CHAPTER 6

INTEGRATED JOB AND RESOURCE MANAGEMENT SCHEME IN GRID SCHEDULING

6.1 INTRODUCTION

Grid computing, which aims at enabling wide-area resource sharing and collaboration, is emerging as a promising distributed computing paradigm Parashar and Lee (2005). Based on how computational jobs are scheduled to resources, the computational grids can be classified into two categories namely centralized and market-like grids. Both the types involve sharing and collaboration among resource providers and resource consumers and the scheduling schemes can be either centralized or decentralized as mentioned by Bahman Arasteh et al (2011). The key difference between the two lies in by whom the scheduling decisions are made. In a centralized grid, the grid system decides which job is to be executed?, when the job is to be executed? and which resource is to be made use of?. In a market-like grid, such decisions are made by each resource provider/consumer.

This chapter focuses on the scheduling problem in the market like computational grids. In particular, it addresses 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 scheduling problem, is to schedule jobs of consumers to resources of providers in the view of optimizing

incentives for the both parties. Most importantly, such objectives should be realized not by an omnipotent scheduler, but rather by the scheduling scheme which is autonomous. That is, each participant makes decisions on his own behalf and the individual economic behaviors of all participants work together to accomplish resource scheduling, with optimized incentives being an emergent property of the grid system Formulation of the above scheduling problem and investigation of market instruments and algorithms are done. Identification of the successful-execution rate of jobs as the incentive for consumers and the inverse of fairness deviation as the incentive for providers made. As even a sub problem of the formulated scheduling problem is NP-complete, this type of scheduling scheme called Integrated Job and Resource management (IJRM) scheduling using local heuristics is done. Job announcement, Independent Factor (IF) and price are defined and used as market instruments. Two heuristic algorithms, local to each participant, are developed to utilize the market instruments and to optimize the incentives. Performance evaluation is conducted via extensive simulations, utilizing both statistically generated workloads and real workloads. The results show that the proposed IJRM scheme outperforms other schemes in optimizing incentives for both consumers and providers.

6.2 PROBLEM FORMULATION

Defining a market-like computational grid as a quadruple G= (R, S, J, M). The grid G consists of a set of m resource providers R = {R0,..., Rm-1} and a set of k resource consumers S = {S0 .... Sk-1}. Over a time period T, a set of n jobs J = {J0,., Jn-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 Ri can decide whether it would offer its resource and each consumer Sj can decide whether it would use a certain resource to execute its jobs.

6.2.1 Consumers and jobs

In this chapter, computation-intensive jobs are considered and all communication/networking overheads are 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. Thus, a job with length = 10 and deadline = 100 means that the job's execution takes 10 time units on a designated standard computer and it must be finished 100 time units after the common base time 0.

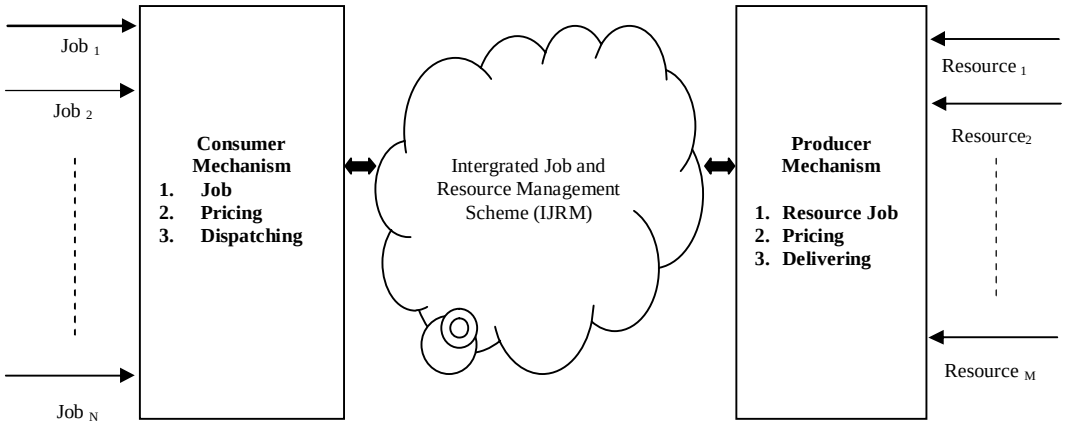

Figure 6.1 IJRM Scheduling

Providers and resources: From the scheduling viewpoint, 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 charges for executing a job of unit length. When a provider with capability of 5 bids to execute a job of length 20 at a unit price of 2(Based on economic pricing mechanism – Chapter 4) and if the consumer accepts the bid and decides to submit the job for a execution, the job will take 20/5 = 4 units of time to complete, generating a profit of $2 \times 20 = 40$ for the provider.

Economy for consumers and providers: Intuitively, consumers are attracted towards a grid, because it offers high quality of computational service at low cost. This could lead to many potential metrics of consumer incentives. However, a fundamental incentive requirement is that a grid should have a high successful-execution rate of jobs, where a successful job execution means that a job is executed without missing its deadline. When this rate is too low, even if the cost is zero (as in the case when a grid is advertising funded), the consumers will lose faith in the grid and quit it. Therefore, the successful-execution rate of the grid system is considered as the incentive for consumers.

6.3 RELATED WORKS

Much attention has been devoted to the area of scheduling in distributed computing Sujai Mistry et al., (2011). 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.

Enterprise is a task scheduler for distributed market like computing environments. The work shows the effectiveness of a bidding model for a decentralized scheduling framework. Spawn 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. Kurt Vanmechelen et al (2008) 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. A consumer initiated bid model is chosen based on chapter 4.

CompuP2P Sujai Mistry et al (2011) is architecture for enabling Internet computing, using Peer-To-Peer networks for sharing 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.

Enterprise tries minimizing the completion time of jobs. Spawn aims at the fairness of resource allocation: the number of CPU slots bought is proportional to the amount of funding. Nimrod/G is a resource management and scheduling system based on the parameter sweeping system. Nimrod and Nimrod/G are built with Globus toolkit. Resources can be associated with prices and jobs can be related to given budgets. The authors do not focus on economic feature and give no further explanation and implementation of their economic idea over Nimrod/G. Libra is an expansion of Nimrod/G for cluster computing. Its objective is to maximize the successful-execution rate under the constraint of budget. Performance evaluation shows its improvement in the rate of accepted jobs compared with FIFO. Unlike most related work that considers performance objective only for resource consumers, First Reward, a value-based heuristic task scheduling scheme for a market based grid setting, tries maximizing the profits of providers.

Partial results of the incentive-based scheduling work are reported in Sebastian Stein et al (2011) consumers assign budgets to jobs and choose providers according to the claimed completion time. No price or IF mechanisms are investigated. In Xiao et al (2010), the impact of IF 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 work does.

6.4 IJRM SCHEDULING

The incentive-based scheduling scheme: An incentive-based scheduling scheme IJRM is proposed here 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

(3) Three market instruments, job announcement, price and IF are used.

6.4.1 Peer-to-Peer Scheduling Framework

The proposed scheduling framework takes advantage of the P2P technology, utilizing its characteristics of decentralization and scalability. A central server is far from robust and the maintenance is costly. Apart from that, as every participant in the computational grid is autonomous and acts individually as shown in Figure 6.2. A decentralized scheduling infrastructure is more favorable. Furthermore, owing to the dynamic changes of grid environments, players may enter or leave at any time. A P2P network can handle such dynamics.

The computational grid G has several portals, through one of which a provider can join the grid. On entering, the provider gets the information of designated neighbors from the portal and then connects into the P2P network.

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. Realization of

the complete competition among all the providers based on two considerations is desired. Firstly, 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, without considering the geographical locations. Secondly, the number of providers will not be too large, (typically not more than several hundred), for a provider represents an administrative domain, within which local scheduling policies are employed. It is well known that blind-flooding-based broadcasting is a fatal weakness of unstructured P2P networks. Many investigators Liu et al (2004) have studied building overlay networks, whose topology closely matches the topology of physical networks. Once an overlay network with the desirable characteristic is built, an efficient broadcasting mechanism with good performance can be constructed.

The P2P scheduling infrastructure enables the effective interactions between consumers and providers and jobs are scheduled as a result. Scheduling scheme of steps is that a single job goes through in the scheduling scheme. All jobs from consumers follow the same steps:

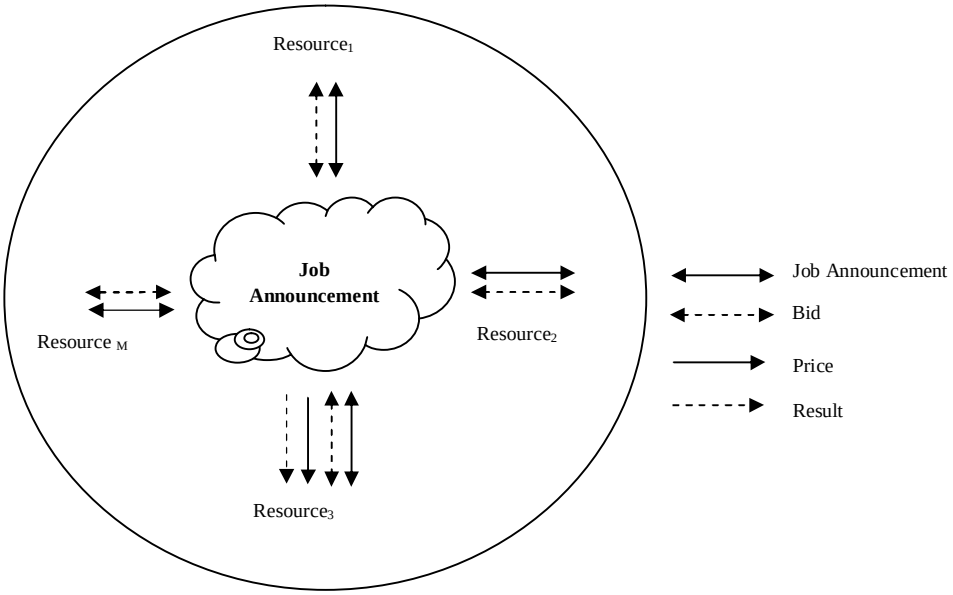

Figure 6.2 Steps for Bidding a Job

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 to the selected provider.

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.

ALGORITHMIC REPRESENTATION OF BROADCAST AND ANNOUNCEMENT

```
Broadcast_job(Job[])       //method used to broadcast the new incoming job to all
                             providers which are on  the computational grid.
{

            Providers_list [];   // list of providers who connected on the grid.
            for(providers=0;providers<=Providers_list[];providers++)
            {
                 Providers_list[providers].getAnnouncement(Job[]);
                   //Exposing the job details to Providers.
}
}
getAnnocement(job[])
{
   //Provider will receive the job announcement for the new incoming job.
    If (provider.availableTime > job.walltime)
   {
         Send_bid(price,job[])
   }
   else
    {
       //ignores the job.
            Remove(job[])
    }
 }
```

Figure 6.3 Representation of Broadcast and Announcement

The value of the parameter-waiting interval in step 3 should try not to miss any potential bid and also to make decisions as soon as possible. In the experiments conducted in this work, the average execution time is chosen as the waiting interval for synthetic workloads and 10 sec for real workloads (Figure 6.3).

Both are rather conservative values so that the performance evaluation results will not be favorably skewed.

6.5 IJRM SCHEDULING ALGORITHMS

The incentive based scheduling algorithm is designed with the criteria such as job levels, local schedule information and dynamic price assignment. Four algorithms have been designed in this work for providers.

- The job competing algorithm describes how a provider bids when receiving a job announcement in step two.
- The heuristic optimization 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 Economic algorithm and the IF-Regulating algorithm help a provider in dynamically adjusting its unit price and IF properly over the period of its participation in the computational grid.

6.5.1 Job Competing Algorithm

As a result of the decentralized scheduling framework, providers make decisions based on local, imperfect and delayed information, which often puts them in a dilemma.

Things get more complex when more jobs are involved. There are two extreme attitudes for providers to compete for jobs. They are aggressive and conservative methods. In aggressive method, the provider never considers the unconfirmed jobs when estimating whether it is able to meet job deadline or not, which is risky but changes always accompany risk. In conservative method, the provider always keep unconfirmed jobs in the job queue to consider for a certain time, this attitude will never lead to deadline missing but may lose potential changes and hence the profits. The different competition attitudes will result in different allocations of profits. To study the impact of competition attitude, a parameter namely Independent Factor (IF) is defined as a real number from 0-1. A provider will insert unconfirmed jobs into its job queue at the probability of 1-IF.

Every time a provider receives a job announcement, it starts the job competing algorithm. The algorithm is stated as follows: Its time complexity is O(q), where q is the number of jobs in the job queue:

Step 1: The provider estimates whether it is able to meet the job deadline.

Step 2: The provider offers a price for the job.

Pseudo code

```
1 price ← p * Ls;
2 if reordered then
3 price ← λ *price ; endif
```

ALGORITHMIC REPRESENTATION OF PRICING MECHANISM

```
getAnnocement(job[])
{
   //Provider will receive the job announcement for the new incoming job.
     IF =1 ; //Competittion attitude value is a real number from 0-1.
    If (provider.availableTime > job.walltime)
   {
          price = price * joblength;
          if (reordered)// The loop will execute only if the reordered is set to 1.
          {
                price=price * λ;
           queue[IF]=job[];   //reserves the 1st place for the incoming job and increase the
price value
sleep(10);    //waits 10 seconds for the incoming job.
}
recheck = getAnnocement(job[]);
if (recheck == null)
{
    reordered=false;
   queue[IF]=null;  // Removes the job from queue.
}
else
   execute(job[]);
     }
   }
```

Figure 6.4 Representation of Pricing Mechanism

Here, p is the unit price of the provider, Ls is the job length of job s and λ is a decimal slightly larger than 1. When the variable reordered is set to true, the price is raised. Generally, jobs are enquired in the order of their arrival. 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. Thus, it is reasonable to charge more for them. On the other side, a tight deadline also increases the possibility of failing to meet it. The providers raise the price to reduce the chance of being chosen to some extent (Figure 6.4).

Step 3: The provider sends the price as a bid and inserts the job at the place that the variable insert place indicates at the probability of 1-IF. 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 that the offered jobs are not to be deleted.

6.5.2 Heuristic Optimization Algorithm

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 its job queue. This is called local scheduling. On calculating the penalty of all the possible permutations of jobs to find out the one with the least penalty is NP-complete, a heuristic approach is applied (Figure 6.5).

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 optimization algorithm. The algorithm is needless for providers whose IF is equal to 0, because they always keep unconfirmed jobs. The heuristic optimization scheduling algorithm is described with the following pseudo code. Its time complexity is O(q2):

6.5.3 Economic Scheduling Algorithm

As the 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 the algorithm of this

work, it is assumed 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 same way that a job announcement is forwarded.

Pseudo Code

```
Insert      ←   Pq;
Penality    ←  Calculating the Penality of  inserting the job at Pq
For i       ←   q – 1 to 0 do
Penalityi   ←     Calculating the penality of inserting the job at Pi;
If penalityi < penality then
Penality    ←   penalityi
Insert      ←  Pi
```

ALGORITHMIC REPRESENTATION OF HEURISTIC ALGORITHM

```
            insertJob[]
{
// Job is inserted into an queue called as the Penalty Queue.
            penaltyQueue[job[]];
            penalty=penaltyQueue[job[penalty_value]]; //Job's penalty value
                                          when its entered into penalty queue.
            For(i=0;i<penaltyQueue.length;i++)
            {
                Penalty_job=job[penalty_value]; //job's penalty value when
                                                    its inserted into the queue
                Penalty=penaltyQueue[job[i.penalty_value]];
                If(Penalty < Penalty_job)
                penaltyQueue[i]=job[]; //inserts the job into the penalty queue.
            }
}
```

Figure 6.5 Representation of Heuristic Algorithm

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 and the price as market price and it acts as a directive as shown in Figure 6.6. 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. The algorithm is stated as the following pseudo code and the time complexity of this algorithm is O(1)

Pseudo Code

$r_1 \leftarrow L_O/L_T$;
$r_2 \leftarrow C/\Sigma\ 0<j<m\ Cj$
if offered a job then
if $r_1 > r_2$ and $p <= P_M$ then
$p \leftarrow \alpha * p$;
endif
else // delete an unconfirmed job
if $r_1 < r_2$ and $p >= P_M$ then
$p \leftarrow \beta * p$;
endif
endif,

Figure 6.6 Representation of Job Offering Mechanism

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. The offered job length and the total job length rewind when the total capability is updated. The r_1 and r_2 are the resources for servicing the jobs.

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. The price-adjusting mechanism in this work is simple and intuitive that is just to make prices different and it differentiates the chances of providers to be chosen and eventually realize the fair allocation of profits. Furthermore, the algorithm skillfully avoids endless increase or

decrease in unit price. Thus, the price will fluctuate around the market price, which is acceptable for both consumers and providers.

Providers can choose not to adjust price every time one job is offered or not but start the algorithm every several jobs. However, if so, the providers are slow to react to the market. The fairness will be degraded accordingly.

6.5.4 Independent Factor Regulating Algorithm

Like human beings, providers have diverse behavior. Thus, providers with various IFs coexist in the computational grid. 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 some 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.

Pseudo Code

```
//Every time the penalty increases
if Rp >= THp and IF >= ε then
IF←IF - ε;
endif
//Every time a certain interval such as 1 day
if Rp<THp and RJ >= THJ and IF <= 1 -ε then
IF← IF+ε;
Endif
```

ALGORITHMIC REPRESENTATION

```
Calculate_penalty
{
        ratio_penalty=0;    //ratio of penalty to profit
        threshold_penalty=0;        // Threshold limit of penalty of profit.
        ratio_job=0;                //ratio of penalty to job
        threshold_job=0;   //Threshold limit of penalty of job.
if(ratio_penalty >=threshold_penalty) && (IF>=ε)
        IF=IF-ε;
        //After one day interval, and if the jobs didn't move to running queue.
        If(ratio_penalty < threshold_penalty)  && (ratio_job>=threshold_job)
                                                        && (IF<=1-€)
        IF=IF+ε;
 }
```

Figure 6.7 Representation of Penalty Factor

A wise provider, whether a conservative or an aggressive one, should never hold its attitude toward when competition if things like that happen. It will adjust its IF according to the situation that it perceives. Thus it is the main objective of the IF-regulating algorithm. The following pseudo code describes the algorithm and the time complexity of this algorithm is O (1):

Here, Rp 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, IF is adjusted accordingly at

the step of ε. As can be seen, the check of Rp 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, Rp is checked every time and the penalty increased, whereas RJ can be checked regularly at a little longer interval such as 1 day (Figure 6.7).

6.6 IJRM IMPLEMENTATION

The cost and economic estimation scheme for computational grids is implemented using the Java environment. The system is designed for three applications namely grid server, resource provider and consumer. The grid server application is designed to handle the authentication and scheduling operations. The resource provider application is designed to provide shared resources among other nodes. The consumer application is used to access the resources. The applications are interconnected using the Remote Method Innovation technique. The resource provider allocates the resources to the consumer with reference to the scheduling scheme provided by the grid server.

The cost and economic estimation scheme is designed with the priority information. The supply and demand factor is used for the cost estimation process. The cost gets increased due to the demand factor and the incentive gets increased with reference to the supply factor. The priority factor is decided by the provider and the consumer during the resource request process. The network delay factors are considered in the proposed scheme.

Grid server: The grid server application is designed to carry out the administrative operations. The user management and authentication tasks are handled by the grid server. This system integrates the scheduling process in

the grid server application with the support of the autonomous information from the applications of providers and consumers. The resource allocation is carried out in the grid server application.

Consumers: In this work, only consideration of computation-incentive jobs is made, where all communication/networking 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 announcement to the computational grid. The Job announcement includes the information of job length and job deadline. The Job length is an empirical value assessed as the execution time of the job on the designated standard platform. The 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. Thus, a job with length = 10 and Deadline = 100 means that the job's execution takes 10 time units on a designated standard computer.

Providers: From the scheduling viewpoint, each resource provider is modeled with three parameters such as 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 jobs length. Unit price refers to the price that the resource offers for executing to its unit price and job length.

6.7 SIMULATION RESULTS

Globus is an open source toolkit which allows the people share their databases, applications, computational power and other tools over the Internet in terms of secure and seamless manner. The Globus can be thought as a middleware that includes software services and libraries for resource monitoring, discovery allocation and scheduling management. It is portable so it can be used for any platform. Resource management and Job management are supported by the Globus. It also provides web mechanisms to create distributed system framework. The proposed scheduling framework can easily be implemented with this toolkit for the following experiments.

Experiment1. The impact of IF on scheduling performance is studied. In this experiment, all the providers in the computational grid are configured with the same and fixed IF. Three typical IF configurations are investigated such as the extremely conservative and extremely aggressive with 0 and 1 respectively and the moderate one is 0.5.

The results are analyzed as follow:

First, the incentive for consumers is studied. Our performance metric for consumer incentives is the successful-execution rate of jobs. There are two related metrics:

- Failure Rate
- Deadline Missing Rate.

A job fails when all the providers think that they cannot meet the deadline and decide not to bid. Thus, the failure rate is defined as the ratio of the number of jobs that fail to n, which is the number of jobs submitted to the computational grid. The deadline missing rate 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 6.1 Failure Rate of Jobs

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

In Table 6.1, it is seen that if providers are extremely aggressive, that is, IF is equal to 1, all the jobs submitted by the consumers can be executed. If providers are not so aggressive, job failure happens, for providers reserve for unconfirmed jobs and tend to fail to meet the deadlines of those coming later. When the conservation comes to an extreme, that is, IF is equal to 0, the failure rate increases greatly when the system load gets heavier. When the system load is 0.7, the failure rate is not above 15 percent as shown in Figure 6.8. The successful execution rate is calculated by failure rate and deadline.

Table 6.2 Deadline Missing Rate of Jobs

System Load	IF=0	IF=0.5	IF=1
0.1	0.00%	0.00%	0.00%
0.2	0.02%	0.00%	0.00%
0.3	0.04%	0.01%	0.00%
0.4	0.22%	0.03%	0.00%
0.5	0.78%	0.038%	0.00%
0.6	0.04%	0.079%	0.00%
0.7	0.36%	0.1290%	0.00%

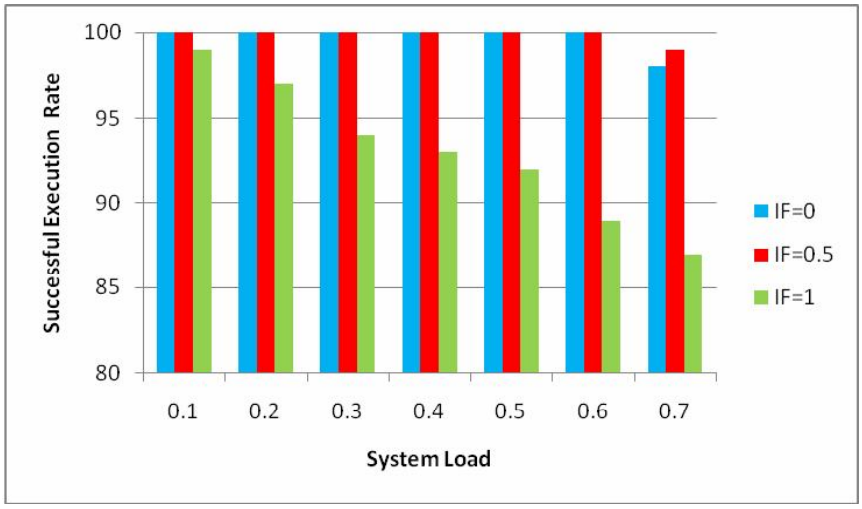

Figure 6.8 Successful Execution Rates of Jobs

Table 6.2 is about the deadline missing rate. When IF 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 gets increased. However, even when the system load is 0.7 and IF is equal to 1, the deadline missing rate is below one percent.

For providers, first it is recalled how IF impacts the total penalty which is the sum of the penalty paid by all the providers. In table 6.3 and figure 6.9 shows the results.

Table 6.3 Total Penalty of Providers

System Load	IF=0	IF=0.5	IF=1
0.1	0.00	0.00	0.00
0.2	0.18	0.00	0.00
0.3	2.83	0.97	0.00
0.4	19.09	9.75	0.00
0.5	86.40	50.00	0.00
0.6	625.00	375.08	0.00
0.7	6425.89	2970.00	0.00

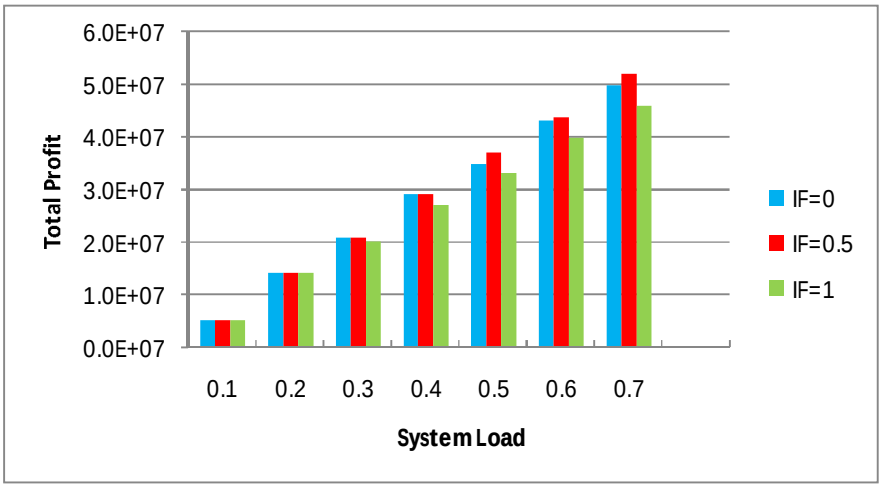

Figure 6.9 System Load with Profit

Apparently, as IF and the system load increase, the total penalty gets increased, which is consistent with the observation about the deadline missing rate. The more meaningful metric is the net profit. The Conservative providers never need to pay the penalty, but they lose potential jobs because

of their conservative attitude. Thus, the total profit is not very satisfactory. The results show that a trade-off attitude, that is, IF is equal to 0.5, is superior to the extreme two

It is obvious that the two algorithms complement each other and are both indispensable. When disabling the IF-Regulating 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 missing rate results in more penalty, finally lower successful-execution rate, and, worse, fairness.

Experiment2. The performance of the incentive based scheduling scheme is compared with other schemes. Several representative scheduling schemes that are deployed in decentralized scheduling frameworks are chosen. Three of them schedule jobs based on the heuristics that sort jobs by their arrival time, length, deadline and value which are the essential characteristics of jobs, respectively. They are called First Come, First Served (FCFS), Shortest Job First (SJF), Earliest Deadline First (EDF) and First Reward (FR) respectively. These local scheduling algorithms are straightforward, easy to implement, and often taken as default policies in scheduling systems.

For fair comparison, adding a job admission control algorithm for them: a provider accepts a new job and bids for it only when it will not lead to new deadline missing if integrated to the job queue. Unlike them, First Reward, which is a value-based scheduling scheme, instead of checking the deadline, accepts a job only when it brings value above a predefined slack threshold and sort’s jobs by value too. After testing various slack thresholds for our simulation workload, setting the value of parameters for First Reward

as 1, the discounted rate as 1 percent, and the slack threshold as 0. In these schemes, consumers randomly choose one provider for each job from those that are willing to accept it. These scheduling schemes are compared with ours on the two performance objectives that reflect the incentive for providers and consumers. Figures 6.10 illustrate the results.

Figure 6.10 Comparison of IJRM with Other Scheduling Algorithms

The scheme outperforms all other schemes in both performance objectives. The intuitive explanation is that the IF-Regulating algorithm keeps the job queue of each provider a proper length, the job-competing algorithm is basically FCFS while allowing tight-deadline job first, the local scheduling algorithm is basically FCFS while minimizing the penalty (to some extent reducing the deadline missing rate), which contributes to a higher successful execution rate, and the price-adjusting algorithm directs the allocation of jobs, with the price being the indication, which contributes to fairness among providers.

Experiment 3 The performance of the incentive-based scheduling scheme is compared with the other schemes under real workloads. The

comparison under the workload trace are collected and the R1 trace is chosen, because it is relatively new, is from a grid context, and records the CPU time used by each job. The workload contains 195,497 jobs after removing the meaningless ones,

Here simulations have been conducted under amplified workloads by dividing the recorded job arrival interval by factors of 2 and 3. The proposed scheme shows a comparable performance in the successful-execution rate and an excellent performance in realizing the fairness among all the providers. Utilization is calculated as the fraction of time that a resource is busy executing jobs. This scheme achieves a satisfied fairness among providers as shown in Figure 6.11. As a result of the fair allocation of profits and the price-adjusting algorithm, the desirable property of balanced utilization is achieved.

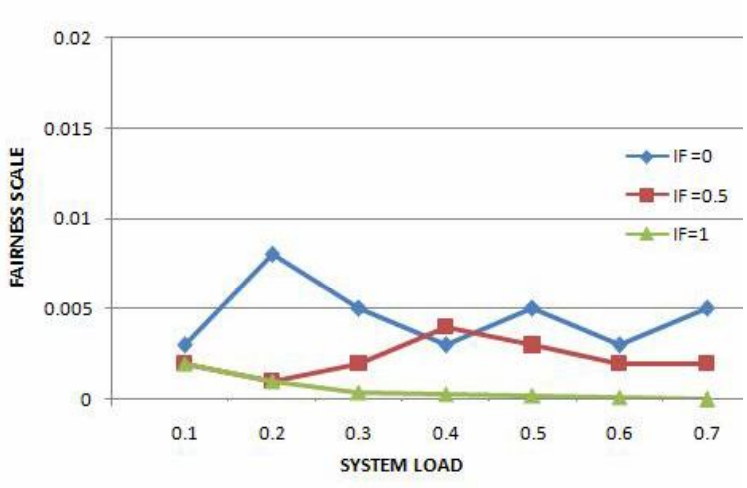

Figure 6.11 Fairness Deviation

6.8 CONCLUSION

This chapter formulates job scheduling in a sustainable market-like computational grid as a double-objective optimization problem to optimize economic factor for both consumers and providers. As the problem is at least NP-complete, this chapter proposes an incentive-based scheduling scheme with heuristics, using a P2P decentralized scheduling framework.

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, such as job announcement, price, and IF, are employed, and the former two are circulated in the grid.

Simulations are conducted under synthetic and real workloads to validate this approach. The results show that this scheme has several advantages. Although each participant makes local/autonomous decisions, desirable properties emerge in the grid system as a whole, including a high successful-job-execution rate, a fair allocation of profits, and a balanced utilization of resources. The proposed scheme achieves the dual objectives better than other methods.

For the real workloads under three different load conditions, the proposed scheme achieves successful-job-execution rates of 94 percent to 96 percent, with the fairness deviation as low as 0.0015 to 0.005. Both the metrics show better results than those of FCFS, SJF, and FR. The earliest

deadline first method has slightly better successful-execution rates (95 percent to 97 percent). However, EDF is an unfair method, with a much larger fairness deviation (0.18 to 0.23). Such an unfair scheduling of EDF leads to a very imbalanced resource utilization (12 percent for the least utilization but 39 percent for most utilization), in sharp contrast to the proposed method, where the gap is under 1 percent. The local mechanisms of adjusting the price and adjusting IF are both effective in improving fairness and the successful-execution rate.