Priority Based Job Scheduling using Nash Equilibrium Strategy for Grid Computing

Dharamendra Chouhan, S.M. Dilip Kumar, Jerry Antony Ajay
Department of Computer Science and Engineering
University Visvesvaraya College of Engineering
Bangalore , India
{dharmu2007, dilipkumarsm}@gmail.com, antonyajay21@hotmail.com

ABSTRACT: *Effective and efficient job scheduling is an important aspect of Grid computing. Task scheduling becomes more complicated in a Grid environment, due to geographically distribution, heterogeneity and dynamic nature of grid resources. In this paper, a new computational job scheduling policy based on Nash Equilibrium is proposed. The jobs are put into scheduling queue based on priority. This priority is computed by the amount the grid user is willing to pay. Our solution is based on the Nash Bargaining Solution which provides a Pareto optimal solution for the distributed system and is also a fair solution to the problem under consideration. One of the goals of our work is to provide fairness to the customers i.e., all the users and their jobs should experience approximately equal expected response time which includes the expected queuing delay, processing time, and communication delay. As another part of our goal, we try to maximize the revenue levels for the grid owners. This scheduling policy is simulated using Alea GridSim toolkit to test the performance.*

Keywords: Grid Computing, Nash Equilibrium, GridSim, Scheduling

Received: 31 July 2012, Revised 7 October 2012, Accepted 12 October 2012

1. Introduction

A distributed system often consists of heterogeneous computing and communication resources. Due to the possible differences in the computing capacities and uneven job arrival patterns, the workload on different computers in the system can vary greatly is presented in [1].This situation can lead to poor system performance. Improving the performance of such a system by an appropriate distribution of the workload among the computers is commonly known as job allocation or load balancing. Formally, this problem can be stated as follows: given a large number of jobs, find an allocation of jobs to the computers optimizing a given objective function (e.g., total expected (mean) response time (expected queuing delay + processing time + any communication time) of the system or total expected cost (the price that has to be paid by the users for using the resources) of the system). There are two main categories of load balancing policies: static policies and dynamic policies as stated in [2]. Static policies base their decisions on collected statistical information about the system. They do not take into consideration the current state of the system. Dynamic policies base their decisions on the current state of the system, where state could refer to, for example, the number of jobs waiting in the queue to be processed and job arrival rate.

The nodes (computers) exchange this information periodically and will try to balance the load by transferring some jobs from heavily loaded nodes to lightly loaded nodes. Despite the higher runtime complexity, dynamic policies can lead to better performance than static policies. A Grid [3] is a conglomeration of computing resources connected by a network, which is used to solve large-scale computation problems.

This system tries to run these applications by allocating the idle computing resources over a network or the internet commonly known as the computational grid. These computational resources have different owners who can be enabled by an automated negotiation mechanism by the grid controllers. The prices that the grid users have to pay for using the computing resources owned by different resource owners can be obtained using a pricing model based on a game theory framework. The objective of job allocation in grid systems can be to find an allocation that reduces the price that the grid users have to pay for utilizing the resources.

In grid computing systems, there are often large amounts of resources available to be used for computing jobs. Scheduling in a grid computing system is not as simple as scheduling on a multi-processor machine because of several factors. These factors include the fact that grid resources are sometimes used by paying customers who have interest in how their jobs are being scheduled [4]. However, grid computing systems usually operate in remote locations, so, scheduling tasks for the clusters may be occurring over a network [5]. Job scheduling algorithms are commonly applied to grid resources to optimally post jobs to grid resources [6][7]. Usually, grid users submit their jobs to the grid manager to utilize and fulfill the facilities provided by grid. The grid manager distributes the submitted jobs among the grid resources to minimize the total response time. In a Grid environment, there is moderately large number of job scheduling algorithms proposed to minimize the total completion time of the jobs [8][9]. The rest of the paper is organized as follows. Section 2 presents the related works. In Section 3 the system model for scheduling in Grid computing environment is presented. In section 4, solution is proposed and an algorithm is written for the same. The simulation of the Nash priority scheduling algorithm using Alea GridSim is presented in section 5. Finally, section 6 concludes the paper.

2. Related Work

There has been significant research continuing to attempt to devise scheduling algorithms for grid environment's problem of efficient job assignment. Some of the jobs scheduling algorithms in a grid environment are given below.

Extensive studies exist on the static load balancing problem in single-class and multi-class job distributed systems. Most of those used the global approach, where the focus is on minimizing the expected response time of the entire system over all the jobs.

Different network configurations are considered and the problem is formulated as a non-linear optimization problem in [10][11][12] and as a polymatroid optimization problem in [13]. These schemes implement the entire system optimization approach in order to determine a load allocation that yields a system-wide optimal expected response time. A few studies exist on static load balancing that provides individual-optimal and user-optimal solutions [14] which are based on game theory. Individual and user-optimal policies for an infinite number of jobs/users based on non-cooperative games using Wardrop equilibrium are studied in [15]. An individual-optimal solution for finite jobs based on cooperative game theory is provided in [16] optimal solutions based on Nash equilibrium are provided in [17] for finite number of users. Game theory is also used to model grid systems [18] and for price-based job allocation in distributed systems [19]. X. He et al. [20] have proposed an algorithm based on the conventional min-min algorithm known as QoS guided min-min which schedules the jobs requiring high bandwidth before others. F. Dong et al. [23] have proposed an algorithm called QoS priority grouping scheduling. This algorithm, considers completion time and acceptation rate of the jobs and the makespan of the entire system as key factors for job scheduling. E. Ullah Munir et al. [24] have proposed a new job scheduling algorithm which makes use of grid computing environments known as QoS Sufferage. K. Etminani et al. [9] have proposed an algorithm which provides a solution on basis of max-min and min-min algorithms. The algorithm discovers the situations where to adopt one of these two algorithms, based on the standard deviation of the estimated completion times of the jobs on every computing resources. L. Mohammad Khanli et al. [21][22] have proposed a QoS based scheduling algorithm for an architecture called Grid-JQA. In this method the solution involves applying an aggregation formula which includes a combination of different parameters together with weighting factors to perform operations on QoS.

3. System Model

In this section, we present the system model of scheduling in Grid computing environment. A queuing network model of Grid resources based on Nash Equilibrium is considered as shown in Figure 1. Set of jobs arrived are kept in the jobs queue. These are sorted out according to pricing bands as shown in table 1. $P1$, $P2$, $P3$,…., Pn denotes the priority of the jobs based on the pricing bands. The payoff matrix computes the Nash Equilibrium according to the algorithm given in this paper. In the figure 1 NE denotes the Nash Equilibrium of each payoff matrix.

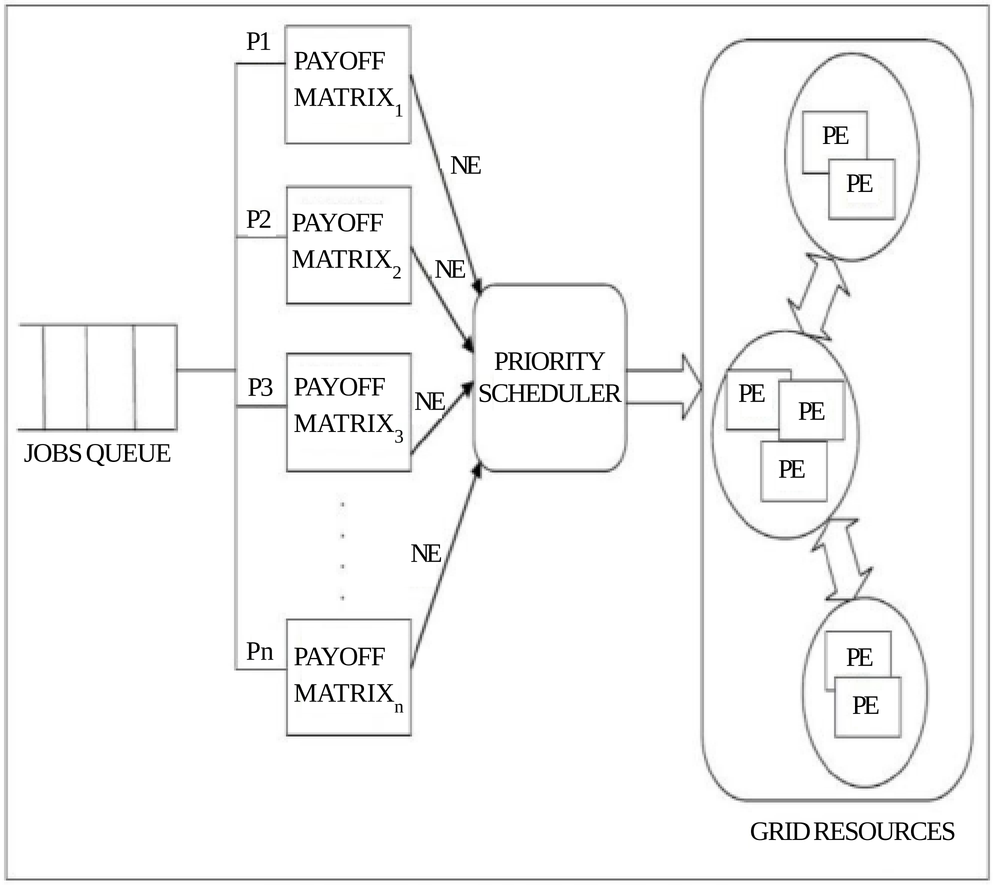

Figure 1. Queuing Model of a Grid System using Nash Equilibrium

JOB ID	OFFERED PRICE($)	REQUESTED RESOURCES
1	2000	50
2	1000	20
3	4000	30
4	200	60
5	5600	100
6	100	21
7	340	34
8	450	10
9	234	15
10	125	06

Table 1. Job Request

These values are further sent to the priority scheduler which assigns grid resources for their execution. The priority scheduler assigns jobs to the grid resources based on their pricing bands. The best offer is selected by computing the Nash Equilibrium from the payoff matrix as shown in the figure.1.

RESOURCE NAME	AVAILABLE CAPACITY
R1	120
R2	150
R3	134
R4	200
R5	140
R6	160

Table 2. Resource Availability

JOBS/ RES	R1	R2	R3	R4	R5	R6
1	(2000,70)	(2000, 100)	(2000,84)	(2000,150)	(2000,90)	(2000,110)
2	(1000,100)	(1000,130)	(1000,114	(1000,180)	(1000,120)	(1000,140)
3	(4000,90)	(4000,120)	(4000,104)	(4000,170)	(4000,110)	(4000,130)
4	(200,60)	(200,90)	(200,74)	(200,140)	(200,80)	(200,100)
5	(5600,20)	(5600,50)	(5600,34)	(5600,100)	(5600,40)	(5600,60)
6	(100,99)	(100,129)	(100,113)	(100,179)	(100,119)	(100,139)
7	(340,86)	(340,116)	(340,100)	(340,166)	(340,106)	(340,126)
8	(450,110)	(450,140)	(450,124)	(450,190)	(450,130)	(450,150)
9	(234,105)	(234,135)	(234,119)	(234,185)	(234,125)	(234,145)
10	(125,114)	(125, 144)	(125,128)	(125,194)	(125,134)	(125,154)

Table 3. Payoff Matrix

3.1 Application Model

3.1.1 Game Theory

Game theory aims to help us understand situations in which decision-makers interact. Like other sciences, game theory consists of a collection of models. A model is an abstraction we use to understand our observations and experiences. We can represent the preferences of a player in a game by considering an entity known as a payoff function, which associates a number with each action of the player in such a way that actions with higher numbers are preferred [26]. More precisely, the payoff function u represents a decision-maker's preferences if, for any actions a in A and b in A, $u(a) > u(b)$ if and only if the decision-maker prefers a to b.

3.1.2 Mixed Strategy

In the generalization of the notion of Nash equilibrium that models a stochastic steady state of a strategic game with VonNeumann-Morgenstern (vNM) preferences, we allow each player to choose a probability distribution over her/his set of actions rather than restricting her/his to choose a single deterministic action. We refer to such a probability distribution as a mixed strategy. A mixed strategy of a player in a strategic game is a probability distribution over the player's actions.

3.1.3 Pure Strategy

A mixed strategy may assign probability 1 to a single action, by allowing a player to choose probability distributions; we do not prohibit her from choosing deterministic actions. We refer to such a mixed strategy as a pure strategy. Player *i's* choosing the pure strategy that assigns probability 1 to the action a_i is equivalent to her simply choosing the action and we denote this strategy simply by a_i.

3.1.4 Nash Equilibrium

(Nash equilibrium of extensive game with perfect information) The strategy profile s^* in an extensive game with perfect information is a Nash equilibrium if, for every player I and every strategy r_i of player I, the terminal history $O(s^*)$ generated by s^* is at least as good according to player i's preferences as the terminal history $O(r_i, s^*\text{-}i)$ generated by the strategy profile (r_i, $s^*\text{-}i$) in which the player i chooses r_i while every other player j chooses $s^* j$. Equivalently, for each player i or every strategy of player i,

$$u_i(O(s^* - \mathrm{i})) \geq u_i(O(u_i, s^* - \mathrm{i}))$$

where u_i is a payoff function that represents player I's preferences and O is the outcome function of the game.

3.1.5 Strategy Profile

A strategy profile (sometimes called a strategy combination) is a set of strategies for each player which fully specifies all actions in a game. A strategy profile must include one and only one strategy for every player.

According to our model, we populate the payoff matrix by considering values from the '*Job request*' table and '*Resource availability*' table. In this model, we consider the two players to be the '*job request*' i.e. obtained from Table 1 and '*Resource availability*' i.e. obtained from Table 2. Table 3 shows the final payoff matrix which is populated according to game theory principles. A cell in the matrix consists of two elements that is obtained by solving the following equations:

$$\mathrm{R}_\mathrm{i} \text{ (first element) } = \text{Price that the customer is willing to pay} \tag{1}$$

where R_i symbolizes the i^{th} row

$$\mathrm{C}_\mathrm{i} \text{ (second element) } = \mathrm{r} \tag{2}$$

$$\text{Where } r = \text{Available Resources} - \text{Requested Resources} \tag{3}$$

Inorder to find the Nash Equilibrium, we need to formulate an objective function that would promise maximum profits for the grid service providers and also maximize the resources that are remaining after the scheduler allocates jobs to the grid.

From basic inference, we observe that we need to maximize the first element in every cell since we are looking forward to develop a model that promises higher revenue to the grid service providers. We also need to maximize the second element in a cell since from equation (3), the remaining resources i.e. r, should be as large as possible inorder to cater to other jobs in the queue. Thus, we can formulate the objective function as follows :

$$n_{(1,1)} = P_{max}(R_1, C_1)$$

$$n_{(1,2)} = P_{max}(R_1, C_2)$$

$$n_{(1,3)} = P_{max}(R_1, C_3)$$

.

.

$$n_{(2,1)} = P_{max}(R_2, C_1)$$

$$n_{(2,2)} = P_{max}(R_2, C_2)$$

.

.

$$n_{(3,1)} = P_{max}(R_3, C_1)$$

$$n_{(3,2)} = P_{max}(R_3, C_2)$$

.

.

$$n_{(i,j)} = P_{max}(R_i, C_j) \tag{4}$$

Where $n_{(i, j)}$ denotes the ith row and the jth column.

Thus, equation (4) is our objective function. And inorder to find the Nash Equilibrium, we need to find the best allocation of a job to the resources that are available. The cell containing the element $n_{(i, j)}$ denotes the desired value.

Now, in-order to provide fairness to the users, we have considered many payoff matrices that compute the Nash Equilibrium for different price ranges. This scheme is depicted in figure 1. The advantage of this scheme is that the low priority jobs are not made to starve for a longer period of time.

4. Proposed Solution

In this section, we briefly explain the proposed solution for scheduling the jobs using Nash priority technique in grid environment. The user submits jobs along with the requirements to the Alea GridSim scheduling system. The submission of jobs to the resources involves computing the Nash for suitability of the available *PEs* and maximum price. If the requirement is satisfied, the jobs are assigned to the respective resources. This technique uses a dynamic priority mechanism to schedule the jobs to the system efficiently and maximize the resource utilization and revenue of the grid providers. The Nash priority scheduling model is depicted in the Figure 1. The jobs waiting for the service is placed in the waiting queue. The jobs selected from waiting queue are compared with maximum resource availability inorder to cater the service efficiently. And our objective here is not to sit idle resources for longer time, so identify those types and assign them the jobs. However, the jobs present in payoff table are executed based on Nash priority scheduling policy. The jobs complete their execution based on price bands as early as possible without rejection. So, smaller average response time is achievable and provides fairness to the grid users.

4.1 Algorithm1

```
// JOBS ARRIVING
1. add new jobs in queue
2. get job from the queue in a FCFS manner.
3. for each job exiting the queue
        Collect
               price willing to pay.
               requested resources.
              from job table
        end collect
   end for
4. if price < price_band1
        add job to payoff matrix 1
               set priority = 1
5. else if price < price_band2
          add job to payoff matrix 2
          set priority = 2
6. else if price < price_band2
          add job to payoff matrix 3
          set priority = 3
7. else
      print "unable to process job or mismatch of price bands"
8. end if-else
9. repeat steps 2 to 7 until all jobs occupy the pay of matrix
```

4.2 Algorithm 2

// PAYOFF MATRIX POPULATION

```
1. for each job present in payoff matrix
       calculate rem_res
       rem_res = resource_avail – resource_req
       add rem_res into column of payoff matrix
   end for

// COMPUTING NASH EUILIBRIUM
2. Consider price_ job1 = Maxrow
       for (price_ job (2...n) >Maxrow)
           Maxrow = price_ job$_i$
       end for
       selected_J = Maxrow

3. Consider rem_res of ‘selected_J’ on R1 =Max$_{col}$
       for ( rem_res(2...m) > Max$_{col}$ )
           Max$_{col}$ = rem_res$_i$
       end for
       selected_R = Max$_{col}$
4. OUTPUT selected_R and selected_J to scheduler
5. REPEAT steps 1 to 4 for the other two pricing bands.
// DONE COMPUTING NASH EQUILIBRIUM
```

4.3 Algorithm 3

```
// WORKING OF PRIORITY SCHEDULER
while ( job_waiting)
    if ( arriving = 3 and existing = 2 or 1 )
            pre-empt priorities 2 and 1
            assign resources to job_priority 3
    end if
    if ( arriving = 2 and existing = 1 )
            pre-empt priority 1
            assign resources to job_priority 2
    end if
    assign resources to Low Priority jobs
end while
// END WORKING OF PRIORITY SCHEDULER
// END OF ALGORITHM
```

The time complexity of the above mentioned algorithms is $O(n^2)$ Though this algorithm works with a quadratic time complexity, no significant delay was observed in the test cases considered below.

5. Simulation and Results

According to figure 2, we observe that at peak loads, i.e., the days between 13 and 18, the FCFS (First Come First Serve) algorithm performs less efficiently as compared to PBS_PRO (A Priority Scheduling Algorithm). Since, our algorithm uses game theory principles to implement priority scheduling, it produces significant results as compared to both FCFS and PBS_PRO. We prefer to call our algorithm as ‘*Nash-Priority Algorithm*’.

These algorithms were simulated on Alea 3.0 that runs on GridSim [25]. We assumed a configuration of 5000 gridlets, 103656

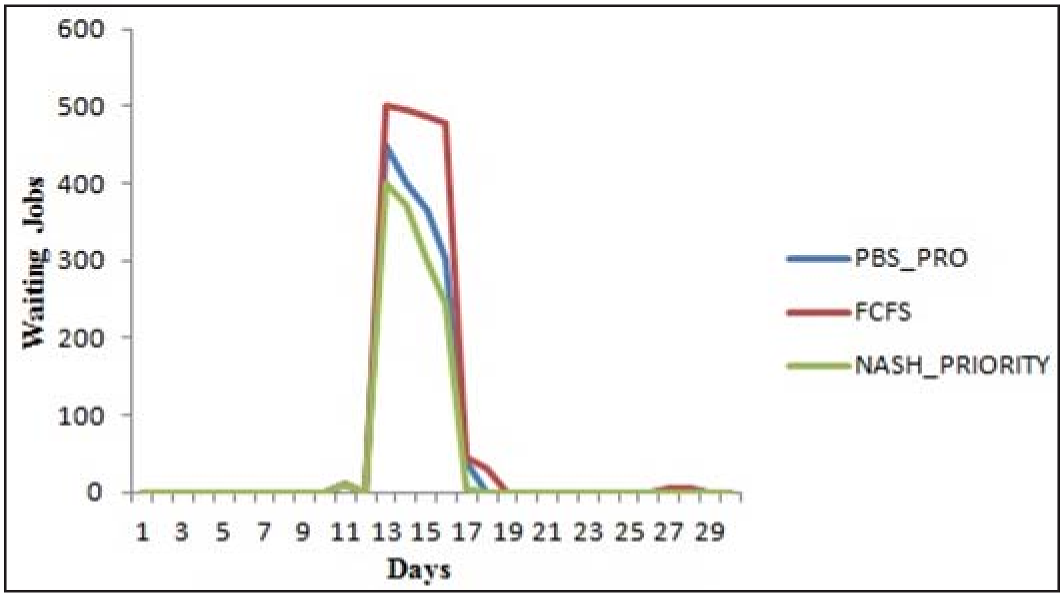

Figure 2. Average waiting Jobs/Day

processing elements and a fairness criterion of 1 for all the 3 algorithms under test.

On the 13th day, when the jobs entering the waiting queue was at it's peak, we observed that the FCFS algorithm chooses the 1st job and therefore makes all the remaining jobs to wait in the queue, thereby, not promising fairness to the users. Considering the priority based algorithm, i.e., PBS_PRO, we observe that on the 13th day, though the job load was high, the number of waiting jobs in the list was reduced to 450 , which is considered better than the FCFS scheduling algorithm. But, by using game theory principle, more specifically, Nash- Equilibrium technique, it is possible to further reduce the number of jobs waiting in the queue to a meager 401. This trend trickles down to the 14th, 15th, 16th and the 17th day, thereby, proving that priority scheduling using game theory principles is more efficient than other algorithms since the waiting jobs are significantly lower as compared to PBS_PRO and FCFS. Thus, Nash_Priority algorithm promises better fairness to the grid users. The data shown in figure 3 shows a statistical representation of the set of jobs that are selected by the '*Nash-Priority Algorithm*' on a typical day. As we can

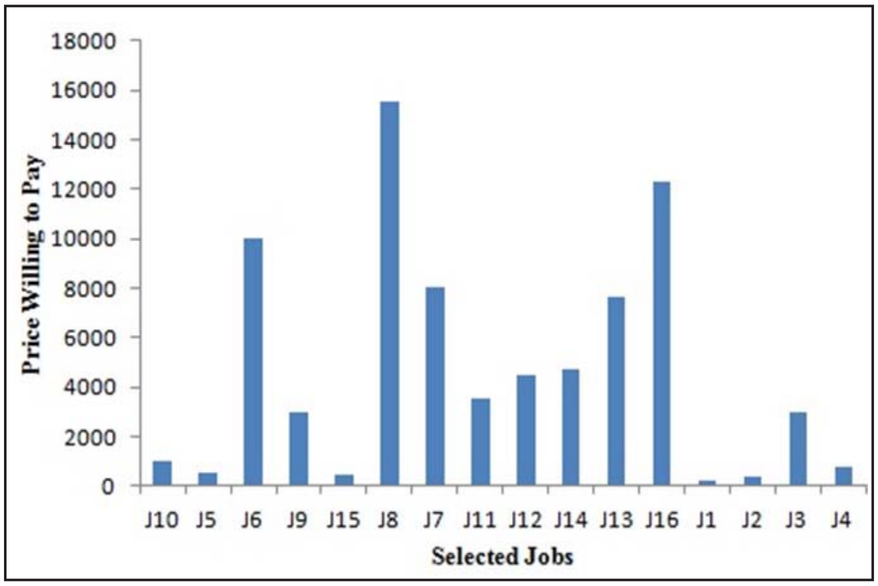

Figure 3. Jobs vs Price

observe, the fairness to every user is guaranteed since a job that pays less is not pre-empted completely and jobs that pay more are not the only ones that are considered for resource allocation. We can infer from our experiences that, firstly, the FCFS doesn't follow such a distribution. Secondly, the priority scheduler PBS_PRO, would give preferences only to jobs that pay more. Thus, we can conclude that 'Nash-Priority' also promises fairness to the users.

6. Conclusions and Future Work

The objective of job scheduling in grid computing is to minimize execution time of the application and maximize the revenue of grid providers by allocating jobs to proper processing elements. In parallel and distributed system, it is important to allocate tasks that can be executed efficiently and exhibit good performance.

As network environment has changed and the internet has grown up so vast, the focus these days are on parallel systems. Now a days, a system has heterogeneous platform and needs new scheduling strategies. This paper proposed a job scheduling algorithm keeping in mind the heterogeneous environment, i. e, each job has different computational price, computation cost, communication delay and each processor has different processing ability and network bandwidth. We used Nash equilibrium to schedule jobs according to their pricing and resource demand.

The game-theoretic model which can achieve higher revenue for the grid service providers, as well as yielding good makespan to the customer jobs. This grid computing applications can serve as a promising results to real-life applications. This service-oriented platform design satisfies the best quality of service requirements and maximizes the revenue in ever changing grid environment has the potential to design the future ubiquitous grid. The exploration on optimal strategies of non co-operative grid providers attests the research avenue of the selfish grids.

In this paper, we have considered only two player games (jobs and grid resources) for the time being. Multiplayer games will be more challenging due to the interaction between the users under many grid server. We plan to extend our work to model multiplayer games for grid resources within its range to give a better approximation of the interactions between the players. Finally, the job allocation strategy can also be improved by considering a Nash Equilibrium function that characterizes the grid scenario in a better fashion. The results showed that a significant improvement in terms of a smaller average response time is achievable and provides fairness to the grid users.

We are also trying to improvise the algorithm to a time complexity of O(log n).

References

[1] Anderson, T., Culler, D., Patterson. (1995). A case for NOW(Network of workstaions) team, IEEE micro 15, 54-64 1.

[2] Shivaratri, N. G., Krueger, P., Singhal, M. (1992). Load Distributing for locally Distributed Systems, IEEE Computer 25 (12) 33-34.

[3] Foster, I., Kesselman, C. (2004). The Grid 2: Blueprint for a New Computing Infrastructure, II Ed. Elsevier and Morgan Kaufmann Press.

[4] Hoschek, J., et al. (2000). Data Management in an International Data Grid Project. *In*: Proc. 1st International Workshop on Grid Computing (GRID Bangalore.

[5] Buyya, R., Steve Chapin, S., Di Nucci, D. (2000). Architectural Models for Resource Management in the Grid. *IEEE/ACM International Workshop on Grid Computing*.

[6] Mohammad Khanli, L., Analoui, M. (2008). Resource Scheduling in Desktop Grid by Grid-JQA. The IEEE 3rd *International Conference on Grid and Pervasive Computing*.

[7] Mohammad Khanli, L., Analoui, M. (2007). Grid_JQA: A QoS Guided Scheduling Algorithm for Grid Computing. The 6th *IEEE International Symp on Parallel and Distributed Computing*.

[8] Dong, F., et al. (2006). A Grid Task Scheduling Algorithm Based on QoS Priority Grouping. *In*: Proc. of the 5th *IEEE International Conf on Grid and Cooperative Computing*.

[9] Etminani, K., Naghibzadeh, M. (2007). A Min-min Max-min Selective Algorithm for Grid Task Scheduling. The 3rd *IEEE/IFIP International Conf on Internet*, Uzbekistan.

[10] Li, J., Wang, J., Kameda, H., Li, K. (1997). An effective scheduling algorithm for parallel transaction processing systems, *In*: Proc. *IEEE National Aerospace & Electronics Conference* (NAECON '97), 181-188, Dayton, Ohio, USA, July 14-18.[11] Kim, C., Kameda, H. (1992). An algorithm for optimal static load balancing in distributed computer systems, *IEEE Trans. on Computers*, 41 (3) 381-384.

[12] Li, Kameda, H. (1998). Load balancing problems for multiclass jobs in distributed/parallel computer systems, *IEEE Trans. on Computers*, 47 (3) 322-332.

[13] Zhang, Y., Kameda, H., Huang, S. L. (1997). A comparison of dynamic and static load-balancing strategies in heterogeneous distributed systems, *IEE Proceedings Computers and Digital Techniques*, 144 (2) 100-106, Mar.

[14] Kameda, H., Li, J., Kim, C., Zhang, Y. (1997). Optimal Load Balancing in Distributed Computer Systems. Springer-Verlag.

[15] Daniel Grosu, Anthony T. Chronopoulos. (2002). Ming-Ying Leung: Load Balancing in Distributed Systems: An Approach Using Cooperative Games. IPDPS.

[16] Daniel Grosu, Anthony T. Chronopoulos: Non cooperative load balancing in distributed systems. J. Parallel Distrib. Comput. 65 (9)1022-1034.

[17] Yu-Kwong Kwok, Kai Hwang, Shan Shan Song. (2007). Selfish Grids: Game-Theoretic Modeling and NAS/PSA Benchmark Evaluation, *IEEE Trans. on Parallel and Distributed Systems*, 18 (5). May.

[18] Ghosh, P., Roy, N., Das, S. K., Basu, K. (2005). A pricing strategy for job allocation in mobile grids using a non-cooperative bargaining theory framework, *Journal of Parallel and Distributed Computing* 65 (11)1366-1383.

[19] Ghosh, P., Basu, K., Das, S. K. (2007). A game theory-based pricing strategy to support single/multiclass job allocation schemes for bandwidth-constrained distributed computing systems Parallel and Distributed Systems, *IEEE Transactions on* 18 (3) 289-306.

[20] He, X., X-He Sun, Laszewski, G. V. (2003). QoS Guided Min-min Heuristic for Grid Task Scheduling, *Jr of Computer Science and Technology* 18, 442-451.

[21] Mohammad Khanli, L., Analoui, M. (2008). Resource Scheduling in Desktop Grid by Grid-JQA The 3rd *IEEE International Conf on Grid and Pervasive Computing*.

[22] Mohammad Khanli, L., Analoui, M. (2007). Grid_JQA: A QoS Guided Scheduling Algorithm for Grid Computing. The 6th *IEEE International Symp on Parallel and Distributed Computing*.

[23] Dong, F., Luo, J., et al. (2006). A Grid Task Scheduling Algorithm Based on QoS Priority Grouping. *In*: Proc of 5th *IEEE International Conf on Grid and Cooperative Computing*.

[24] Ullah Munir, E., Li, J., Sh Shi. (2007). QoS Sufferage Heuristic for Independent Task Scheduling in Grid. J Information Technology 6 (8) 1166-1170.

[25] www.fi.muni.cz/~xklusac/alea

[26] Martin J. Osborne. An introduction to Game theory.