ORIENTAL JOURNAL OF COMPUTER SCIENCE & TECHNOLOGY
An International Open Free Access, Peer Reviewed Research Journal
www.computerscijournal.org

ISSN: 0974-6471
December 2011,
Vol. 4, No. (2):
Pgs. 305-315

An Incentive-based Peer-to-Peer Grid Scheduling

K.SRINIVASA RAO* and M.V.S.N. MAHESWAR

Assistant Professor, Department of Computer Science, Bhimavaram Institute of Engineering & Technology, Bhimavaram, Affiliate To JNTUK, Kakinada - 534 243 (India).
Assistant Professor of CSE Department PVPSIT, Vijayawada, Affiliate To JNTU, Krishna (District) A.P. - 520 00 (India).
*Corresponding author: E-mail: ksr9_kgrl@yahoo.co.in

(Received: July 26, 2011; Accepted: August 29, 2011)

ABSTRACT

In a grid commuting environment, resources are autonomous, wide-area distributed, and they are usually not free. These unique characteristics make scheduling in a self-sustainable and market-lime grid highly challenging. The goal of our work is to build such a global computational grid that every participant has enough incentive to stay and play in it. There are two parties in the grid: resources consumers and resource providers. Thus the performance objective of scheduling is two-fold: for consumers, high successful execution rate of jobs, and for providers, fair allocation of benefits. We propose an incentive-based grid scheduling, which is composed of a P2P decentralized scheduling framework and incentive-based scheduling algorithms. We present an incentive-based scheduling scheme, which utilizes a peer-to-peer decentralized scheduling framework, a set of local heuristic algorithms, and three market instruments of job announcement, price, and competition degree. The results show that our approach outperforms other scheduling schemes in optimizing incentives for both consumers and providers, leading to highly successful job execution and fair profit allocation.

Key words: Grid Computing, Scheduling, Peer to Peer, Resource Consumers, Resource Providers.

INTRODUCTION

Grid computing which aims at enabling wide-area resource sharing and collaboration is emerging as a promising distributed computing parading. According to how they schedule. Computational jobs to resources, computational grids can be classified into two types: controlled and market-like grids. Both types involve sharing and collaboration among resource providers and resource consumers, and the scheduling schemes can be either centralized or decentralized. The key difference between the two lies in who makes scheduling decisions. In a controlled grid, the grid system decides when to execute which job on which resource. In market-like grid, such decisions are made by each resource provider/consumer, but all the individual participants utilize some market instruments such as price to achieve the grid system wide objectives.

This paper focuses on the scheduling problem in market-like computational grids. In particular, we address the issues of optimizing incentives for both resource consumers and resource providers so that every participant has

sufficient incentive to stay and play, leading to a sustainable market. The main challenge, phrased as a scheduling problem, is to schedule jobs of consumers to resources of providers to optimize incentives for both parties.

The advantage of P2P is that a lot of otherwise unused resources can be harvested for the development of science, engineering, and business. Different types of P2P systems have been developed to support file sharing, distributed computing, collaboration, searching instant messaging.

One of the most important tasks of any grid coordinator is an effective allocation of jobs to available resources.

Although different optimization heuristics might be applicable here, not all of them satisfy particular Grid Scheduling requirements (such as necessity to operate in fully-automated mode). Our current project is aimed to investigate the performance of different Grid Scheduling algorithms and find the most suitable one for the practical use.

With the rapid development of high-speed wide-area networks and powerful yet low-cost computational resources, Grid computing has emerged as an attractive computing paradigm. In this paper the goal is realized by building a computational market supporting fair and healthy competition among consumers and providers. Each participant in the market competes actively and behaves independently for its own benefit. A market is said to be healthy if every player in the market gets sufficient incentive for joining the market.

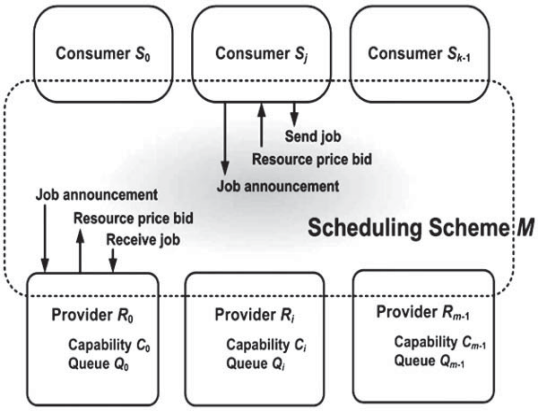

Fig. 1: Scheduling in the market-like computational grid

Decentralized computing systems are becoming increasingly popular as they enable organizations to use existing computing resources that otherwise lie idle. Whether this paradigm will be successful largely depends on the flexibility and easiness with which it can be implemented and managed.

Statement of the problem

We define a market-like computational grid as a four-tuple G=(R, S, J, M), as depicted in Fig. 1. The grid G consists of a set of m resource providers $R = \{R_{0,.....,} R_{m-1}\}$ and a set of k resources consumers $S = \{S_{0.....,} S_{k-1}\}$. Over a time period T, a set of n jobs $J = \{J_{0.....,} J_{n-1}\}$ are submitted to the grid by the consumers, scheduled by the scheduling scheme M, and executed by resources of the providers. The scheduling scheme M should employ market instruments to allow each provider and each consumer to make the scheduling decision autonomously. That is, each provider R_i can decide whether it would offer its resource, and each consumer S_j can decide whether it would use a certain resource to execute its jobs.

Consumers and Jobs

In this paper, we only consider computation-intensive jobs, where all communication overheads can be ignored. All jobs are independent of one another. The k consumers altogether have n jobs to execute in time period T. The consumers first submit job announcements to the computational grid. A job announcement includes the information of job length and job deadline.Job length is an empirical value assessed as the execution time of the job on a designated standard platform. Job deadline is a wall clock time by which a consumer desires a job to be finished, expressed as a number between 0 and T.

Providers and Resources

Each resource provider is modeled with three parameters: capability, job queue, and unit price. Capability is the computational speed of the underlying resource, expressed as a multiple of the speed of the standard platform. The job queue of a resource provider keeps an ordered set of jobs

scheduled but not yet executed. Each job, once it is executed on a resource, will run in a dedicated mode on that resource, without time-sharing or preempting. A provider charges for a job according to its unit price and job length.

Unit price refers to the price that the resource offers for executing a job of unit length.

Many incentive metrics can be defined for providers, but a fundamental one is the fairness of the market, where each provider has equal opportunity to offer its resource, and it can obtain a fair profit according to its capability. Fare allocation of profits, by which we mean that a provider is allocated the same share of profit as that of the capability that it invests to the market, is attractive to all potential providers, including not only those with higher capabilities but also those with lower capabilities.

Related work

Much attention has been devoted to the area of scheduling in distributed computing[2-18]. However, to the best of our knowledge, there is still no work investigating effective scheduling to optimize incentives for both consumers and providers, utilizing market information.

Many previous research projects focused on optimizing traditional performance metrics, like system utilization, system load balance, and application response time in controlled grids. They did not consider market-like grids, where providing sufficient incentives for participants is a key issue. Many projects[7-18].Have investigated the effectiveness of introducing economic models and theories into distributed resource scheduling. Researches in [12] and [17] study incentives for participants to behave honestly. Our work learned from these researches, these researches only consider consumer objectives or provider objectives, whereas we focus on optimizing incentives for both consumers and providers.

Enterprise[7]]is a task scheduler for distributed market-like computing environments.

The work shows the effectiveness of bidding for a decentralized scheduling framework. Spawn [8] is a market-based computational system that utilizes idle computational resources in a distributed network of heterogeneous computer workstations. The auctions employed by Spawn are sealed-bid second-price auctions. Buyya *et al.,*[9] identify the distributed resource management challenges and requirements of economy-based grid systems and discuss various representative economy-based systems. They also present commodity and auction models for resource allocation. The evaluation results of computational and data grid environments demonstrate the effectiveness of economic models in meeting users' QoS requirements. We choose a consumer initiated bid model in our work.

Compute P2P[17] is an architecture for enabling Internet computing using peer-to-peer (P2P) networks for haring of computing resources. The work focuses on modeling pricing with the game theory and microeconomics to deal with selfish behavior and proves that its model guarantee the incentive for all the providers to share resources and not to cheat. SHARP[18] is a framework for secure distributed resource management in an Internet-scale computing infrastructure. Its main focus is not scheduling but the security mechanisms that protect against various threats and vulnerabilities. Their definition of the oversubscription degree (OD) bears some similarity to the CD in our work. However, OD takes effect when providers issue tickets, and CD takes effect after providers bid for jobs. In addition, we consider penalty as a feedback to help adjust CD and focus on achieving fairness among providers.

Partial results of our incentive-based scheduling work are reported in [19] and [20]. In [19], consumers assign budgets to jobs and choose providers according to the claimed completion time. No price or CD mechanisms are investigated. In[20], the impact of CD is studied. It does not formulate the dual-objective scheduling problem, develop a complete scheduling scheme, evaluate performance in detail, or provide quantitative comparison with related work, as what the current paper does.

The incentive-based scheduling scheme

We define an incentive-based scheduling scheme with heuristics, employing a P2P decentralized scheduling framework. The scheme

is characterized as follows: 1) each consumer or provider autonomously makes scheduling decisions, 2) all scheduling algorithms are local to a resource provider, and 3) three market instruments, job announcement, price, and CD, are used.

Peer-to-Peer Scheduling Framework

Our scheduling framework takes advantage of the P2P technology, utilizing its characteristics of decentralization and scalability. Aside from that, as every participant in the computational grid is autonomous acts individually, a decentralized scheduling infrastructure is more favorable. Furthermore, owing to the dynamics of grid environments, players may enter or leave at will at any time. A P2P network can handle such dynamics.

A consumer submits a job announcement to the computational grid via one portal. Then, the job announcement spreads throughout the P2P network, similar to query broadcast in an unstructured P2P system. The providers that receive a job announcement may bid for the job. We want to realize the complete competition among all the providers based on two considerations. First, the job execution time is sufficiently long such that the overhead of executing them on remote computers becomes relatively negligible. Thus, all the providers should have an equal chance to compete for any job, no matter where their geographical locations are. Second, the number of providers will not be too large, typically not more than several hundreds,

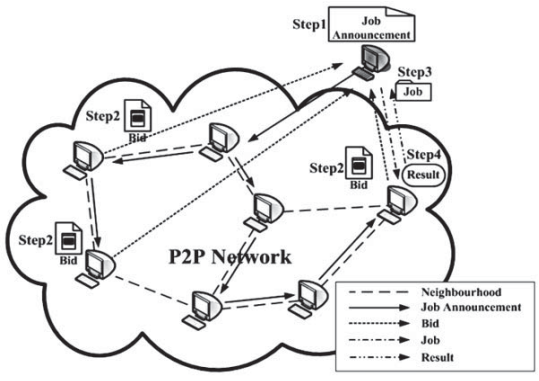

Fig. 2: The steps that a single Job goes through in the incentive-based scheduling scheme

for a provider represents an administrative domain, within which local scheduling policies are employed.

The P2P scheduling infrastructure enables the effective interactions between consumers and providers, and jobs are scheduled as a result. Fig. 2 depicts the complete sequence of steps that a single job goes through in the scheduling scheme M. All jobs from consumers follow the same steps:

Step 1

A consumer submits a job announcement to the computational grid, and the job announcement is broadcast to all the providers.

Step 2

Each provider, upon receiving a job announcement, estimates whether it is able to meet the deadline of the job. If yes, the provider sends a bid that contains the price for the job directly back to the consumer; otherwise, the provider ignores the job announcement.

Step 3

After waiting for a certain time, the consumer processes all the bids received, chooses the provider who charges the least, and sends the job is finished, the provider sends the result to the consumer.

Step 4

The provider who receives the job inserts it into its job queue. When the job is finished, the provider sends the result to the consumer.

Incentive-based scheduling algorithms

We design four algorithms for providers.The job competing algorithm describes how a provider bids when receiving a job announcement in step 2. The heuristic local scheduling algorithm is responsible for arranging the execution order of jobs in the job queue of a provider. It starts when a provider receives a job in step 4. The price-adjusting algorithm and the CD-adjusting algorithm help a provider in dynamically adjusting its unit price and CD properly over the period of its participation in the computational grid.

Job Competing Algorithm

If bidding, one job likely misses its deadline

when both jobs are received. Things get more complex when more jobs are involved. There are two extreme attitudes for providers to compete for

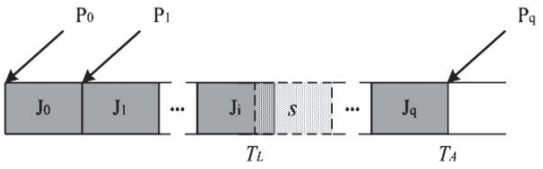

Fig. 3: Job queue of a provider

jobs. One is aggressive. It means that a provider never considers the unconfirmed jobs when estimating whether it is able to meet job deadline. This is a risky one, but chances often accompany risks. The other is conservative. It means that a provider always keeps the unconfirmed jobs in the job queue for consideration for a certain time. This attitude will never lead to deadline missing but may lose potential chances and, thus, profits. Different competition attitudes will result in different allocations of profits. To study the impact of competition attitude, we define a parameter named CD, a real number from 0 to 1. A provider will insert unconfirmed jobs into its job queue at the probability of 1-CD.

For example, for aggressive providers, CD is 1. For conservative ones, CD is 0. For providers who just toss up, CD is 0.5.

Step 1

The provider estimates whether it is able to meet the job deadline.

As Fig. 3 shows, there are q jobs in the job queue. If we call the potential new job as s,P_0,P_1,……., and P_q represent the q+1 possible places for s to be inserted into. T_A, the available time, is the time instance when all the jobs in the job queue are completed. T_L is calculated by subtracting the execution time of s from the deadline of s. It is the latest time to begin the execution of s if the provider does not want to let s miss its deadline. The estimation is described with the following algorithm:

```
1    if T_L>T_A then
2    can-meet ←true;
3    reordered ←false;
4    insert-place ←P_q;
5    else // T_L is covered by the execution of J_i in
     the queue
6    if insert s at P_i-1, none of J_i ~ J_q will miss its
     deadline then
7    can-meet ←true;
8    reordered ←true;
9    insert-place ←P_i-1;
10   else
11   can-meet ←false;
12   end if
13   end if
```

Step 2

The provider offers a price for the job. The pseudo code is given as follows.

```
1    price ←p*L_s;
2    if reordered then
3    price ←λ * price;
4    end if
```

Here, p is the unit price of the provider, L_s is the job length of job s, and l is a decimal slightly larger than 1. When the variable reordered is set to true, the price is raised. To meet job deadlines, some jobs may be inserted into the job queue ahead of foregoing jobs, which indicates that the deadlines of these jobs are somewhat tight and the jobs need to be given higher priority. Step 3. The provider sends the price as a bid and inserts the job at the place that the variable inset place indicates at the probability of 1-CD. If the provider chooses to insert and the job does not come after a certain time, it deletes the job from its job queue. The duration of keeping an unconfirmed job should be as short as possible but long enough to guarantee not to delete offered jobs.

Heuristic Local Scheduling Algorithm

We employ a punishment mechanism that providers are obliged to pay penalty for missing job deadlines. A simple linear penalty model is used. The amount of penalty is proportional to the exceeding time, that is, how much time the completion time T_C exceeds the deadline T_D.

Once the penalty model is introduced, providers must take some measures to minimize the loss. What a provider can do is to arrange the execution order of jobs in CD is exits job queue. The approach is based on the heuristic rule that

when a job is inserted, the relative order of the jobs in the origin queue is unchanged. Every time a provider is offered a job that is not kept in the job queue, it starts the heuristic local scheduling algorithm. The algorithm is needless for providers whose CD equal to 0, because they always keep unconfirmed jobs. The heuristic local scheduling algorithm is:

```
1    insert place ←P_q;
2    penalty ←calculate the penalty of inserting
     the job ←at P_q;
3    for i ←q-1 to 0 do
4    penalty i ←calculate the penalty of inserting
     the Job at P_i;
5    if penalty i < penalty then
6    penalty ←penalty i;
7    insert-place ←P_i;
8    end if
9    end for
10   insert the job at inset_place
```

Price-Adjusting Algorithm

Because a consumer chooses the cheapest provider, it is the price mechanism that directs almost the whole scheduling. After all, computer systems are very different from human economies in many aspects [8] first, human decision making is difficult to model, and human beings are diverse in their methods for decision making. Second, human beings make decisions based on the information from the whole society via various media. However, it is not the case for computer systems. Therefore, enabling computer systems to exhibit meaningful market-like behavior is still an open problem. To emphasize the main idea, we do not take great pain trying to model the price mechanism of the real market. Instead, we design a simple and intuitive price-adjusting algorithm.

As our performance objective for providers is the fair allocation of profits, it involves all the providers. It is almost impossible to be realized if every provider just behaves based on the local information. Inevitably, all the providers need to know some global information. In our algorithm, we assume that every provider is informed with the aggregated capability of all the providers in the computational grid. The information can be acquired when a provider enters the grid via a portal and is updated in the dame way that a job announcement is forwarded.In a certain period of time, every commodity has a predominant price in the market. For a commodity like CPU cycles, such a price is easier to determine, because commodities of this kind do not have great difference in quality. We call the price as market price, and it acts as a directive. When entering the grid, a provider gets the market price from a portal and sets it as the initial unit price. Then, every time a provider is offered a job or deletes an unconfirmed job, it starts the price-adjusting algorithm.

```
1    r1 ←L_O/L_T;
2    r2← C/Σ_{0≤j<m} C_j;
3    if offered a job then
4    if r1 > r2 and p<=P_M then
5    p ←α*p;
6    endif
7    else // delete an unconfirmed job
8    if r1 < r2 and p >=P_M then
9    p← β*p;
10   endif
11   endif
```

L_O, which is the offered job length, is the aggregated length of jobs offered to the provider. L_T, which is the total job length, is the aggregated length of jobs whose announcements are received by the provider. $\Sigma_{o\leq j<m} C_j$ is the aggregated capability of all the providers. The offered job length and the total job length rewind when the total capability is updated. In addition, C and p are the capability and unit price of the provider, respectively, P_M is the market price, α is a decimal above 1, and β is a positive decimal under 1. Our price-adjusting mechanism is simple and intuitive: just to make prices different, differentiate the chances of providers to be chosen and eventually realize the fair allocation of profits.

Competition-Degree-Adjusting Algorithm

The more conservative ones are relatively less competitive than the more aggressive ones. They always keep unconfirmed jobs in their job queues and tend to lose potential jobs because of being unable to bid. Most likely, these jobs are offered to the more aggressive ones. As a result, fairness among all the providers is hard to achieve. Moreover, the jobs that could have been done by the conservative ones may bring the aggressive ones not only profit but also penalty, of course, which

results from deadline missing. A wise provider, whether a conservative or an aggressive one, should never hold its attitude toward competition if things like that happen. It will adjust its CD according to the situation that it perceives. Thus, we design the CD-adjusting algorithm.

//Every time the penalty increases
1. if $R_p >= TH_p$ and CD >=ε then
2. CD←CD-e;
3. endif

//Every time a certain interval such as 1 day
1. if $R_p < TH_p$ and $R_j >= TH_j$ and CD <=1-ε then
2. CD←CD+ε;
3. endif

Here R_p is the ratio of penalty to profit, and R_j is the ratio of jobs that the provider does not bid for. TH_p and TH_j are thresholds for them, respectively. If one rate gets above its threshold, CD is adjusted accordingly at the step of ε. As can be seen, the check of R_p is not only timelier but also prior.The reason is that the rate of penalty to profit is a more obvious index to providers. Thus, R_p is checked every time the penalty increase in penalty in time, whereas interval such as 1 day.

Performance evaluation

We develop a discrete event-driven simulator with the Java programming language to simulate the computational grid. Consumers and providers are modeled as two kinds of entities in the simulation system. The communications between them are performed by event delivery. As for the advance of simulation time, there are mainly two drives: one is the network delay of communications, and the other is job execution. We ignore delay in the simulator but focus on implementing the algorithms and evaluating them. We do four experiments: the first investigates the impact of CD on performance, the second analyzes the incentive-based scheduling scheme by disabling the CD-adjusting algorithm or the price-adjusting algorithm, and the last two compare our scheme with four other schemes under synthetic workloads and real workloads, respectively. Altogether, our simulator scheduled over 30 million jobs to generate the results of the four experiments including about three million for the experiment under real workloads.

Simulation results

System Load

As viewed from mathematics, system load can be defined as a ratio of aggregated load to aggregated capability in a period of time. The following equation gives a formal explanation.

$$System\_Load = \frac{a\_a\_r * a\_j\_l * n\_c}{a\_c * n\_p}$$

a _ a _ r : average arrival rate of jobs
a _ j _l : average length of jobs
n _ c : number of consumers
a _ c : average capability of providers
n _ p : number of providers

Consumer and Provider

Consumers independently generate jobs from time to time. For each consumer, job generation is modeled as a poisson process. Thus the interval between two job generations is exponentially distributed. The mean is the reciprocal of average arrival rate of jobs and can be calculated with the aforementioned system load definition equation. The job length is uniformly distributed, and the duration from the time instance of job generation to the deadline is uniformly distributed as well.

The capability of providers is normally distributed due to the observation that during a certain time, computers of some capability predominate in the computer market. To study the impact of CD on scheduling performance, we devise the simulator so that one simulation can work out results of different CD configurations, while other configurations are the same including length, deadlines order and arrival intervals of jobs, and capabilities of providers.

Simulation Results

In our simulation experiments, there are in total 20 consumers and 80 providers. Lengths of jobs average 100, and capabilities of providers average 10. λ is assigned as 1.05, α as1.1, and β is0.9. Met price is 1. System load of simulations varies from 0.1 to 0.7 with step of 0.1. Every simulation runs as long as 110 days in simulation time, working out results of three different CD configuration: 0,0.5, and 1.

First, we study the incentive of consumers. Our performance objective for consumers is high successful execution rate of jobs. There are two related metrics: failure rate and deadline missing rate. A job fails because all the providers think that they cannot meet the deadline and decide not to bid. The definition of the two matrices is listed below.

$$\text{Failure\_rate} = \frac{n\_j\_fail}{n\_j\_submitted}$$

Table: 1 Failure rates of jobs

System Load	CD = 0	CD = 0.5	CD = 1
0.1	0.18%	0.00%	0.00%
0.2	2.16%	0.00%	0.00%
0.3	5.09%	0.01%	0.00%
0.4	7.12%	0.02%	0.00%
0.5	8.91%	0.05%	0.00%
0.6	10.27%	0.11%	0.00%
0.7	12.12%	0.27%	0.00%

Table 2: Deadline Missing Rates of Jobs

System Load	CD = 0	CD = 0.5	CD = 1
0.1	0.0000%	0.0000%	0.0000%
0.2	0.0000%	0.0000%	0.0002%
0.3	0.0000%	0.0001%	0.0004%
0.4	0.0000%	0.0003%	0.0022%
0.5	0.0000%	0.0038%	0.0078%
0.6	0.0000%	0.0079%	0.0431%
0.7	0.0000%	0.1290%	0.3631%

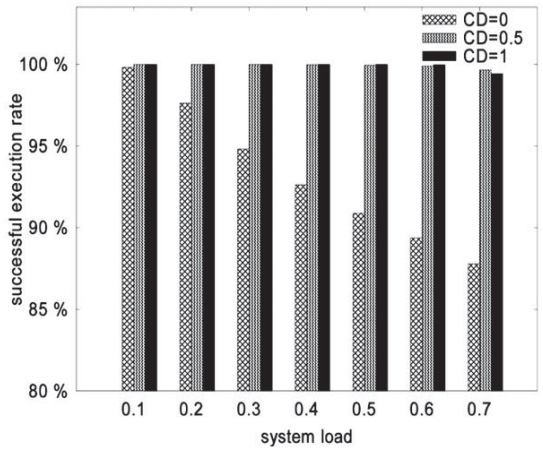

Fig. 4: Successful-execution rates of jobs

$$\text{Deadline\_missing\_rate} = \frac{n\_j\_miss}{n\_j\_finished}$$

n _ j _ fail : number of jobs that fail
n _ j _submitted : number of jobs submitted
n _ j _miss : number of jobs that miss deadline
n _ j _ finished : number of jobs finished

The failure rate ζ is defined as the ratio of the number of jobs that fail to n, which is the number of jobs submitted to the computations grid. The deadline missing rate η is defined as the ratio of the number of jobs that miss their deadlines to the number of jobs that the grid accepts and executes.

Table 1 and Table 2 show the simulation results of the two metrics. Table 1, we can see that if providers are extremely aggressive, that is CD is equal to 1, all the jobs submitted by consumers can be executed. If providers are not so aggressive, job failure happens, for providers reserve for unconfirmed jobs and tends to fail to meet the deadlines of those coming later. When the conservation comes to an extreme that is CD is equal to 0, the failure matrices increases greatly when the system load gets heavier. When the

Table 3: Total Penalty of Providers

System Load	CD = 0	CD = 0.5	CD = 1
0.1	0.00	0.00	0.00
0.2	0.00	0.00	0.18
0.3	0.00	0.97	2.83
0.4	0.00	9.75	19.09
0.5	0.00	50.00	86.40
0.6	0.00	375.06	625.00
0.7	0.00	2970.00	6425.89

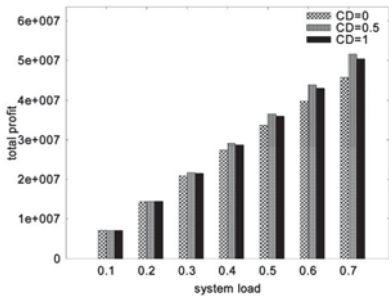

Fig. 5: Total profit of providers

system load is 0.7, the failure rate is above 10 percent. Table 2 s about the deadline missing rate. When CD is equal to 0, jobs executed never miss their deadlines. As providers get more aggressive and the system load is heavier, the deadline missing rate increases. However, even when the system load is 0.7 and CD is equal to1, the deadline missing rate is under 1 percent. The successful execution

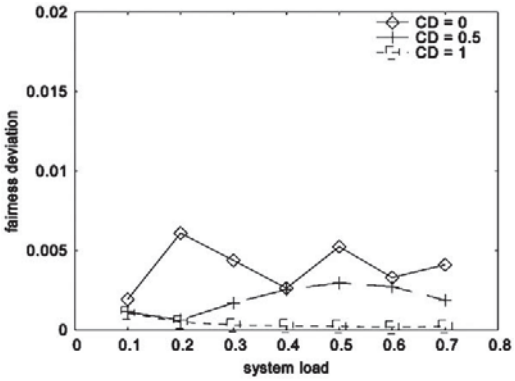

Fig. 6: The fairness deviation of providers

rate θ can be calculated with the following:

$$\theta = (1 - \zeta) * (1 - \eta)$$

For providers, first we study how CD impacts the total penalty, sum of penalty paid by all the providers. Table 3 shows the results. Apparently, as CD and the system load increase, the total penalty gets bigger, which is consistent with the observation about the deadline missing rate? The more meaningful metric is the net profit.

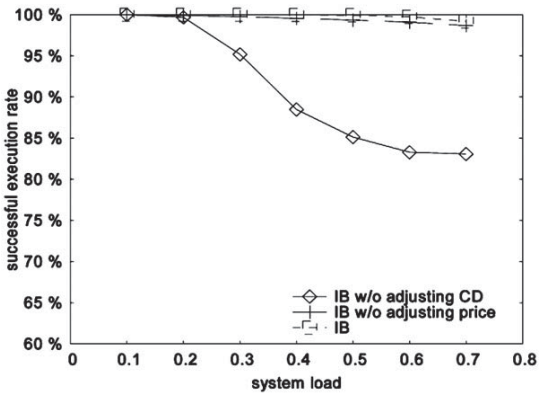

Fig. 7: Analysis of IB on the successful-execution rate

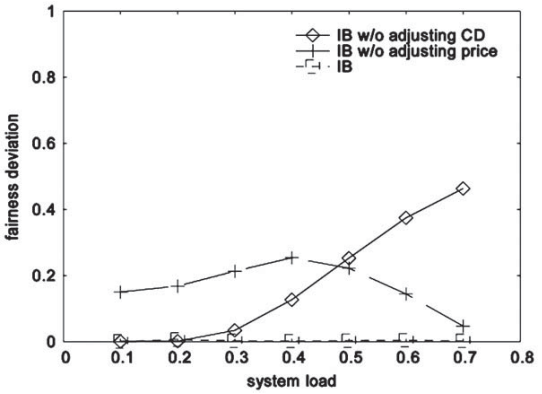

Fig. 8: Analysis of IB on the fairness deviation

Fig. 5 shows the simulation results of this. Conservative providers never need to pay the penalty, but they lose potential jobs because of their conservative attitude. Thus, the total profit\it is not very satisfactory. The results show that a trade-off attitude, that is, CD is equal to 0.5, is superior to the extreme two. Our performance objective for providers is to minimize the fairness deviation ó. Fig 6 shows the results.

Fig. 6 indicates that aggressive competition can achieve the most desirable fairness. Our price-adjusting algorithm tries reasonable allocation the job length among all the providers. However, profits are related not only to the length of jobs offered but also to the price. Intuitively, when CD is equal to 1, providers compete aggressively with each other, and resources are always sold at a low base price. However, in the case that providers are more conservative, the price tends to be more diverse. Altogether, within the tested system load, the fairness deviation is under 0.01.

Study the effectiveness of the price-adjusting algorithm and the CD-adjusting algorithm. We work out the simulation results of the incentive-based scheduling scheme without adjusting the price and CD separately, and we compare them with that of the complete incentive based scheduling scheme. Fig. 7 and 8 show the results. It is obvious that the two algorithms complement each other and are both indispensable. When disabling the CD-adjusting algorithm and the system load is high, our incentive-based scheduling scheme achieves poor performance. It can be explained that those more aggressive providers keep getting more jobs,

more of which they fail to finish before their deadlines because of the heavy system load. A higher deadline kissing rate results in more penalty, finally lower successful-execution rate, and, worse, fairness.

CONCLUSIONS

In this paper, we develop an incentive-based scheduling scheme with heuristics, using a P2P decentralized scheduling framework and Incentive-based scheduling algorithms. This scheme has the following features:1) each consumer or provider autonomously makes scheduling decisions, 2) all scheduling algorithms are local to a resource provider, and 3) three market instruments, that is, job announcement, price, and CD, are employed, and the former two circulate in the grid. This paper has two major contributions. First, we have defined incentive as scheduling objective for a decentralized market-like computational grid, and the incentive is two-fold, not only for consumers but also for providers. Second, we have studied how the competition attitude of providers influences the performance of scheduling. With further work, The Scheduling Scheme performs only Java Programme, it can be also extend to other Programmes like Oracle, C, C++ etc.,. In future it can be developing MAN &WAN.

REFERENCES

1 Lijuan Xiao, Yanmin Zhu, Lionel M. Ni and Zhiwei Xu. "GridIS: an Incentive-based Grid Scheduling" Institute of Computing Technology, Chinese Academy of Science Beijing 10080, China (2005).

2 Berman F. Wolski, H. Casanova, W. Cirnr, H. Dail, M. Faerman, F. Figueira, J. Hayes, G. Obertelli, J. Schopf, G. Smallen,, N. Spring, "Adaptive Computing on the Grid Using Apples, IEEE Trans. Parallel and Distrubuted System, **14**(4): 369-382 (2003).

3 Malone T.W, Fikes R.E, Gran K.R, and Howard M.T, "Enterprise: A Market-Like Task Scheduler for distributed Computing Environments", The Ecology of Computation, B.A. Hubermen, ed., 177-205 (1988).

4 Buyya R. Abramson D. and Venugopal S. "The Grid Economy" Proc. IEEE, 93: 698-714 (2005).

5 Lai K. Rasmusson L. Adar E. Zhang L. and Huberman B.A. "Thcoon: An Implementation of a Distributed, Market-Based Resource Allocation System," *Multiagent and Grid Systems*, **1**(3): 169-182 (2005).

6 Xiao L. Zhu L.M.Ni. and Xu Z. "GridIS: An Incentive-Based Grid Scheduling," Proc. 19th IEEE Parallel and Distributed Processing Symp. (IPDPS'05), 65 (2005).

7 Liu Y. Zhuang Z. Xia, L. and Ni L.M. "A Distributed Approach to Solving Overlay Mismatching Problem," Proc.24th IEEE Int'l Conf. Distributed Computing Ssytems (ICDCS'04), 132-139 (2004).

8 Zhu Y. Xiao L. Ni L.M., and Xu Z., "Incentive-Based P2P Scheduling in Grid Computing," Proc. Of Grid and Cooperative Computing, **3251**/2004: 209 (2004).

9. Nazareno Andrade, Lauro Costa, Guilherme Germoglio, Walfredo Cirne., "Peer-to-Peer Grid Computing with OurGrid Community," Lanoratorio de sistemas Distribuidos – Universidade Federal de Campina Grande, Av. Aprigio Veloso, 882 Bloco CO, Bodocongo-58. 109-970, Campina Grande, PB, Brasil.

10. Saroiu S., Gummadi P.K., and Gribble, S.D., A Measurement Study of Peer-to Peer File Sharing Systems. In PJroc. Multimedia Computing and Networking 2002 (MMCN'02), san Jose CA, USA (2002).

11. Tan Tien Ping, Gian Chan Sodhy, Chan Huah Youg, Fazilah Haron and Rajkumar Buyya, "A Market-Based Scheduler for JXTA-Based Peer-to Peer Computing System,"School of Computer Science Universiti Sains Malaysia, 11800 Penang, Malaysia.

12. Vijay Subramani, Rajkumar Kettinuthu Srividya, Srinivasan P. Sadayappan, "Distributed Job Scheduling on Computational Grids using Multiple Simultaneous Requests," Proc. Of the 11th

IEEE International Symposium of High Performance Distrubuted Computing HPDC-11 20002 (HPDC'02), Scoltland (2002).

13. Parashar .M and Lee C.A. Proc. IEEE, special issue on grid computing, eds., **93**(3): 479-714 (2005).
14. Foster I. and Kesselman C, "Globus: A Meta Computing Infrastructure Toolkit" Int'l J. High-Performance Computing Applications, **11**(2): 115 (1997).
15. Litzkow M. Livny M. and Mutka M., "Condor: A Hunter of Idle Workstations," Proc. Eighth Int'l Conf. Distributed Computing systems (ICDCS'88), 104-111 (1988).
16. Chapin S.J. Katramatos D. Karpovich J. and Grimshawn A., " Resource Management in Legion," Future Generation systems, **15**(5-6): 583-594 (1999).
17. Casanova H. and Dongarra J., "NetSlove: A Network Server for Solving Computational Science Problems," Int'l J. Supercomputer applications and High-Performance Computing, **11**(3): 212-223 (1997).
18. Waldspurger C.A., Hogg T. Huberman B.A ., Kephart J.O. and Stometta S. "Spawn: A Distributed Computational Economy," IEEE Trans. Software Eng., **18**(2): 103-177 (1992).
19. Buyya R., Abramson D., and Venugopal S., " The Grid Economy," Proc. IEEE, **93**(3): 693-714 (2005).
20. Regev O., and Nisan N., " The POPCORN Market: An Online Market for Computational Resources," Proc. First Int'l Conf. Information and Computation Economics (ICE'98), 148-157 (1998).
21. Wolski R., Pland J.S., Bryan T., and Brevik J., "G-Commerce: Market Formulations Controlling Resource Allocation on the Computational Grid," Proc. 15th Int'l Parallel and Distributed Processing Symp. (IPDPS'01), 8 (2001).
22. Padala P., Harrison C., Pelfort N., Jansen., E., Frand M.P., and Chokkareddy C., "OCEAN: The Open Computation Exchange and Arbitration Network, a Market Approach to Meta Computing," Proc. Second Int'l Symp. Parallel and Distributed Computing, 185-192 (2003).
23. Sherwani J., Ali N., Lotia N., Hayat Z., and Buyya R., "Libra: A Computational Economy-Based Job Scheduling System for Clusters," Software: *Practice and Experience,* **34**(6): 573-590 (2004).
24. Irwin D.E., Grit L.E., and Chase J.S., "Balancing Risk and Reward in a Market-Based Task Service," Proc. 13th Int'l Symp. High Performance Distributed Computing (HPDC'04), 160-169 (2004).
25. Yun F., Jeffrey C., Brent C., Stejphen S., and Amin V., "SHARP: An Architecture for Secure Resource Peering," Proc. 19th ACM Symp. Operating Systems Principles (SOSP'03), pp. 133-148 (2003).