J Supercomput (2007) 40: 249–267
DOI 10.1007/s11227-006-0025-0

A resource broker with an efficient network information model on grid environments

Chao-Tung Yang · Po-Chi Shih · Cheng-Fang Lin · Sung-Yi Chen

Published online: 31 March 2007

Abstract This paper describes a resource broker whose main function is to match available resources to user needs. The resource broker provides a uniform interface for accessing available and appropriate resources via user credentials. We also focus on providing approximate measurement models for network-related information using NWS for future scheduling and benchmarking. We first propose a network measurement model for gathering network-related information (including bandwidth, latency, forecasting, error rates, etc.) without generating excessive system overhead. Second, we constructed a grid platform using Globus Toolkit that integrates the resources of five schools in Taichung integrated grid environment resources (TIGER). The resource broker runs on top of TIGER. Therefore, it provides security and current information about available resources and serves as a link to the diverse systems available in the Grid.

Keywords Resource broker · Grid computing · Globus toolkit · MDS · NWS · Efficient network model

C.-T. Yang (✉) · C.-F. Lin · S.-Y. Chen
High-Performance Computing Laboratory, Department of Computer Science and Information Engineering, Tunghai University, Taichung City 40704, Taiwan, R.O.C.
e-mail: ctyang@thu.edu.tw

C.-F. Lin
e-mail: superfun@sslab.cs.nthu.edu.tw

S.-Y. Chen
e-mail: g942805@thu.edu.tw

P.-C. Shih
Department of Computer Science, National Tsing Hua University, Hsinchu, 30013 Taiwan, R.O.C.
e-mail: shedoh@gmail.com

1 Introduction

Grids offer a way to solve Grand Challenge problems like protein folding, drug discovery, financial modeling, earthquake simulation, and climate/weather forecasting, among others. Grids enable organizations to make optimal use of information technology resources. And grids offer a means to act as a utility bureau in providing information technology to commercial clients who pay only for what they use, as with electricity or water [1, 3–12, 14–21].

Grid computing involves sharing, over an open-standards network, heterogeneous resources from various hardware and software platforms, computer architectures, and computer languages located in different places and belonging to different administrative domains. In short, it involves vitalizing computing resources. Functionally, one can classify grids as:

- computational and
- data.

Regardless of grid type, bandwidth management is a question of manipulating a number of variables to support the system and maximizing grid performance. As Grid Computing becomes a reality, there is a need to manage and monitor available resources worldwide, as well as a need to convey these resources to everyday users.

Most grids serving research and academic communities in North America and Europe utilize the Globus Toolkit® as their core middleware. The Globus Information Service, Monitor and Discover Service (MDS) provides good system-related information support on CPU speeds, CPU loading, memory utilization, etc., but no network-related information support. Therefore, we use the open-source program, Network Weather Service (NWS) [2], for network information.

NWS can measure point-to-point network bandwidth and latency that may be important for grid scheduling and load balancing. NWS detects all network states during time periods selected by the user. Because this kind of side-to-side measure results in $N(N-1)$ network measurement processes, the time complexity is O(N^2). Our network model focuses on solving the problem of reducing this time complexity without losing too much precision. There is another question. We want to know how NWS parameters (time period, frame size) influence our model and whether the NWS measurement value is inaccurate compared with real-world networks.

In this paper, we first describe a resource selection consideration and strategy which contains four phases and ten steps. And we implement a Grid resource broker based on these strategies. Second, we provide approximate measurement models for network-related information using NWS for future scheduling and benchmarking. Third, we provides a uniform interface for using our resource broker to accessing available and appropriate resources via user credentials. Fourth, we constructed a grid platform using Globus Toolkit that integrates the resources of five schools in Taichung integrated grid environment resources (TIGER). The resource broker runs on top of TIGER. Therefore, it provides security and current information about available resources and serves as a link to the diverse systems available in the Grid.

The remainder of this paper is organized as follows. Related studies are presented in Sect. 2 and the resource selection and strategy is introduced in Sect. 3. Our network information model is outlined in Sect. 4, and experimental results and a performance evaluation of our broker are presented in Sect. 5. Section 6 concludes this research paper.

2 Background review

2.1 Globus toolkit

The Globus Toolkit® is the open-source product of the Globus Project. It can implement large grid infrastructures and is freely available in [13]. The Globus Toolkit has emerged as the *de facto* standard for grid middleware with protocols to handle these four services:

- Resource management: Grid Resource Allocation & Management Protocol (GRAM)
- Information Services: Monitoring and Discovery Service (MDS)
- Security Services: Grid Security Infrastructure (GSI)
- Data Movement and Management: Global Access to Secondary Storage (GASS) and GridFTP

The Monitoring and Discovery Service (MDS) is the information service component of the Globus Toolkit. It provides information on available resources and computational states. The information may include machine properties, grid computers and networks, available processors, CPU loading, network interfaces, and file system information, bandwidth, storage devices, and memory. [4, 7, 8].

2.2 Network weather service

The Network Weather Service, though not targeted on Beowulf clusters, is a distributed system that periodically monitors and dynamically forecasts the performance various network and computational resources can deliver 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. NWS is a widely used measurement tool for grid environments. Studies on topics, such as load balancing, scheduling, brokering, etc. are available [4, 5, 10].

2.3 Java CoG Kit

The Java CoG Kit [9] provides access to Grid services through the Java 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. However, some of the components are provided as prototypes in JNI wrappers; we may work on a pure Java implementation of them, time permitting.

2.4 MPICH-G2

MPICH-G2 [2] is a grid-enabled implementation of the MPI v1.1 standard. That is, using services from 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 with 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.

3 Resource selection and strategy

Our resource broker is built on top of the Globus Toolkit. It makes use of Globus services, such as resource allocation, information, and GridFTP service. Our network monitor includes a measurement tool, cluster information provider, as well as NWS for forecasting network bandwidth (see Fig. 1).

Grid Resource Brokering involves four main phases: Resource Discovery, which generates lists of potential resources, Application Modeling, which enables users to characterize application behavior, Information Collection, which collects dynamical resource information, System Selection, which filters out resources that do not satisfy user requirements, then selects the best set of resources depending on system information, and Job Execution, which includes file transferring, pre-compilation, job execution, and result retrieval. These phases and the steps are shown in Fig. 2.

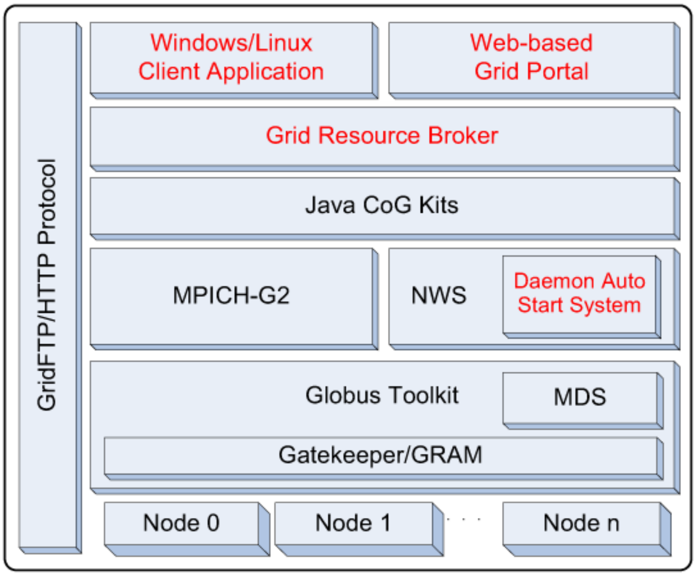


Fig. 1 Resource broker architecture

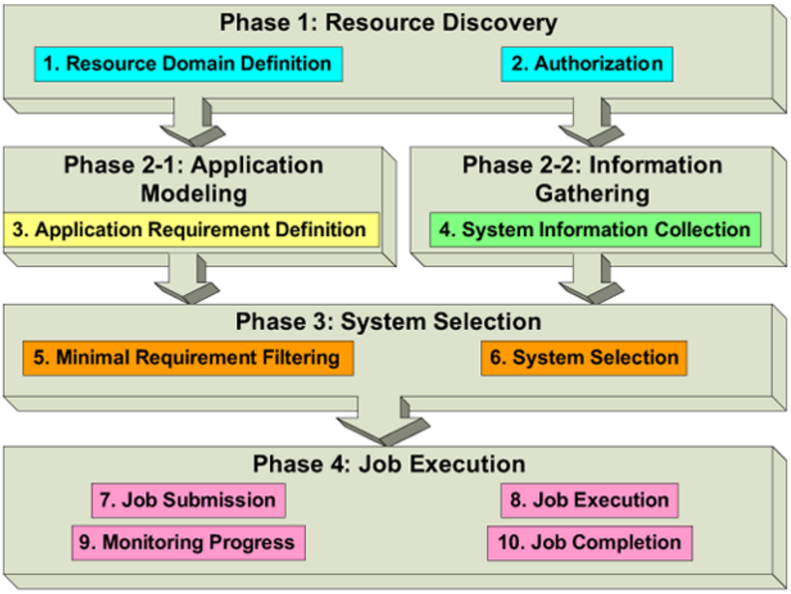

Phase 1: Resource Discovery
1. Resource Domain Definition
2. Authorization
Phase 2-1: Application Modeling
3. Application Requirement Definition
Phase 2-2: Information Gathering
4. System Information Collection
Phase 3: System Selection
5. Minimal Requirement Filtering
6. System Selection
Phase 4: Job Execution
7. Job Submission
8. Job Execution
9. Monitoring Progress
10. Job Completion

Fig. 2 Resource broker phases

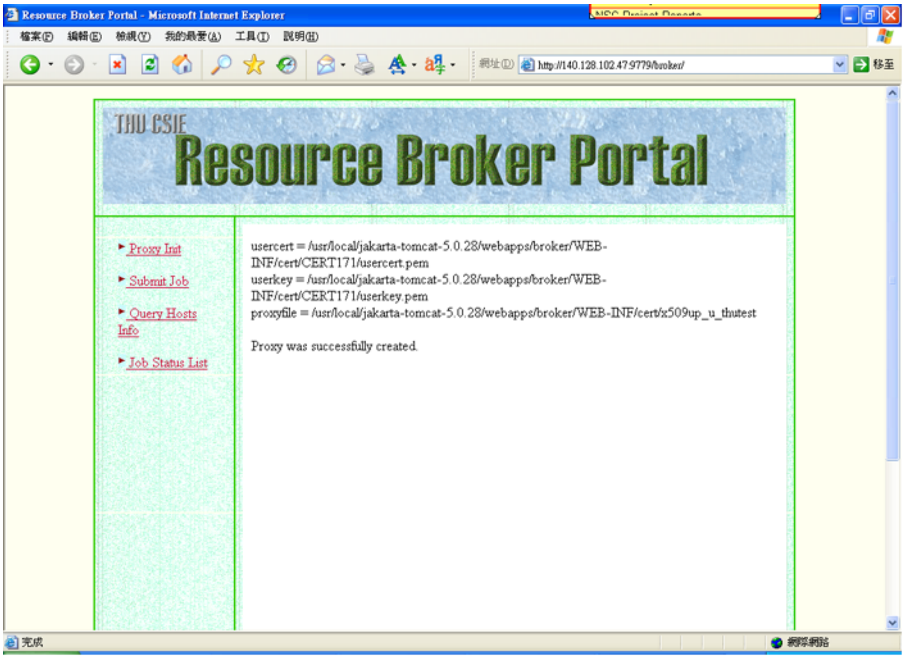
Phase 1 Resource Discovery The first stage of Resource Brokering determines which resources are available to various users. This is done in two steps: resource domain definition and authorization filtering.

Step 1: Resource Domain Definition The first step of resource discovery is choosing a set of resources jobs can submitted to. This is implemented in a host-list file showing all resources the user can access.

Step 2: Authorization Filtering The second step "authorizes" the user to go anywhere in the grid environment. The essence of a grid is that jobs can be submitted to anywhere from anywhere. So in this step the user gets a passport to access the resources defined in Step 1. We used GIS in Globus to implement this authorization, and the Java Cog Kit to develop a GUI, as shown in Fig. 3.

Phase 2-1 Application modeling This lets users define application characteristics and limitations. The model provides basic information that enables the broker scheduling algorithm to select the best resource distribution strategy.

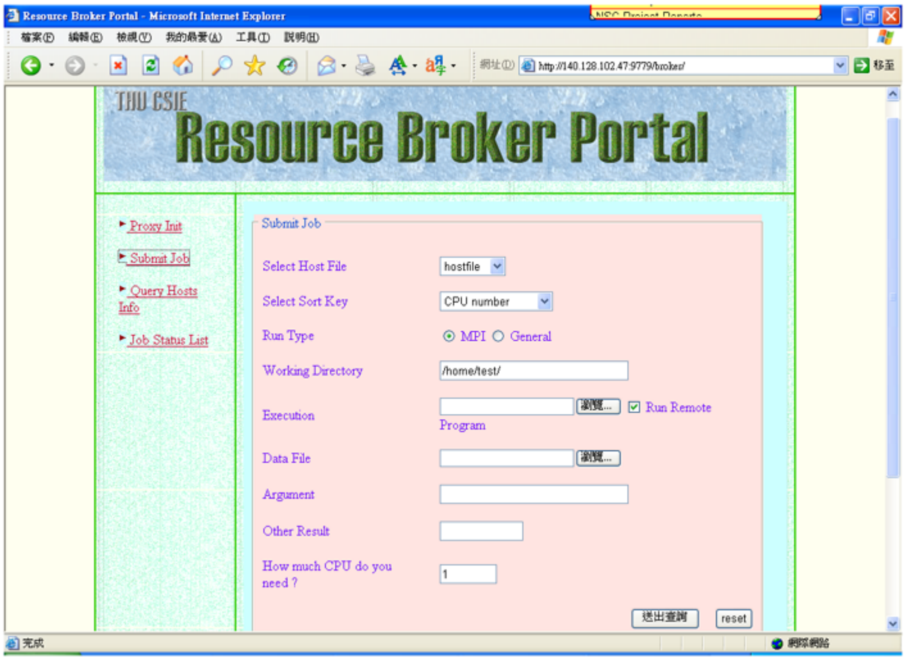
Step 3: Application Requirement Definition In order to select the proper resources to execute their programs, users must be able to specify some minimal set of job requirements and program types. Different jobs have different running requirements. The more details that are included, the better the matching effort will be. We created the GUI shown in Fig. 4 to allow manual selection of some minimal requirements. Also you can identify what types of environment(the item named "Select Sort Key") you wants (CPU intensive, network intensive).


Resource Broker Portal - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
http://140.128.102.47:9779/broker/
移至
THU CSIE
Resource Broker Portal
Proxy Init
Submit Job
Query Hosts Info
Job Status List
完成
網際網路

```
usercert = /usr/local/jakarta-tomcat-5.0.28/webapps/broker/WEB-
INF/cert/CERT171/usercert.pem
userkey = /usr/local/jakarta-tomcat-5.0.28/webapps/broker/WEB-
INF/cert/CERT171/userkey.pem
proxyfile = /usr/local/jakarta-tomcat-5.0.28/webapps/broker/WEB-INF/cert/x509up_u_thutest

Proxy was successfully created.
```

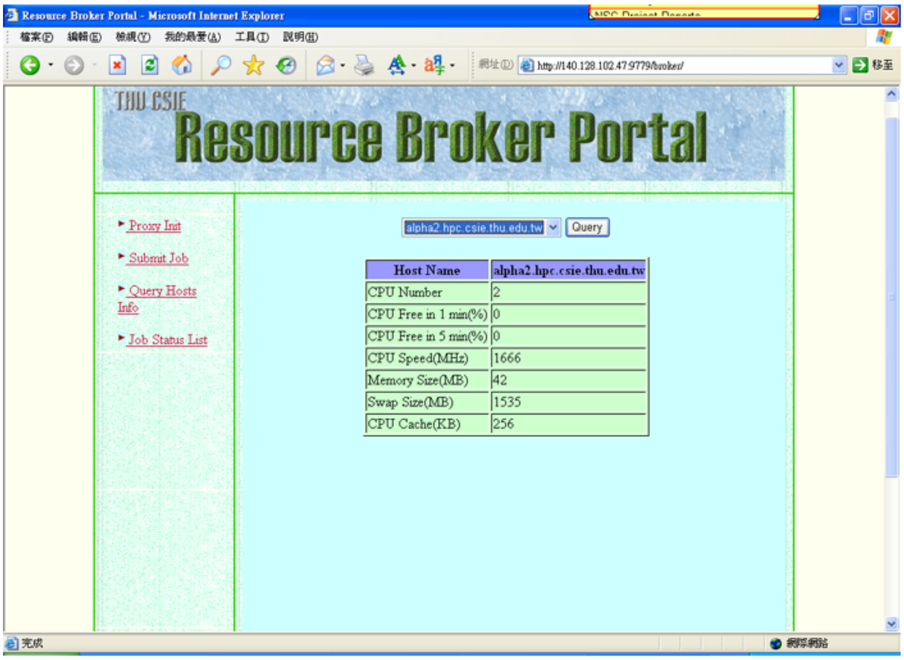
Fig. 3 Proxy initialization


Resource Broker Portal - Microsoft Internet Explorer
檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)
http://140.128.102.47:9779/broker/
移至
THU CSIE
Resource Broker Portal
Proxy Init
Submit Job
Query Hosts Info
Job Status List
瀏覽...
送出查詢
reset
完成
網際網路

Submit Job

Select Host File	hostfile
Select Sort Key	CPU number
Run Type	[● MPI] ○ General
Working Directory	/home/test/
Execution	[✓ Run Remote Program]
Data File	
Argument	
Other Result	
How much CPU do you need ?	1

Fig. 4 Application modeling


Resource Broker Portal - Microsoft Internet Explorer
http://140.128.102.47:9779/broker/
THU CSIE
Resource Broker Portal
Proxy Init
Submit Job
Query Hosts Info
Job Status List
alpha2.hpc.csie.thu.edu.tw
Query

Host Name	alpha2.hpc.csie.thu.edu.tw
CPU Number	2
CPU Free in 1 min(%)	0
CPU Free in 5 min(%)	0
CPU Speed(MHz)	1666
Memory Size(MB)	42
Swap Size(MB)	1535
CPU Cache(KB)	256

Fig. 5 Dynamic machine information

Phase 2-2 Information collection This phase is used to collect information on the machines used in the grid computing environment.

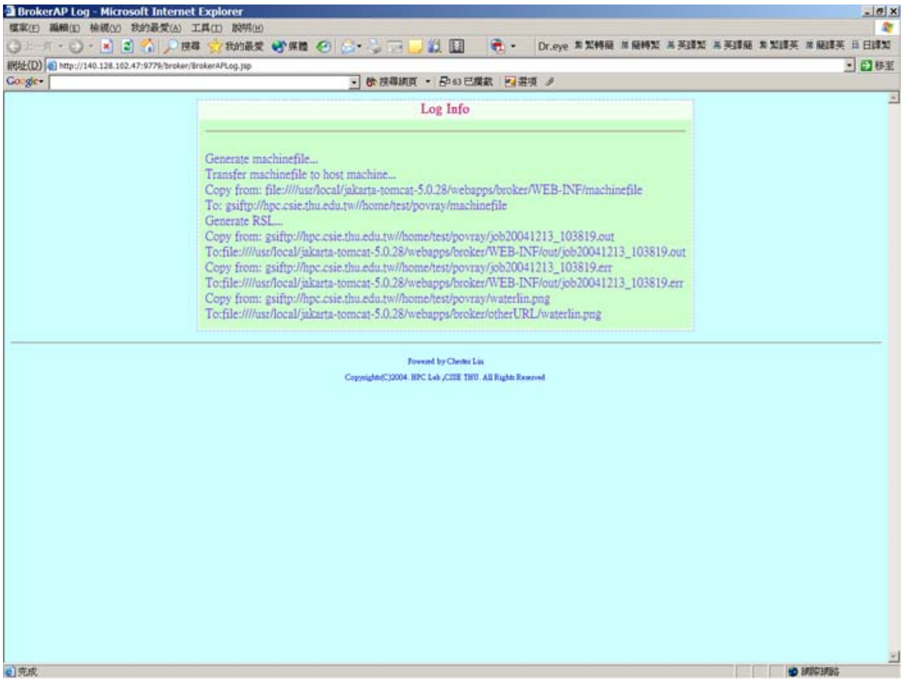
Step 4: System Information Collection Detailed dynamic information on resources is needed to make the best possible job/resource matches. The information helps the broker assess resource availability and status (speed, utilization, network Bandwidth, etc). Because it changes dynamically, this real-time information makes dynamic scheduling a reality. We implemented this step using Globus MDS and NWS. Detailed information is shown in Fig. 5.

Phase 3 System selection This phase selects the most appropriate resources for users.

Step 5: Minimal Requirement Filtering The fifth step filters out resources that do not satisfy the application requirements described in Step 3. The main function of this step is to reduce the size of the available and suitable resource set.

Step 6: System Selection Since dynamic information on resources is available, system selection is easy and scalable. When application requirements are defined and dynamic information about resources is available, users can make their own scheduling algorithms to handle various situations. At the end of this step, a set of appropriate resources has been generated, saved as a machine list, and is ready to run jobs.

Phase 4 Job execution The fourth phase of grid scheduling actually runs jobs on those resources.


BrokerAP Log - Microsoft Internet Explorer
http://140.128.102.47:9779/broker/BrokerAPLog.jsp
Log Info

```
Generate machinefile...
Transfer machinefile to host machine...
Copy from: file:////usr/local/jakarta-tomcat-5.0.28/webapps/broker/WEB-INF/machinefile
To: gsiftp://hpc.csie.thu.edu.tw//home/test/povray/machinefile
Generate RSL...
Copy from: gsiftp://hpc.csie.thu.edu.tw//home/test/povray/job20041213_103819.out
To:file:////usr/local/jakarta-tomcat-5.0.28/webapps/broker/WEB-INF/out/job20041213_103819.out
Copy from: gsiftp://hpc.csie.thu.edu.tw//home/test/povray/job20041213_103819.err
To:file:////usr/local/jakarta-tomcat-5.0.28/webapps/broker/WEB-INF/out/job20041213_103819.err
Copy from: gsiftp://hpc.csie.thu.edu.tw//home/test/povray/waterlin.png
To:file:////usr/local/jakarta-tomcat-5.0.28/webapps/broker/otherURL/waterlin.png
```

Powered by Chester Lin
Copyright(C)2004. HPC Lab ,CSIE THU. All Rights Reserved

Fig. 6 Job has been compiled and run

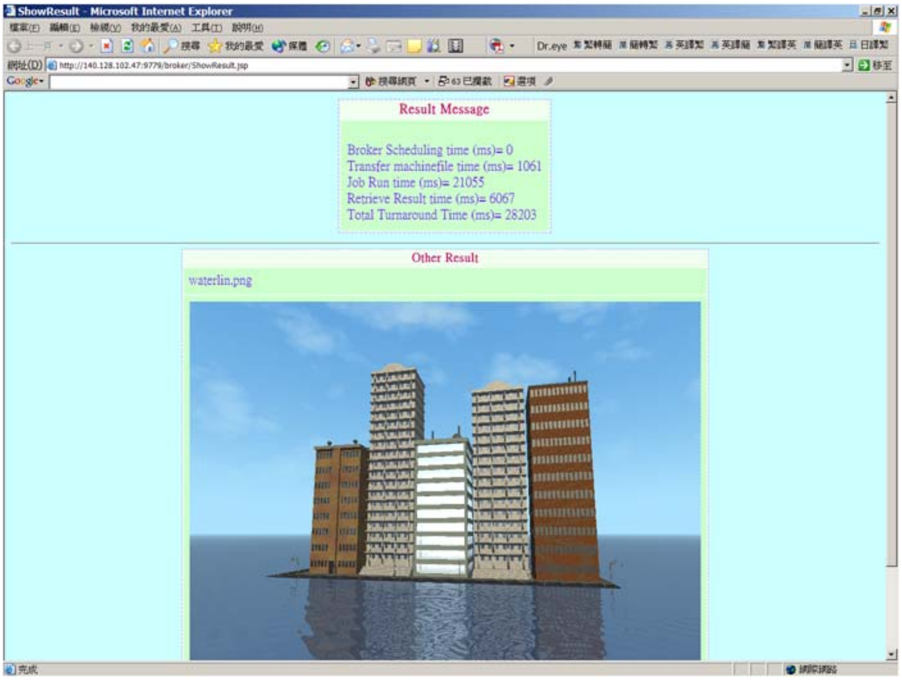
Step 7: Job Submission Before actually running a job, the application must be submitted to the resource set described in Step 6. This step is performed in two parts. The first action transfers the machine list, needed for MPI, to the first machine on our machine list via GridFTP because the first choice is the best based on application modeling. The second action transfers the application to all resources.

Step 8: Preparation Tasks Preparation may involve setup, compilation, and other actions needed to ready resources to run the application, and making sure program files, data files, argument files, and other set files are placed correctly. After the program is uploaded to target resources, the broker simultaneously sends the compiled operation to all machines and to the compilation source program. When compilation is finished, the broker begins parallel program execution.

Step 9: Progress Monitoring While the job runs, users can monitor the progress of their application and may elect to cancel or re-submit jobs. Historically, such monitoring has typically been done by repetitively querying resources for status information. However Globus is only able to interrupt users when jobs finish. GRAM provides basic status information such as running, finished, and failed. GRAM estimates how much time is needed to finish jobs, so only "running" is reported.

Step 10: Job Completion Users must be notified when jobs finish. The broker must able to interrupt users upon job completion. Sending e-mail or voice messages to user cell phones may implemented in the future.

While programs run, our broker shows a progress bar to indicate job status, and waits for completion. When jobs finish, the broker automatically retrieves results and


ShowResult - Microsoft Internet Explorer
http://140.128.102.47:9779/broker/ShowResult.jsp
Google
Dr.eye

Result Message

Broker Scheduling time (ms)= 0
Transfer machinefile time (ms)= 1061
Job Run time (ms)= 21055
Retrieve Result time (ms)= 6067
Total Turnaround Time (ms)= 28203

Other Result

waterlin.png

Fig. 7 Job is finished and result is shown

displays them to users. The four steps in this phase are continuous actions implemented in our resource broker. Snapshots are shown in Figs. 6 and 7.

4 Network information model and analyses

We constructed a network measurement model to solve a complete point-to-point network measurement problem [20]. Consider the grid environment with twelve nodes shown in Fig. 8. The lines linking the nodes represent site-to-site network measurement. This model is often used for local grids or cluster environments when the scale is not too large. In large-scale grid environments this kind of architecture results in excessive bandwidth overhead. In order to reduce the total number of times that NWS measures, we proposed the "domain" concept shown in Fig. 9 to cut the network measurement environment.

In our domain-based model, we treated several hosts as domains. Figure 9, shows three domains, 5 hosts at left, 3 hosts on top, and 4 hosts at right-bottom. We linked hosts in each domain, called Borders, with one another forming a central-domain. We thus only needed pair measurements within domains. Domain-to-domain data was measured by the central-domain, considerably reducing the number of measurements required. We extend existing clique models in NWS and consider how to construct domain-based models from geographical and real bandwidth perspectives.

We may separate schools or organizations into domains on the basis of geography. Hosts in each domain are thought as tightly coupled, and it's convenient for each

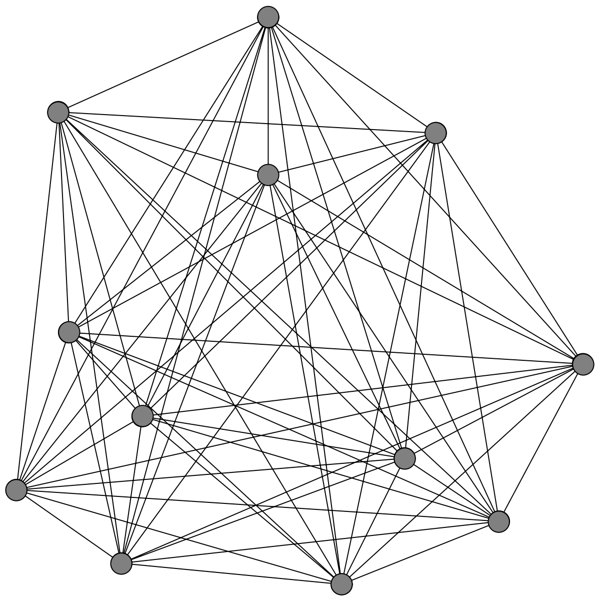


Fig. 8 Network measurement model

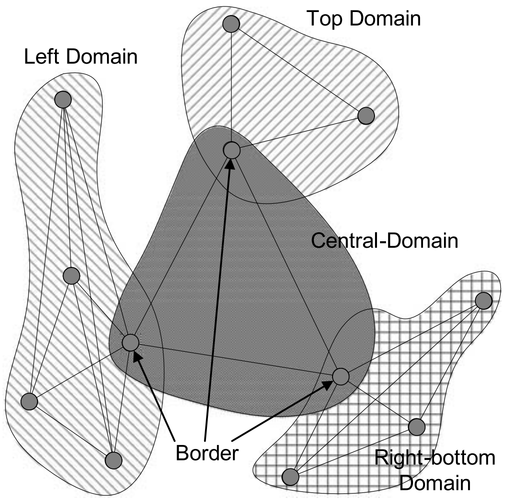


Fig. 9 Domain-based network measurement model

domain to control and maintain its hosts. The domains may each use a different network infrastructure: Fast Ethernet, Gigabit, or InfiniBand. This design ensures that local fluctuations won't affect the entire grid system.

Some questions about the model remain:

- how to select a representative host in each domain to form a central domain without loss of generality?
- how to accurately evaluate host-to-host network information?

These questions are explored in detail below. Constructing a domain-based grid is the key issue. Domains must first be constructed by the schools or organizations. One

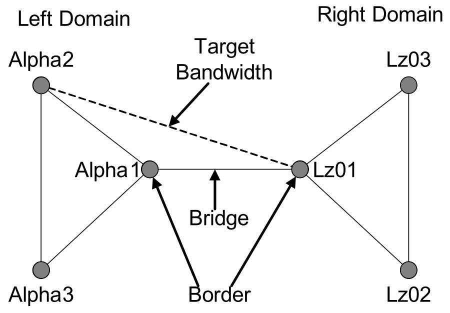


Fig. 10 Network estimate model

way to select the best Border in each domain is to conduct an all-pair network test. But this may not work in a real grid environment because each organization controls its hosts according to a different policy. Another way is to let domain administrators select Borders according to network topology or architecture. Both methods are not smart and are hard to scale, so we propose an alternative way to select Borders.

First, when the grid is just being built up, pick partial domains (maybe 2–4 domains) to start with. Second, let administrators select Borders or perform an all-pair test to select the best starting Borders. Then save these hosts in a Border list. Third, let each group administrator select a Border according to network topology or architecture, or test all hosts in each domain against each Border on the list to find the best one and add it to the list. Fourth, repeat the third step until every domain finds a best Border. This method simplifies the construction complexity and makes the grid environment scalable. When a new member joins this grid, only steps 3 and 4 need be performed to select its Border. There will be no need to change the original settings of the other domains.

In order to determine the all-pair network value, we use a few measured values from NWS to estimate other point-to-point values. Figure 10 shows an example of a network estimate model. The line connecting Alpha1 and Lz01 is part of the central-domain, which we called the Bridge in our experimental environment. The solid lines mean that our domain model has gotten the network information; the dotted lines are examples of many lines our model has not measured, so we use an evaluation model to calculate them. We use this notation throughout the paper:

- B_in_{avg}: Average inner-domain bandwidth.
- B_out_{avg}: Average outer-domain bandwidth.
- P_{flu}: Bandwidth fluctuation rate.
- N_{flu}: Number of times fluctuation occurs.
- P_{vaflu}: Valid fluctuation rate.
- $Lij(N_{\text{flu}})$: Latest N_{flu} measured from host i to host j.

B_in_{avg} and B_out_{avg} are obtained by averaging the bandwidth history or from the beginning value assigned by the administrator. P_{flu} detects bandwidth use. N_{flu} traces network fluctuations in a given time period ignoring pulse or bandwidth noise. P_{vaflu} shows how much fluctuation during N_{flu} time is treated as actual bandwidth use. We employ our algorithm in three separate cases in order to consider possible bandwidth usage patterns.

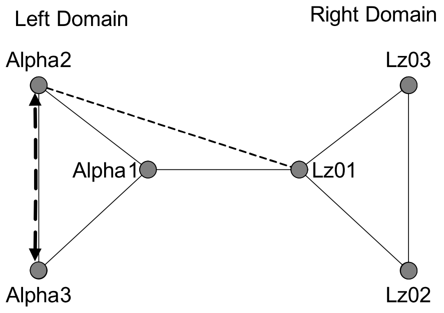


Fig. 11 Bandwidth usage in case 1

Case 1:

Assume the inner domain bandwidth use shown in Fig. 11. This is complex because the usage between Alpha2 and Alpha3 may not affect the Bridge bandwidth much. We use an algorithm to calculate target bandwidth.

First, left domain bandwidth fluctuation is examined, ignoring pulse or bandwidth noise fluctuation:

$$Use = \text{CountIf}\left(\frac{|L_{ij}[k] - B\_in_{\text{avg}}|}{B\_in_{\text{avg}}} > P_{\text{flu}}\right) > (N_{\text{flu}} * P_{\text{vaflu}}),\ k = 1, \ldots, N_{\text{flu}} \forall ij \quad \text{in left domain} \tag{1}$$

We then calculate the remaining bandwidth use, ignoring the maximal and minimal bandwidth values by first Sort (L_{ij}), then compute:

$$B_{\text{rem}} = \frac{\sum_{k=2}^{N_{\text{flu}}-1} L_{ij}[k]}{N_{\text{flu}} - 2} \tag{2}$$

Finally, the target bandwidth is calculated as follows:

$$B_{\text{tar}} = \frac{B_{\text{rem}}}{B\_in_{\text{avg}}} \times B\_out_{\text{avg}} \times \alpha \tag{3}$$

The symbol α here indicates a value converted from internet bandwidth to LAN and is used throughout.

Case 2:

We assumed that bandwidth use occurs in the same organization but not with other members of the domain, as shown in Fig. 12. Figure 13 shows the general topology of this network architecture. Using a simple test, we discovered that target bandwidth almost always follows bridge bandwidth. So we summarize briefly that target bandwidth is almost always equal to bridge bandwidth.

Case 3:

We assumed that bandwidth use occurs between two domains, as shown in Fig. 14. Bandwidth use between Alpha2 and Lz02 will affect the available bridge bandwidth, so we summarize briefly that target bandwidth is almost always equal to bridge bandwidth.

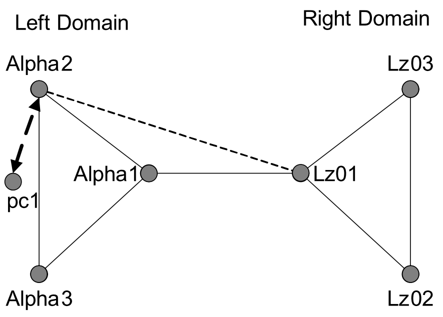


Fig. 12 Bandwidth usage in case 2

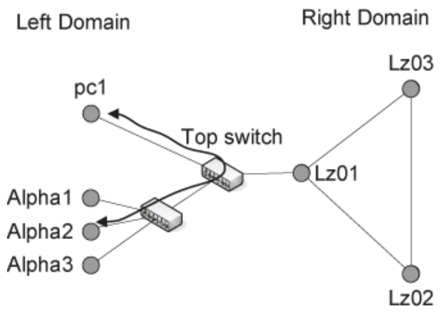


Fig. 13 Topology for case 2

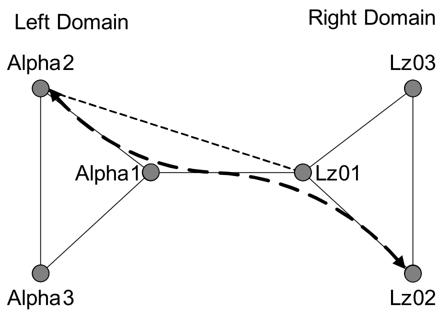


Fig. 14 Bandwidth usage in case 3

5 Experimental results of resource broker

We used the grid test setup shown in Fig. 15 on experiments with Taichung integrated grid environment resources (TIGER). TIGER contained four domains, with four hosts in THU, 8 hosts in PU, 4 hosts in HIT, and 4 hosts in LZ. The resource specifications are shown in Table 1. We ran three well-known parallel applications: Matrix Multiplication, the Prime problem, and CFD.

In our first experiment, we ran all four jobs with 1 to 8 CPUs using our resource broker and on other random hosts. The results are shown in Figs. 16 to 20.

The list below shows the experimental variables for our broker scheduling step.

- IST—Information Search Time.
- BST—Broker Scheduling Time.
- TMT—Machine File Transfer Time.

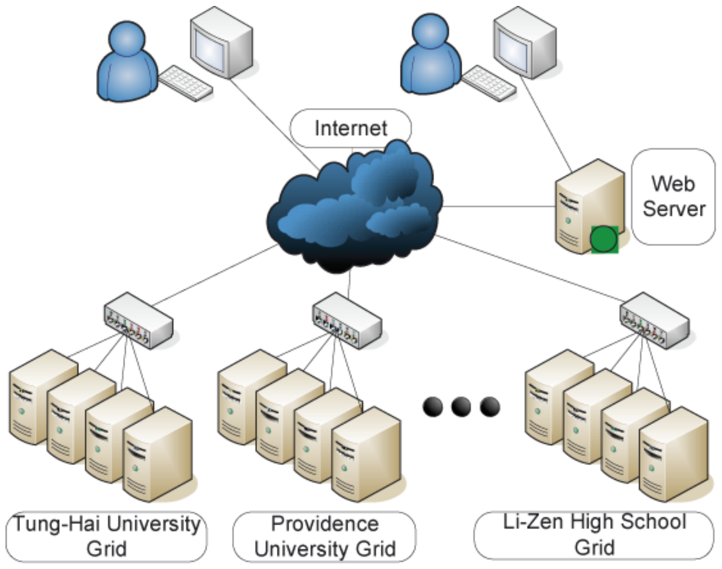


Fig. 15 The TIGER experimental environment

Table 1 Resource specification

	Resource	CPU Type	speed	Mem	Network	OS	kernel
THU	alpha1	AMD Athlon(tm) MP	2400+ x 2	1G	10/100	fedora core 1	2.4.22
	alpha2	AMD Athlon(tm) MP	2000+ x 2	768 MB	10/100	fedora core 1	2.4.22
	alpha3	AMD Athlon(tm) MP	1800+ x 2	512 MB	10/100	fedora core 1	2.4.22
	alpha4	AMD Athlon(tm) MP	1800+ x 2	512 MB	10/100	fedora core 1	2.4.22
LZ	lz01	Celeron	900	256 MB	10/100	fedora core 1	2.4.20-31.9
	lz02	Celeron	900	256 MB	10/100	fedora core 1	2.4.20-31.9
	lz03	Celeron	900	384 MB	10/100	fedora core 1	2.4.20-31.9
	lz04	Celeron	900	256 MB	10/100	fedora core 1	2.4.20-31.9
HIT	gridhit0	Pentium 4	2.8G	512 MB	10/100	fedora core 1	2.4.20-8
	gridhit1	Pentium 4	2.8G	512 MB	10/100	fedora core 1	2.4.20-8
	gridhit2	Pentium 4	2.8G	512 MB	10/100	fedora core 1	2.4.20-8
	gridhit3	Pentium 4	2.8G	512 MB	10/100	fedora core 1	2.4.20-8
PU	hpc09	AMD Athlon(tm) XP	2400+	1G	10/100	fedora core 1	2.4.22
	hpc10	AMD Athlon(tm) XP	2400+	1G	10/100	fedora core 1	2.4.22
	hpc11	AMD Athlon(tm) XP	2400+	1G	10/100	fedora core 1	2.4.22
	hpc12	AMD Athlon(tm) XP	2400+	1G	10/100	fedora core 1	2.4.22
	hpc13	AMD Athlon(tm) XP	2400+	1G	10/100	fedora core 1	2.4.22
	hpc14	AMD Athlon(tm) XP	2400+	1G	10/100	fedora core 1	2.4.22
	hpc15	AMD Athlon(tm) XP	2400+	1G	10/100	fedora core 1	2.4.22
	hpc16	AMD Athlon(tm) XP	2400+	1G	10/100	fedora core 1	2.4.22

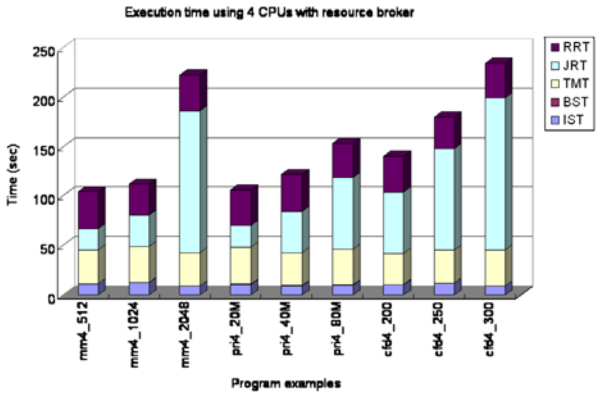


Fig. 16 Execution times for 4 CPUs using resource broker

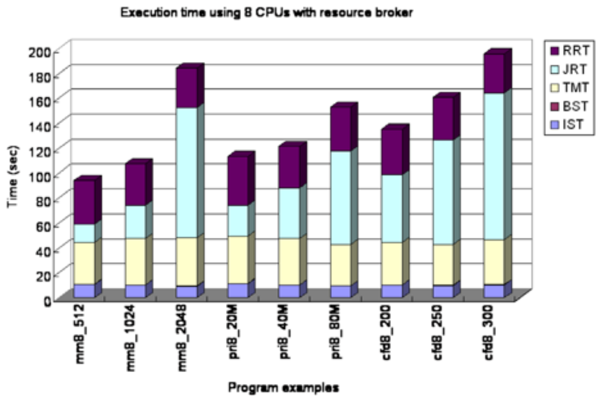


Fig. 17 Execution times for 8 CPUs using resource broker

- JRT—Job Run Time.
- RRT—Result Retrieval Time.
- TTT—Total Turnaround Time.

Figures 16 and 17 show that when the broker schedules with a small matrix, the overhead seems large relative to JRT, but when the matrix size is increased, the overhead decreases compared with JRT. Grid computing is suitable for large-scale problems, thus our broker overhead was constant and did not increase linearly with problem size. This means overhead can be ignored for large problems.

Figures 18 and 19 show JRT times compared to those for Linux console execution without our broker interface. We can see that constant overhead existed for Globus and Java Cog, but was acceptable compared with job size.

Figure 20 shows comparisons of execution times for our resource broker (left) and random host selections (right) with various numbers of CPUs ("m1_512" means matrix multiplication of a 512 ∗ 512 matrix using 1 CPU). It shows using our resource broker and achieves better performance compare with random select resources. But the key point is user need to know detail behavior of their jobs to select "right" types of job in order to get best performance.

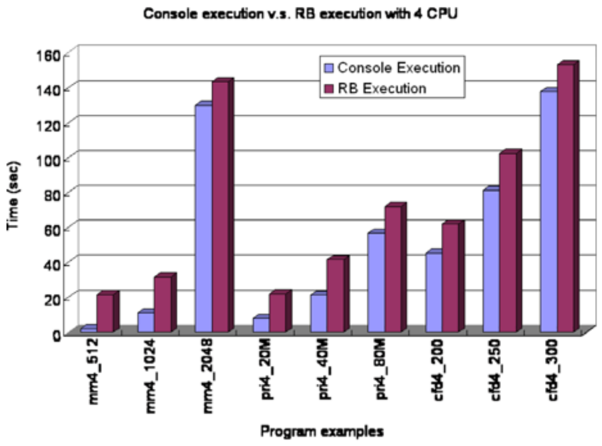


Fig. 18 Console execution vs. RB execution with 1 CPU

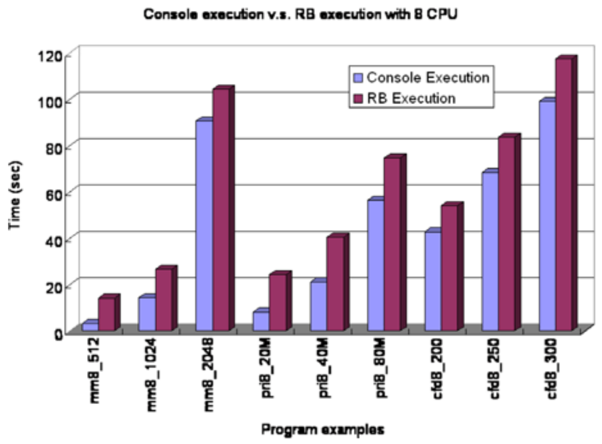


Fig. 19 Console execution vs. RB execution with 2 CPU

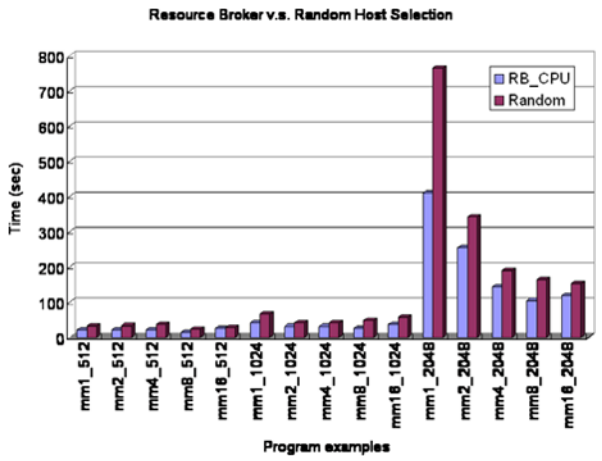


Fig. 20 Resource Broker vs. random host selection

6 Conclusions

As Grid Computing becomes a reality, there is a need to manage and monitor available resources worldwide, as well as a need to convey these resources to everyday users. This paper describes a resource broker whose main objective is to match available resources to user needs. The resource broker provides a uniform interface for accessing available and appropriate computing resources. Our experiments showed that our resource broker is able to find suitable resources that reduce total execution times. We proposed a network measurement model for gathering network-related information (including bandwidth, latency, forecasting, error rates, etc.) without generating excessive system overhead. We constructed a grid platform using Globus Toolkit that integrates the resources of five schools in Taichung integrated grid environment resources (TIGER). The resource broker runs on top of TIGER. Therefore, it provides security and current information about available resources and serves as a link to the diverse systems available in the Grid. In the future, we will plan to construct an efficient network measurement model to enable our broker to handle large communication jobs and predict execution times.

Acknowledgements This paper is supported in part by National Science Council Taiwan R.O.C., under grants no. NSC93-2213-E-029-026, NSC94-2213-E-029-002, and NSC95-2221-E-029-004. The authors would like to acknowledge the National Center for High-Performance Computing in providing resources under the National project "Taiwan Knowledge Innovation National Grid."

References

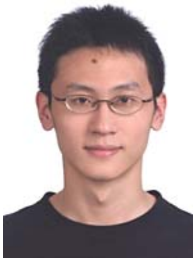
1. Allcock B, Bester J, Bresnahan J, Chervenak AL, Foster I, Kesselman C, Meder S, Nefedova V, Quesnal D, Tuecke S (2002) Data management and transfer in high performance computational grid environments. Parallel Comput 28(5):749–771
2. Allcock B, Tuecke S, Foster I, Chervenak A, Kesselman C (2000) Protocols and services for distributed data-intensive science. In: ACAT2000 proceedings, 2000, pp 161–163
3. Allcock W, Bester J, Bresnahan J, Chervenak A, Liming L, Meder S, Tuecke S (2002) GridFTP protocol specification. GGF GridFTP working group document, September 2002
4. Czajkowski K, Foster I, Kesselman C (1999) Resource co-allocation in computational grids. In: Proceedings of the 8th IEEE international symposium on high performance distributed computing (HPDC-8), 1999, pp 219–228
5. Czajkowski K, Fitzgerald S, Foster I, Kesselman C (2001) Grid information services for distributed resource sharing. In: Proceedings of the 10th IEEE international symposium on high-performance distributed computing (HPDC-10), August 2001
6. Foster I, Kesselman C, Tuecke S (2001) The anatomy of the grid: enabling scalable virtual organizations. Int J Supercomput Appl High Perform Comput, 15(3):200–222
7. Global grid forum. http://www.ggf.org/
8. IBM Redbooks (2003) Introduction to grid computing with globus. http://www.redbooks.ibm.com/redbooks/pdfs/sg246895.pdf, IBM Press, September 2003
9. Laszewski V, Foster I, Gawor J, Lane P (2001) A Java commodity grid kit. Concurr Comput Pract Exp 13:645–662
10. MPICH-G2. http://www.hpclab.niu.edu/mpi/
11. Network weather service. http://nws.cs.ucsb.edu/
12. Park SM, Kim JH (2003) Chameleon: a resource scheduler in a data grid environment. In: Proceedings of the 3rd IEEE/ACM international symposium on cluster computing and the grid, May 2003, pp 258–265
13. The globus alliance. http://www.globus.org/

14. Yang C-T, Chu WC (2004) Grid computing in Taiwan. In: Proceedings of 10th international workshop on future trends of distributed computing systems (FTDCS 2004), Suzhou, China, May 26–28, 2004, pp 201–204
15. Yang C-T, Ho H-C (2005) An e-learning platform based on grid architecture. J Inf Sci Eng 21(5):115
16. Yang C-T, Kuo Y-L, Lai C-L (2005) Designing computing platform for BioGrid. Int J Comput Appl Technol (IJCAT), special issue applications for high performance systems 22(1):3–13, Inderscience Publishers, ISSN (Paper): 0952-8091, UK
17. Yang C-T, Lai C-L (2004) Apply cluster and grid computing on parallel 3D rendering. In: Proceedings of the 2004 IEEE international conference on multimedia and expo (ICME 2004), Grand Hotel, Taipei, Taiwan, June 27–30, vol 2, 2004, pp 859–862
18. Yang C-T, Lai C-L, Shih P-C, Li K-C (2004) A resource broker for computing nodes selection in grid environments. In: Jin H, Pan Y, Xiao N (eds), Grid and cooperative computing—GCC 2004: third international conference, lecture notes in computer science, vol 3251. Springer, Oct 2004, pp 931–934
19. Yang C-T, Shih P-C, Li K-C (2005) A high-performance computational resource broker for grid computing environments. In: Proceedings of the international conference on advanced information networking and applications (AINA 2005), 2005, Tamkang University, Taipei, Taiwan, March 28–30, vol 2, pp 333–336
20. Yang C-T, Shih P-C, Chen S-Y (2006) A domain-based model for efficient network information on grid computing environments. IEICE Trans Inf Syst, special issue on parallel/distributed computing and networking E89-D(2):738–742
21. Yang C-T, Yang I-H, Li K-C, Wang S-Y (2006) Improvements on dynamic adjustment mechanism in co-allocation data grid environments. J Supercomput (accepted)
22. Zhang X, Freschl JL, Schopf JM (2003) A performance study of monitoring and information services for distributed systems. In: Proceedings of HPDC, IEEE CS Press, August 2003, pp 270–282

Chao-Tung Yang received a B.S. degree in computer science and information engineering from Tunghai University, Taichung, Taiwan in 1990, and the M.S. degree in computer and information science from National Chiao Tung University, Hsinchu, Taiwan in 1992. He received the Ph.D. degree in computer and information science from National Chiao Tung University in July 1996. He won the 1996 Acer Dragon Award for outstanding Ph.D. Dissertation. He has worked as an associate researcher for ground operations in the ROCSAT Ground System Section (RGS) of the National Space Program Office (NSPO) in Hsinchu Science-based Industrial Park since 1996. In August 2001, he joined the faculty of the Department of Computer Science and Information Engineering at Tunghai University, where he is currently an associate professor. His researches have been sponsored by Taiwan agencies National Science Council (NSC), National Center for High Performance Computing (NCHC), and Ministry of Education. His present research interests are in grid and cluster computing, parallel and high-performance computing, and internet-based applications. He is both member of the IEEE Computer Society and ACM.

Po-Chi Shih received the B.S. and M.S. degrees in Computer Science and Information Engineering from Tunghai University in 2003 and 2005, respectively. He now is studying Ph.D. degree at Computer Science in National Tsing Hua University, Hsinchu, Taiwan from September 2005. His present research interests are grid computing and internet-based applications.

Cheng-Fang Lin received a B.S. and M.S. degree in Department of Computer Science and Information Engineering from Tunghai University in 2004 and 2006, respectively. He is studying Ph.D. degree in Department of Computer Science at National Tsing Hua University, Hsinchu, Taiwan from September 2006. His research interests include parallel and distributed processing, high-performance computing, grid and pervasive computing.



Sung-Yi Chen received a B.S. degree in Department of Computer Science and Information Engineering from Tunghai University in 2005. He is studying M.S. degree in Department of Computer Science and Information Engineering from Tunghai University from September 2005. His research interests include grid and cluster computing, parallel and high-performance computing, grid and pervasive computing.