DIRSS-G: an Intelligent Resource Scheduling System for Grid Environment Based on Dynamic Pricing*

Li Lu [1] and Shoubao Yang [2]

[1, 2]Department of Computer Science and Technology
University of Science and Technology of China
Jinzhai Road, Hefei 230026, P.R.China

[1]lulixiao@mail.ustc.edu.cn
[2]syang@ustc.edu.cn

Abstract

Existing resource management and scheduling systems for Grid have defects in scalability and load balance. Based on computational economy framework, this paper presents a decentralized intelligent resource scheduling system (DIRSS-G) for Grid environment using the supply and demand theory. The system provides a bi-directional choosing mechanism and a QoS guaranteeing mechanism for users and resources to supervise them heuristically. The simulation result of the system shows that the system is scalable, flexible, and capable of handling load balance well. Meanwhile it guarantees QoS of tasks. Task Accomplishment ratio in DIRSS-G is greater than that in Nimrod/G by 22.5%.
Keywords: computational economy-based model, bi-directional choosing, load balance, QoS.

1. Introduction

Grid technology is deemed as a critical element of current high performance computing environments. The real and specific problems that underlie the Grid concept are coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations [1]. At present, Grid technology is applied to many fields, such as scientific computing, data management, system integration, etc. One of the most important problems in Grid technology is how to assign grid resources and schedule grid tasks reasonably. And resource scheduling in Grid environment is distinct from conventional distributed computing by autonomic heterogeneous resources and large-scale resource sharing. Existing centralized resource-scheduling systems for Grid expose more and more maladjustments. For these reasons, the computational economy-based model [2] is introduced, harnessing economic principles to solve resource sharing in Grid environment. Now some resource scheduling systems based on the model are proposed, yet they are in the primary stage, having many deficiencies. In this paper, we design and simulate a decentralized intelligent resource scheduling system (DIRSS-G) based on the model, which is an improvement on other systems of its kind. DIRSS-G is characteristic of load balance and task QoS. Meanwhile it has good scalability and flexibility.

The remainder of this article is organized as follows. In section 2, we review current resource scheduling systems for Grid environment based on the model. In section 3, we describe abstract entities in DIRSS-G. And then in section 4, we propose the design of DIRSS-G. In section 5, some

* This paper is supported by the National Natural Science Foundation of China under Grant No.60273041 and the National '863' High-Tech Program of China under Grant No. 2002AA104560.

experiment results are shown and explained in detail. Finally, we draw a conclusion with regarding future work in section 6.

2. Related Works

Representatives of current researches based on the model are POPCORN project [3] in Israel and Nimrod/G [4] project in Australia. We will analyze them in detail.

In POPCORN, the scheduling system utilizes an entity called Market to distribute Grid tasks under specific auction rules centrally. Obviously this mechanism has defects in scalability, forming a bottleneck in Market entity. Moreover all buyers and sellers must comply with the same auction rule. In Nimrod/G, although the scheduling system is decentralized, it distributes Grid tasks based on the forecast that may be not accurate. And strategies in Nimrod/G are based on fixed prices. In this condition, load is unbalance undoubtedly. In addition, choices between users and buyers are unidirectional, i.e. users can choose resources freely, but resources only accept tasks passively. Resources will accept more and more tasks with time go on, but couldn't reject them. The running time of a task will be prolonged. Task Accomplishment ratio and task QoS go down. In conclusion, they couldn't deal with resource scheduling well in Grid environment.

3. Abstract Entities in DIRSS-G

DIRSS-G possesses essential abstract entities in the model too. Behaviors of users and buyers abide by economic principles. But these entities must be adapted for DIRSS-G. We will describe these entities in detail.

One essential theory in economics is the supply and demand theory. We apply the theory to DIRSS-G to balance load. A resource can adjust its price according to its load, to affect users' choices.

myGridResource:
Resource ID: RID
Gdollor: ResMoney
Resource description file: ResourceCharacteristics
Pricing function: updatePrice(Current Load). An example is the formula 1.
Task scheduling: AllocPolicy: Round Robin Scheduling, or Time Slices Scheduling, etc.

$$P = \begin{cases} \min \Pr ice \times l + \min \Pr ice & (0 \le l \le b) \\ (1+b) \times \min \Pr ice + ((l-b) \times 10)^2 / c & (l > b) \end{cases} \qquad (1)$$
$$\min \Pr ice \le P \le \max price$$

In formula 1, P is the price of a Grid resource with certain load; variable l is a load factor representing current load. Constant b represents the valve value of resource load for price increasing or decreasing; c controls the intensity degree of price changing. Resource administrator should be able to define minPrice and minPrice as the lowest and highest price threshold respectively.

When $l > b$, the resource is over loads, we use quadratic function to raise its price rapidly until reaching its maxPrice, to hamper users to choose the resource; when $0 \le l \le b$, we use linear function to reduce price slowly. So the resource will not quickly fall into overload again. In different conditions, we can choose distinct pricing functions to adjust price in time.

myUser:
UserID: UID

Gdollor: UserMoney
Resourcebroker: myBroker
Task queue: myTaskList queue of tasks needed completing

myBroker:
A myBroker acts as an agent of a special user and can be configured for special requirements, such as cost optimization algorithm, time optimization algorithm, etc. It does some works for the user, including choosing resources, submitting tasks, and receiving results.

myTask:
TaskID: GridletID
Task description: GridletScript, describing properties of a task, such as task Length, task deadline, etc. You can record anything you concern in this file.

GridInfoServer:
The GridInfoServer provides public services in DIRSS-G, like storing resource location information, some static resource descriptions and the user number. Its function is similar to Grid Information Services in realistic Grid environment. In Globus Toolkits, the function is achieved by MDS module.

4. Design of DIRSS-G

4.1 Logical Design

The architecture of DIRSS-G is decentralized. Choices between users and resources are bi-directional. The logic design for DIRSS-G is shown in Fig. 1.

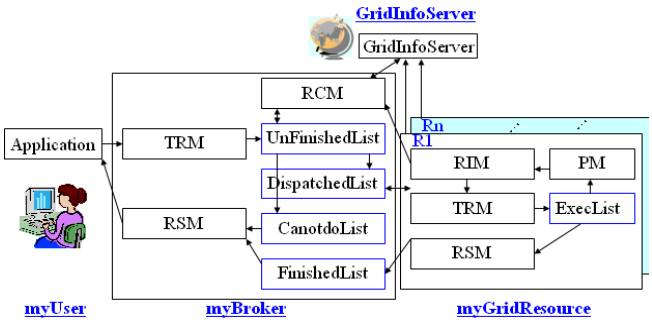


Fig. 1 Logical Design of DIRSS-G

TRM: task-receiving module RSM: result-submitting module RCM: resource-choosing module RIM: resource information module PM: pricing module.

MyBroker module consists of result-submitting module, task-receiving module, resource-choosing module, and four task queues. Four task queues are, UnfinishedList queue which recodes tasks needed completing; DispatchedList queue that recodes tasks which are already submitted but haven't finished; CanotdoList queue that recodes tasks which already exceed their deadlines; and FinishedList queue which records tasks finished already.

DIRSS-G is a message-driven system. To assure task QoS, we implement the QoS guaranteeing mechanism through bi-directional choosing. When a user submits a task to a resource, the resource should decide whether accepts it or not. The decision is made by present running information of the resource to assure quality of tasks running on it already. A task's quality demand will be assured as long as it is accepted by a resource.

4.2 Running Steps of myGridResource

1. A resource is initialized and registers itself in GridInfoServer.
2. When a task is submitted to it, the resource will decide whether accepts it or not. And then sends reply to the task's owner.
3. When some events occur, the pricing module will update its price. In our experiment, the event set is {accepting a new task, a task finished, passing a specific time slice}. The set can be set freely.
4. It executes tasks in its ExecuteList queue. When a task finished, the resource compiles its result, sends the result and updates its ResMoney.
5. It just needs to send a quitting message to GridInfoServer for quitting.
6. The resource continues doing step2-step5 until an experiment is over.

4.3 Running Steps of myUser

1. A user is initialized and reports itself to GridInfoServer. It submits all tasks to its myBroker. The myBroker adds these tasks to its UnFinishedList queue.
2. MyBroker firstly requests location information of all available resources registered in GridInfoServer, and then requests running information of these resources when some events occur. In our experiment, the event set is {submitting a task}.
3. MyBroker submits every task in its UnFinishedList queue to a suitable resource.
4. It waits for replies sent by elected resources. If a reply is an accepting message, it will delete the task from UnfinishedList queue and add the task to DispatchedList queue. If a reply is a rejecting message, the myBroker will submit the task again. If a task misses its deadline, it will be added to the myBroker's CanotdoList queue, and the myBroker will not submit it anymore.
5. When the myBroker receives a task's result, the myBroker adds the task to its FinishedList queue, updates its UserMoney and sends the result to the user.
6. The myBroker continues step3-step5, untill every task is completed, or misses its deadline, or UserMoney of the myBroker equals zero. Then the user sends ending message to GridInfoServer and quits from the system.

4.4 Running Steps of GridInfoServer

1. The GridInfoServer is initialized and set its resource queue empty at first.
2. The GridInfoServer waits for messages from other entities. If a message is a registration message, it will register the entity. If a message is a quitting message, it will delete the entity from the system. If a message is a requesting information message, it will send its running information to the sender.
3. The GridInfoServer periodically sends "KeepAlive" messages to resources recorded in its resource queue to check whether these resources are alive.
4. The GridInfoServer continues step2, step3, until the administrator of the GridInfoServer shuts down it.

When the GridInfoServer receives quitting messages from all users, an experiment is over. The sequence diagram of DIRSS-G is shown as Fig. 2.

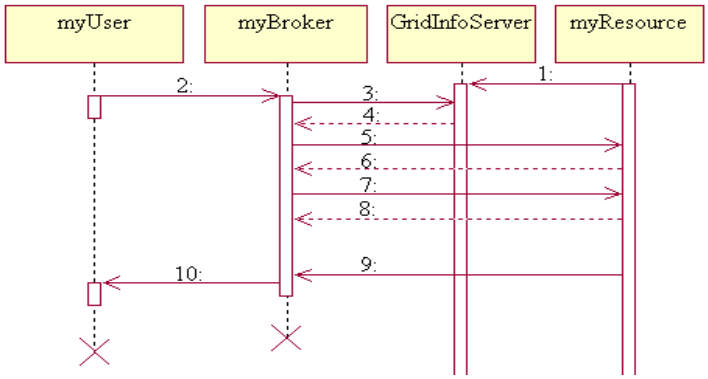


Fig. 2 Sequence Diagram of DIRSS-G: 1.register; 2.submit tasks; 3.request resource location information; 4.reply of 3; 5.request running information; 6.reply of 5; 7.submit a task; 8.reply of 7; 9.result; 10.result

5. Experiments and Results

5.1 Simulation Platform

We choose GridSim [5] as the simulation platform, which is a simulator for Grid environment. Through rewriting some interfaces in it, we can define our own users and resources with their special strategies. Furthermore GridSim is based on SimJava [6], which utilizes multi-thread mechanism in Java to simulate entities in discrete events. So they are suitable for simulating diverse and discrete events in Grid environment. Because of the implement mechanisms of GridSim and SimJava, using GridSim to simulate resource scheduling in Grid environment is feasible and suitable. And the simulating system provides easy access to users through graphic interface shown in Fig. 3.

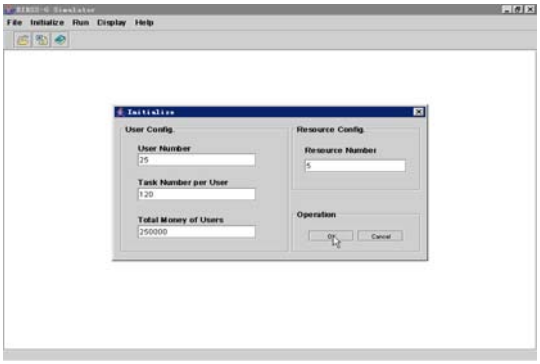


Fig. 3 The interface of the simulating system

5.2 Experiment Configuration

For simplicity, in our experiment, each user has the same resource choosing strategy: cost optimization algorithm, and each resource has the same pricing function, formula 1, and its task scheduling is Round Robin scheduling. Of course, we can configure DIRSS-G with other special requirements.

5.3 Experiment Data

To identify the performance of DIRSS-G, a simulation experiment is made based on the resources in Hanhai Grid testbed environment. For simplicity, all tasks are assumed to be independent and have no communications and data exchanges with each other. We set 25 users, and a modeled of task

farming application is established which consists of 25*120 tasks packaged containing the parameters needed by myBroker. We simulated 5 resources shown in Table 1. We assume that the costs of resources shown in Table 1 are their prices of low-load.

Table 1 Resource Information

Resource Name	Resource ID	Operating System	CPU Number	GHZ/ CPU	Gdollar/ Second
HP superdome	R0	HP-Unix	32	2.8	1300
HP server 128	R1	RedHat	32*2	1.5	1000
Cluster	R2	Debian	10	0.866	800
Dawn 2000	R3	AIX	128	0.375	700
SGI origin 2000	R4	IRIX	8	0.25	600

5.4 Analysis of Result

Because experiments are based on the multi-thread mechanism in Java, users and resources run randomly, and results of experiments are random too. So the experiment result cited later is the average of many experiment results with the same configuration.

5.4.1 Analysis of Task Accomplishment Ratio

In the same task set, task accomplishment ratios in DIRSS-G and in Nimrod/G are shown in Table 2. Because of the information-requesting mechanism and bi-directional choosing mechanism in DIRSS-G, its task accomplishment ratio is greater than that in Nimrod/G by 22.5%. But a myBroker needs to request dynamic running information of resources and choose a suitable resource to submit tasks according to special algorithm, so time cost affects the task accomplishment ratio little.

Table 2 Task Accomplishment Ratio in DIRSS-G and in Nimrod/G

System Name	Nimrod/G	DIRSS-G
Total Number of Tasks	25*120	25*120
Complete Number of Tasks	2195	2872
Task accomplishment ratio	73.2%	95.7%

5.4.2 Number of Tasks Completed on Each Resource

Fig. 4 is the histogram comparing the number of tasks completed on each resource in DIRSS-G with that in Nimrod/G. We can observe that, in the same task set, a very few resources completed most tasks with full load in Nimrod/G, while other resources are idle. Owing to some influences, such as task deadlines and resource capabilities, although R4's price is lower than any other resources, the number of tasks accomplished on R4 is not the greatest for its low capability. In DIRSS-G, tasks don't concentrate on a few resources. When R4 is over loads, tasks will be submitted to other resources. Because R2's capability is comparatively great and its price is moderate, it completes the most tasks of all resources. In DIRSS-G, for time cost, R2 completed most tasks, and R3, R4 follow on.

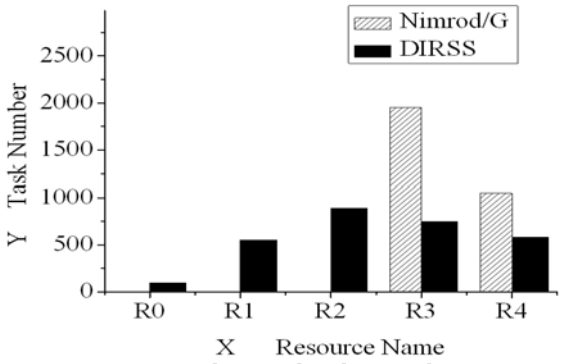


Fig. 4 Task Completed on Each Resource

5.4.3 Price Fluctuation of One Resource

The price fluctuation of each resource varies with its load. So a resource price reflects its load information. When a resource price increases to a certain value, tasks do not choose the resource. Then its price will decrease. Over a certain period, tasks will choose it again, causing its price to increase again. This process continues until an experiment is over. Due to its expensive minPrice and its great capability, R0 completes fewer tasks than any other resources, and its load and price remain steady. Not losing generality, we depict the process of R2 in Fig. 5.

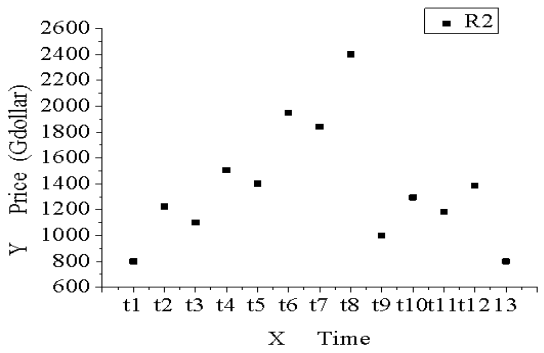


Fig. 5 Price Fluctuation of R2

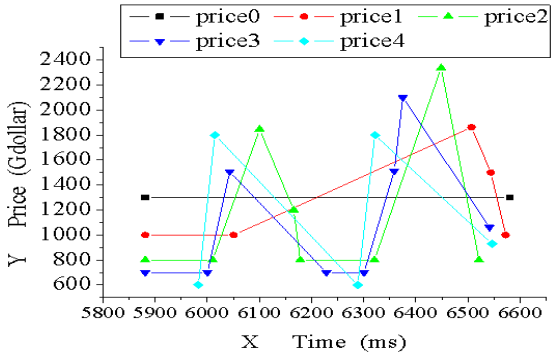


Fig. 6 Price Fluctuation of Each Resource

5.4.4 Price Fluctuation of All Resources

The price fluctuation process of all resources in a period is shown in Fig. 6. The price of each resource starts at its minPrice, and prices follow the rule mentioned in section 5.4.3. Since minPrices of R1, R2, and R3 increase sequentially, we can find that prices of R1, R2, and R3 begin to increase sequentially too. Moreover capabilities of R1, R2, and R3 increase sequentially, growth speeds of their prices decrease sequentially. And we can know it by comparing slopes of price lines. The price fluctuation of R0 is depicted in 5.4.3 previously.

6. Conclusion and Future Works

In this paper, we design and simulate a resource scheduling system, DIRSS-G in Grid environment. The experiment result shows that, DIRSS-G could solve the problem of load balance well; at the same time guarantees task QoS with good scalability and flexibility.

Our future researches mainly focus on two aspects. On the one hand, we'll study resource choosing strategy of users and pricing function of resources. On the other hand, we'll deployed DIRSS-G in the Hanhai Grid environment, which is based on China Nation Grid Software-VEGA

GOS[7]. DIRSS-G will provide transparent services to users through portals, effectively integrating resources in USTC.

References

[1] I. Foster, C. Kesselman, and S. Tuecke. "The Anatomy of the Grid: Enabling Scalable Virtual Organizations". The International Journal of High Performance Computing Applications, 2001, 15(3): 200-222.

[2] R. Buyya, H. Stockinger, J. Giddy, and D. Abramson. "Economic Models for Management of Resources in Peer-to-Peer and Grid Computing". Proceedings of the SPIE International Conference on Commercial Applications for High-Performance Computing. Denver, CO, U.S.A., August 20-24, 2001, Vol. 4528: 13-25.

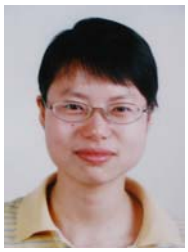
[3] O. Regev, N. Nisan. "The POPCORN Market – an Online Market for Computational Resources". Proceedings of the first International Conference on Information and Computation Economies. Charleston, SC, U.S.A., October 25-28, 1998: 148-157.

[4] D. Abramson, J. Giddy, L. Kotler. "High Performance Parametric Modeling with Nimrod/G: Killer Application for the Global Grid?". Proceedings of the 14th International Parallel & Distributed Processing Symposium. Cancun, Mexico, May 1-5, 2000: 520-528.

[5] R. Buyya, M. Murshed. "GridSim: a toolkit for the modeling and simulation of distributed resource management and scheduling for Grid computing". Concurrency and Computation: Practice and Experience, 2002, 14: 1175-1220.

[6] F. Howell, R. McNab. "Simjava: a discrete event simulation package for Java with applications in computer systems modeling". Proceedings of First International Conference on Web-based Modeling and Simulation. San Diego, CA, U.S.A., January 11-14, 1998: 51-56.

[7] China Nation Grid. http://vega.ict.ac.cn



Li Lu is a third-year MS student in the Department of Computer Science and Technology at the University of Science and Technology of China. She graduated from the Department of Computer Science at the HuaZhong Normal University in 2003, receiving a dual degree in Computer Science and Mathematics and Applied Mathematics. Her current research interests are Grid and Grid Computing, including the task scheduling and the scalability in Grid, and Web Services.

Shoubao Yang is a Professor and Ph.D. Supervisor in the Department of Computer Science and Technology, and the Director of USTC Network and Information Center, University of Science and Technology of China. At the same time, he is a Senior Member of China Computer Federation. His interest areas include: Grid and Grid Computing, Next Generation Network Protocol and Mobile Computing, Modern Cryptography and Network Security.