chapter 5

# Experimental Environment and Results

## 5.1 Experimental Environment

In this work, the experiment was conducted and evaluated on the twenty two nodes showed in Table 5-1 by number of CPU/core, speed(MHz), memory(MB), and HPL(GFLOPA).

**Table 5-1.** The machine information of test-bed

| Host | Number of CPU/Core | Speed (MHz) | Memory (MB) | HPL (GFLOPA) |
|---|---|---|---|---|
| beta1 | 2 | 2,813 | 1,024 | 7.490 |
| beta2 | 2 | 2,813 | 1,024 | 7.490 |
| beta3 | 2 | 2,813 | 1,024 | 7.490 |
| beta4 | 2 | 2,813 | 1,024 | 7.490 |
| delta1 | 2 | 3,000 | 1,024 | 8.142 |
| delta2 | 2 | 3,000 | 1,024 | 8.142 |
| delta3 | 2 | 3,000 | 1,024 | 8.142 |
| delta4 | 2 | 3,000 | 1,024 | 8.142 |
| eta3 | 4 | 2,000 | 1,024 | 13.391 |
| eta4 | 4 | 2,000 | 1,024 | 13.391 |
| eta5 | 4 | 2,000 | 1,024 | 13.391 |
| eta6 | 4 | 2,000 | 1,024 | 13.391 |
| gamma2 | 2 | 2,806 | 1,024 | 7.125 |
| gamma3 | 2 | 2,806 | 1,024 | 7.125 |
| gamma4 | 2 | 2,806 | 1,024 | 7.125 |
| medicare | 4 | 1,793 | 2,048 | 12.652 |
| medicare1 | 4 | 1,000 | 2,048 | 10.176 |
| mu1 | 2 | 1,866 | 1,024 | 6.251 |
| mu2 | 2 | 1,866 | 1,024 | 6.251 |
| puthu01 | 1 | 1,991 | 1,024 | 4.058 |
| puthu02 | 1 | 1,991 | 1,024 | 4.058 |
| puthu03 | 1 | 1,991 | 1,024 | 4.058 |

## 5.2 Tested Programs

This experimental used several parallel programs. The programs and explanation are described in the following:

- bucketsort_mpi: This program sorts a list of evenly distributed numbers. The numbers are randomly generated and are evenly distributed in the range between 0 and 2n.
- pi_mpi: This program computes the value of PI by using numerical integration, and it spends a large amount of computing power.
- prime_mpi: This program computes the largest prime of the argument, and it spends a large amount of computing power.
- mmd_mpi: This program performs a square matrix multiplication and uses up the memory gradually when the argument, matrix size, glows up
- jacobi_mpi: This program solves Laplace equation by executing T steps of the smoothing part of the algorithm. It decomposes the matrix in strips that are assigned to processors.
- cfd_mpi: This program performs computational fluid dynamics (CFD) simulations. CFD is a computational technology that can use to simulate the dynamics of flow. The CFD outputs a prediction of the fluid dynamics of a computational model that represents a system or device by applying the fluid flow physics to this virtual prototype.
- nqueen_mpi: This program solves the *N*-Queen problem, which is required to place the *N* queens on the *N* by *N* chessboard such that no two queens attack each other, i.e., no two queens can be placed on the same row, the same column, and the same diagonal.

- sat3_mpi: This program solves the Circuit-Satisfiability problem. Given a Boolean combination circuit composed of AND, OR, and NOT gates, whether there is an input that makes the circuit output True.
- sieve5_mpi: This program uses the Sieve of Eratosthenes method for efficiently listing prime numbers. First, a list of integers beginning with 2 and ending with some number, say, *N*. Then remove all multiples of 2. Move to the next number, which in this case is 3, and then remove all its multiples. Continue in this fashion until there are no new numbers.

## 5.3 Experimental Results

In this section, we do the five experimental results. Among the first three experiment respectively measurement the execute time for different specific program which use the different metric in five strategy environment. In the fourth experiment, we execute the entire parallel programs which as previous section mention. The last experiment randomly selected twenty different types of parallel programs and uses the different number of processors to execute. The above-mentioned results are all order to compare the execution time between different resource selection strategies.

We calculates the value of performance and network for each machine, and according the score to sort the machines list which represent the resource selection priority, and previous chapter mention the MGRSS algorithm. We divided MGRSS into three levels from level one to level three by different weighted value $\beta$. The maximum weighted value of the network is MGRSS3（$\beta$ =0.7）; the medium weighted value（$\beta$ =0.5）;the maximum weighted value of the performance is MGRSS1（$\beta$ =0.3）.

First experiment show the result for bucketsort MPI programs total executes time in ten times, and the program parameter size using the 512, 1024, 4096 respectively. The bucketsort MPI program use a small number of resources. In this case, the rapid transmission normally will decide the time of program execute. The MGRSS2 and MGRSS3 have better strategies in this experiment, because that has more proportion of network in three level of MGRSS algorithm. Experiment result show in Figure 5-1 and Figure 5-2.

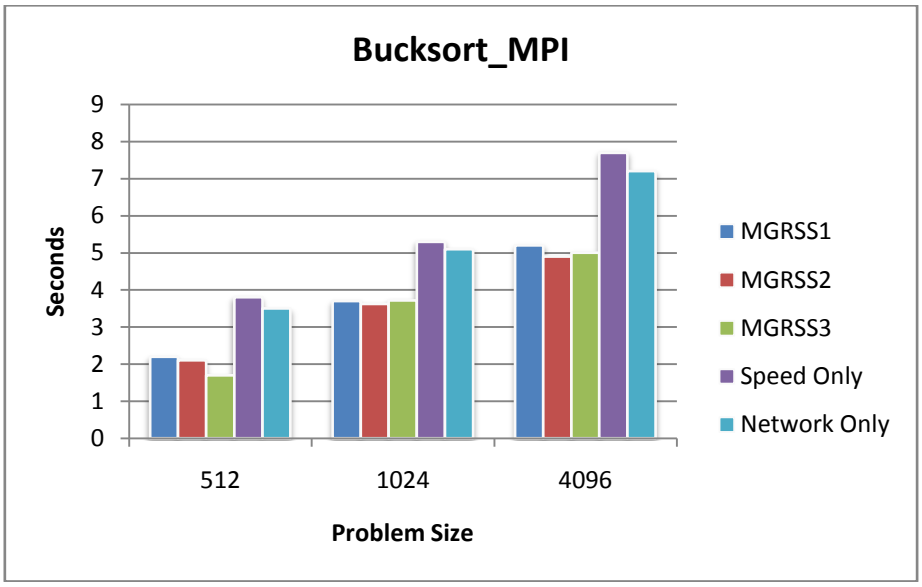


**Figure 5-1.** Result for Bucketsort_MPI in different parameters sequence

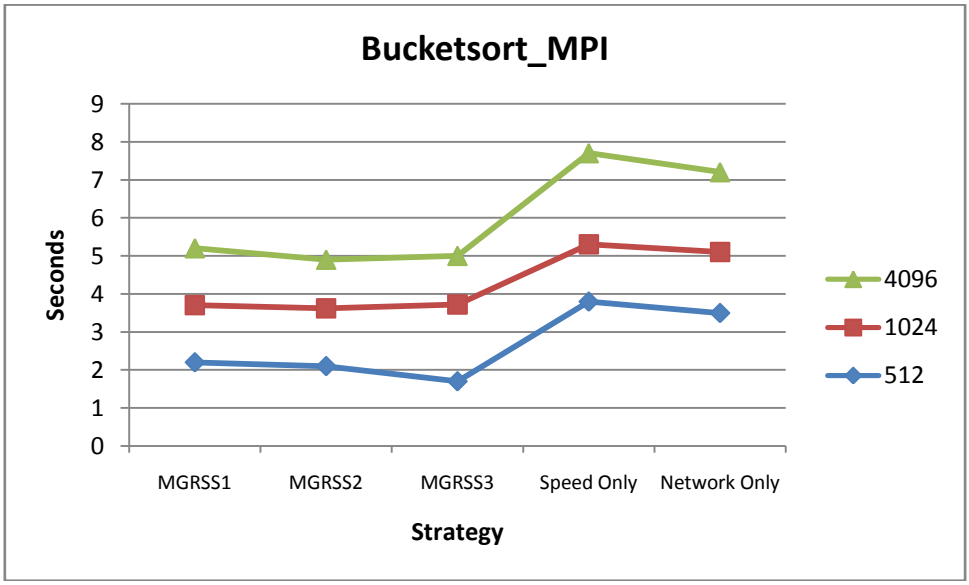


**Figure 5-2.** Result for Bucketsort_MPI in different strategies sequence

Second experiment shows the result for matrix multiplication MPI programs total execute time in ten times, and the program parameter size using the 256, 512, 1024 respectively. The matrix multiplication requires a large amount of computing power and network, but the network flow is still more proportion. So the MGRSS3 spent the less time executing the jobs. Experiment result show in Figure 5-3 and Figure 5-4.

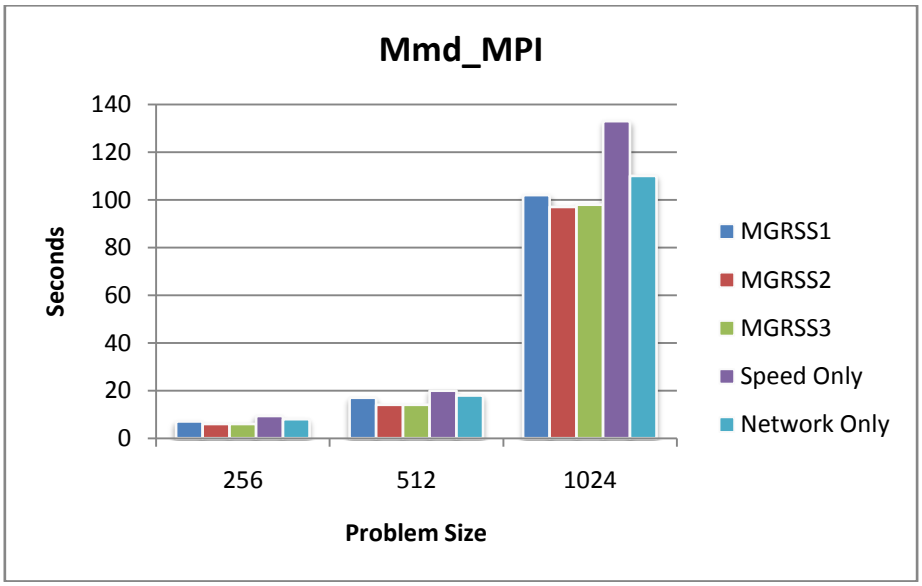


**Figure 5-3**. Result for Mmd_MPI in different parameters sequence

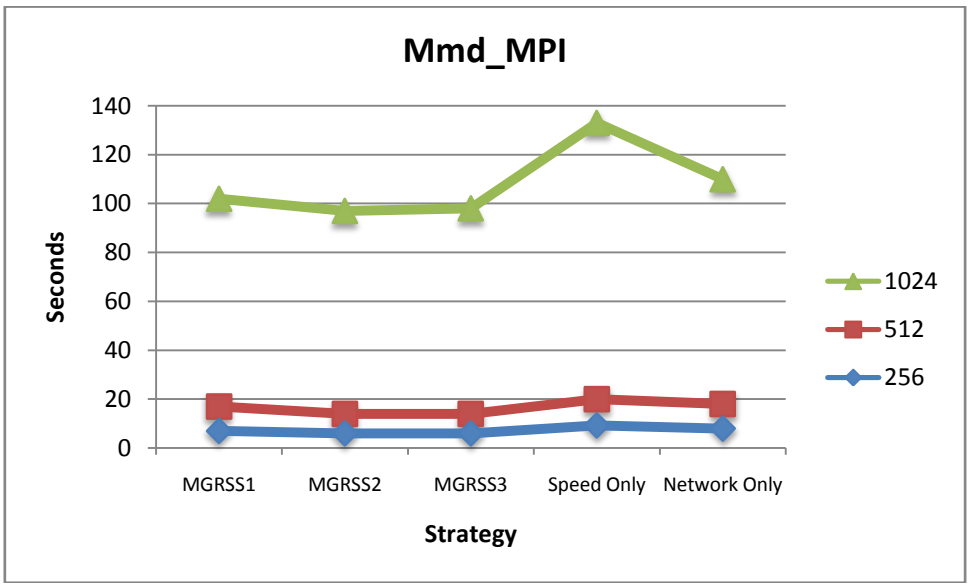


**Figure 5-4.** Result for Mmd _MPI in different strategies sequence

Third experiment shows the result for cfd problem MPI programs total execute time in ten times, and the program parameter size using the 80, 100, 120 respectively. This program uses a large of computing power and network flow, but compare with former experiment the usage of computing power is still greater than network flow, Experiment result show in Figure 5-5 and Figure 5-6.

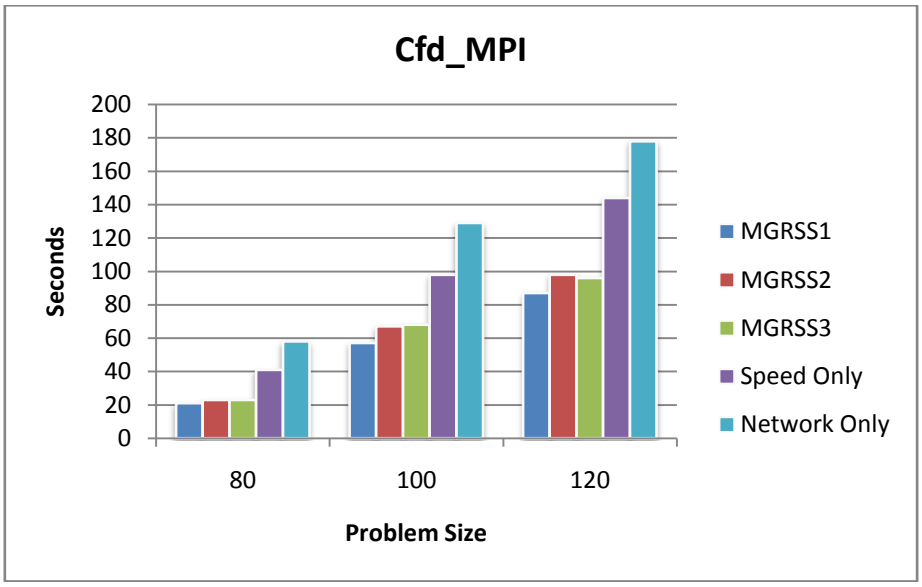


**Figure 5-5.** Result for Cfd_MPI in different parameters sequence

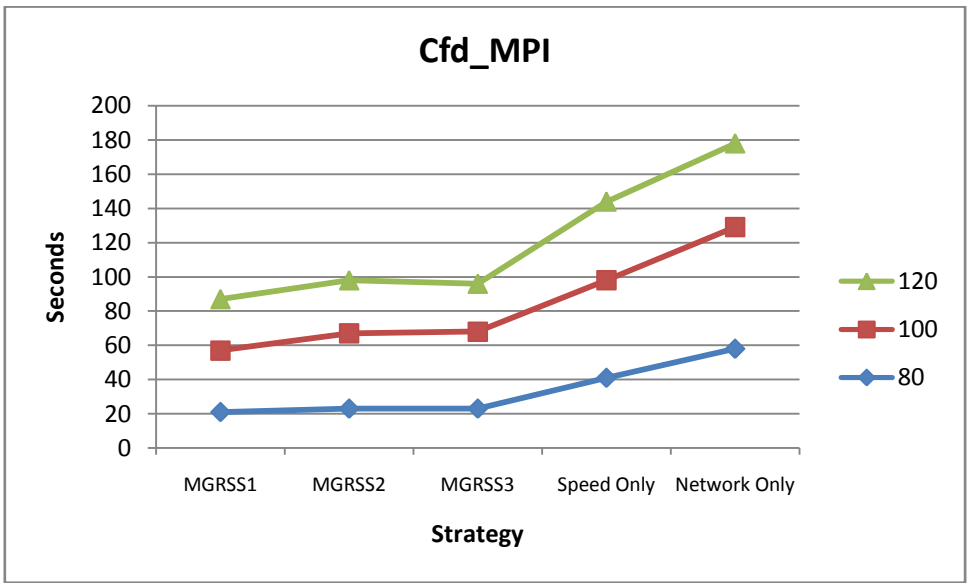


**Figure 5-6.** Result for Cfd _MPI in different strategies sequence

We execute the nine parallel programs which are describe in previous section. Similarly, we compared the MGRSS and other differences strategy again. The different characteristics of the program, so use the different strategies will produce different results between these situations. Overall, three levels of MGRSS strategies still better than other strategies. Experiment result show in Figure 5-7. The last experiment we randomly selected twenty different types and parameter of parallel programs to execute with the different strategies in twenty times, as shown in Figure 5-8. In some care, the speed only and the network only is better than the MGRSS, but MGRSS is better than other two strategies in the most of case.

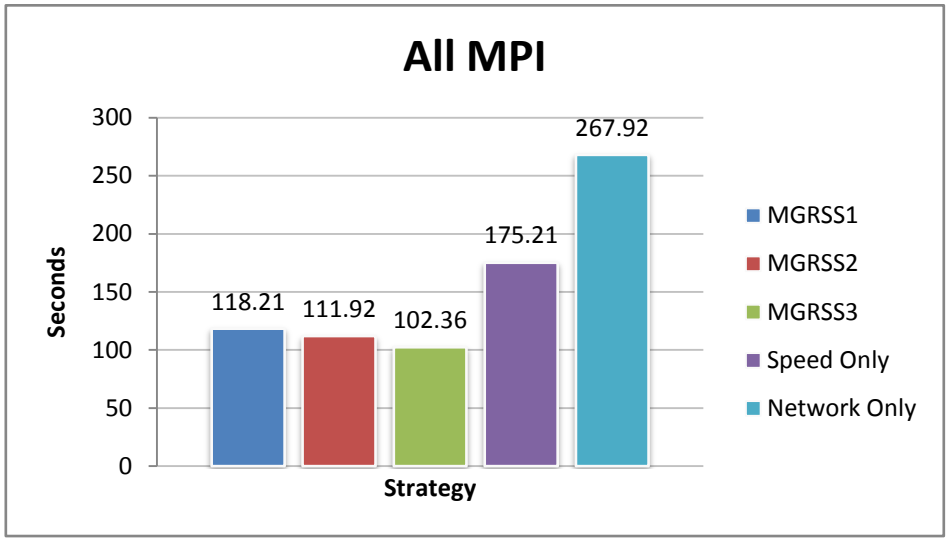


**Figure 5-7.** Result for all mpi programs

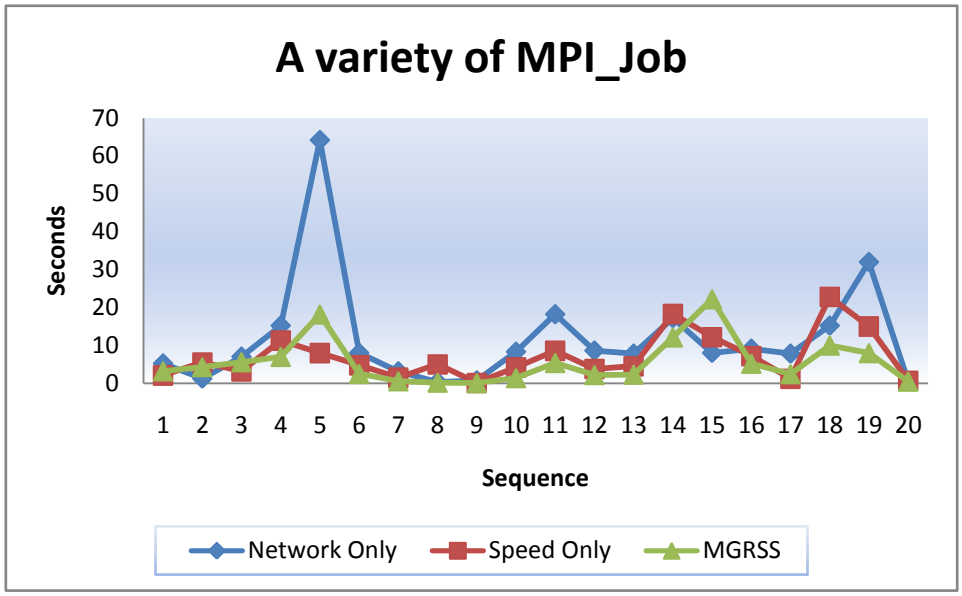


**Figure 5-8.** Result for variety of mpi programs

# Conclusion and Future Work

In this thesis, we construct a resource broker for multi-grid computational environment which integrates the four single grids into our environment, and provide the cross grid service. The resource broker enables users to submit job via web portal, and use the multi-grid resource selection strategy algorithm to select better resource avoid network congestion caused of decrease of performance.

The user execute the workflow achieve more high-performance computing by web portal without comprehend complicated instructions, and that can monitor status of multi-grid or grid itself. Our goal, more and more virtual organizations will join in the multi-grid environment, a huge computing resource also progressively grow up.

In the future, we focus on efficiently integrate the different grid middleware, such as Globus and GLite. Even though most of organization use the Globus as the grid middleware in Taiwan, but the GLite has still a lot of users in international. We also hope invite other school let their grid join our architecture, and allocate the account to use resource broker. Final, we will continue to enhance and improve the function of resource broker and multi-grid resource selection strategy.

# Bibliography


1. I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit," *International Journal of Supercomputer Applications*, 1997, vol. 11, pp. 115-128.

2. V. Laszewski, I. Foster, J. Gawor, and P. Lane, "A Java commodity grid kit," *Concurrency and Computation: Practice and Experience*, 2001, vol. 13, pp. 645-662.

3. K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, "Grid Information Services for Distributed Resource Sharing," Proceedings of *the Tenth IEEE International Symposium on High-Performance Distributed Computing*, 2001, pp.181-194.

4. I. Foster and N. Karonis, "A Grid-Enabled MPI: Message Passing in Heterogeneous Distributed Computing Systems," Proceedings of *1998 Supercomputing Conference*, 1998, pp. 46- 46.

5. J. Tang, and M. Zhang, "An Agent-based Peer-to-Peer Grid Computing Architecture," *Semantics, Knowledge and Grid, 2005, SKG '05, First International Conference on*, 2005, pp.57-57.

6. G. Aloisio and M. Cafaro, "Web-based access to the Grid using the Grid Resource Broker portal," *Concurrency Computation: Practice and Experience*, 2002, vol. 14, pp. 1145-1160.

7. K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Software Practice and Experience*, 2002, vol. 32, pp. 135-164.

8. L. Baduel and S. Matsuoka, "Peer-to-Peer Infrastructure for Autonomous Grid Monitoring," Proceedings of *Parallel and Distributed Processing Symposium*, 2007, vol. 35 , pp. 1-8..

9. D.H. Kim and K.W. Kang, "Design and Implementation of Integrated Information System for Monitoring Resources in Grid Computing," *Computer Supported Cooperative Work in Design*, 2006, pp.1-6.

10. F.D. Sacerdoti, M.J. Katz, M.L. Massie, and D. E. A. C. D. E. Culler, "Wide area cluster monitoring with Ganglia," *Cluster Computing, 2003.* Proceedings of *2003 IEEE International Conference on*, 2003, pp. 289-298.

11. W.C. Chung, R.S. Chang, "A new mechanism for resource monitoring in grid computing," *Future Generation Computer Systems*, 2009, vol. 25, pp. .

12. C.T. Yang, T.T. Chen and S.Y. Chen, "Implementation of Monitoring and Information Service Using Ganglia and NWS for Grid Resource Brokers," Proceedings of *2007 IEEE Asia-Pacific Services Computing Conference*, 2007, pp. 356-363.

13. C.T. Yang, C.L. Lai, P.C. Shih, and K.C. Li, "A Resource Broker for Computing Nodes Selection in Grid Environments," Proceedings of *Grid and Cooperative Computing - GCC 2004: 3rd International Conference*. 2004, vol.

3251, pp. 931-934.

14. C.T. Yang, P.C. Shih, and K.C. Li, “A high-performance computational resource broker for grid computing environments,” *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, 2005, vol.2, pp. 333-336.

15. C.T. Yang, K.C. Li, W.C. Chiang, and P.C. Shih, “Design and Implementation of TIGER Grid: an Integrated Metropolitan-Scale Grid Environment,” Proceedings of *the 6th IEEE International Conference on PDCAT’05*, 2005, pp. 518-520.

16. C.T. Yang, C.F. Lin, and S.Y. Chen, “A Workflow-based Computational Resource Broker with Information Monitoring in Grids,” Proceedings of *Fifth International Conference on Grid and Cooperative Computing (GCC'06)*, 2006, pp. 199-206.

17. C.T. Yang, S.Y. Chen, and T.T. Chen, “A Grid Resource Broker with Network Bandwidth-Aware Job Scheduling for Computational Grids,” Proceedings of *Grid and Pervasive Computing - Second International Conference.* 2007, vol. 4459, pp. 1-12.

18. C.T. Yang, P.C. Shih, and S.Y. Chen, “A Domain-Based Model for Efficient Measurement of Network Information on Grid Computing Environments,” *IEICE - Trans. Inf. Syst*. 2006, vol. E89-D, pp. 738-742.

19. W. Zhang, B. Fang, H. He, H. Zhang, M. Hu, "Multisite resource selection and scheduling algorithm on computational grid", Proceedings of *Parallel and Distributed Processing Symposium*, 2004, pp. 105-115.

20. F. Ian and K. Carl, “Globus: A Metacomputing Infrastructure Toolkit,” *International Journal of Supercomputer Applications*, 1997, vol. 11, pp. 115-128.

21. M.L. Massie, B.N. Chun, and D.E. Culler, “The Ganglia Distributed Monitoring System: Design, Implementation, and Experience,” *Parallel Computing*, 2004, vol. 30, pp. 817-840.

22. F.D. Sacerdoti, M.J. Katz, M.L. Massie, D.E. Culler, “Wide Area Cluster Monitoring with Ganglia,” Proceedings of *IEEE Cluster 2003 Conference*, 2003, pp. 289-298.

23. R. Wolski, N. Spring, and J. Hayes, “The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing,” *Future Generation Computing Systems*, 1999, vol. 15, pp. 757-768.

24. D.H. Kim, K.W. Kang, “Design and Implementation of Integrated Information System for Monitoring Resources in Grid Computing,” *Computer Supported Cooperative Work in Design*, 2006, pp. 1-6.

25. F. Ian and K. Carl, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1st edition, 1999.

26. Multi Grid, http://gamma2.hpc.csie.thu.edu.tw/

27. Tiger Grid, http://gamma2.hpc.csie.thu.edu.tw/ganglia/
28. Medical Grid, http://eta1.hpc.csie.thu.edu.tw/ganglia/
29. Bio Grid, http://140.128.98.25/ganglia/
30. Ganglia, http://ganglia.sourceforge.net/
31. Java CoG Kit, http://wiki.cogkit.org/
32. Globus, http://www.globus.org/
33. Condor, http://www.cs.wisc.edu/condor/
34. LEGION, http://www.cs.virginia.edu/~legion/
35. GridPP, http://www.gridpp.ac.uk/
36. EGEE, http://www.eu-egee.org/
37. P-Grid, http://www.p-grid.org/
38. DutchGrid, http://www.dutchgrid.nl/
39. ESnet, http://www.es.net/
40. GridBus, http://www.gridbus.org/
41. Cacti, http://www.cacti.net/
42. MPI, http://www-unix.mcs.anl.gov/mpi/.
43. MPICH, http://www-unix.mcs.anl.gov/mpi/.
44. MPICH-G2, http://www3.niu.edu/mpi/.
45. Network Weather Service, http://nws.cs.ucsb.edu/ewiki/.
46. Open Bioinformatics Grid, http://www.obigrid.org/.
47. The DataGrid Project, http://eu-datagrid.web.cern.ch/eu-datagrid/default.htm.
48. The Earth Simulator Center, http://www.es.jamstec.go.jp/index.en.html.
49. JRobin, http://oldwww.jrobin.org/
50. JFreeChart, http://www.jfree.org
51. HPL, http://www.netlib.org/benchmark/hpl/
52. CrossGrid, http://www.eu-crossgrid.org/