Volume 2, Issue 8, August 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

A QoS Based Grid Job Allocation Scheme Using Game Theoretic Approach

G. Murugaboopathi [1], Leelambika.K.V. [2], V.Sujathabai [3], T.K.S. Rathish Babu [4]
[1]Head of Research and Development, VelTech MultiTech Dr.Rangarajan Dr.Sakumthala Engineering College, Avadi.
[2,4]Assistant Professor, Dept.of CSE, VelTech HighTech Dr.Rangarajan Dr.Sakumthala Engineering College, Avadi.
[3]Assistant Professor, Dept.of MCA VelTech HighTech Dr.Rangarajan Dr.Sakumthala Engineering College, Avadi.

***Abstract** - The grid is a platform that offers integrated architecture for sharing and aggregation of geographically distributed resources to solve potentially large computationally intensive problems. A computational grid requires heterogeneous computing resources, as well as the communication link that connect different nodes together. In grid, service providers need to cooperate and allocate jobs among users. The main aim of this project is to propose a Game Theoretic Solution to the QoS grid job allocation problem. The proposed algorithm is Cooperative Game approach which is fair to all users and represents a Pareto Optimal solution to the QoS objective. The advantage of this scheme is the relatively low overhead and robust performance against inaccuracies in performance prediction information.*

***Keywords:** Load balancing and task assignment, Job allocation, Scheduling and task partitioning, Pareto Optimal solution, Nash bargaining solution.*

I. Introduction

The grid is an infrastructure that enables the integrated collaborative use of high-end computers, networks, databases and scientific instruments owned and managed by multiple organizations. The grid applications often involve large amounts of data and computing and often require secure resources sharing across organizational boundaries.

The common issue of grid system is that, the job allocation and the load balancing scheme, providing (Quality of Service) QoS to the users. The job allocation mechanism aims to equally spread the load on each computing node, maximizing their utilization and minimizing the average job execution time. The job allocation algorithm is classified as static and dynamic. In static algorithm, the job allocation decisions are made at compile time and remain constant during runtime. Such algorithm is not applicable to grid system because the job allocation decisions are made in advance. In dynamic algorithm, the job allocation decisions are made at runtime and results in better performance.

The load balancing is a technique to spread work between two or more computers or resources, in order to get optimal resources utilization, throughput or response time. Load balancing scheme [4] is classified as centralized and decentralized. In centralized approach, one node in the system acts as a scheduler and makes the entire load balancing decisions. Information is sent from the other nodes to this node and is therefore the single point of failure. In the decentralized approach, multiple nodes in the system are involved in the load balancing decisions. It is therefore very costly for each node to obtain and maintain the dynamic state information of the whole system. Most decentralized approaches have each node obtaining and maintaining only partial information locally to make suboptimal decisions. Decentralized approaches, however offer better fault tolerance than centralized ones.

II. Related work

The existing algorithm is Proportional Scheme (PS) algorithm and Non-cooperative Game (NG) [2] algorithm. The proportional-scheme algorithm allocates jobs to providers in proportion to its computing power (that is, a job processing rate) of the provider. The proportional-scheme (PS) algorithm is a distributed decentralized algorithm. Also, the PS algorithm cannot and does not take into account the communication delays incurred in transferring jobs from one site to another. The non-cooperative game algorithm is also a distributed decentralized algorithm [3], [5]. In this game, the players are the brokers and each player tries to minimize its own average job completion time independently. In non-cooperative games, each player selfishly maximizes its own utility and makes decisions independently of the other players; these independent decisions may be extremely detrimental to the other players

Cooperative game approach

The grid system consists of p users, n brokers, and m providers (Fig. 1 shows the relationship between users, brokers, and providers). A user generates jobs to be executed by the processors; each user sends a job to a broker to be scheduled for processing. A user may send jobs to more than one broker. A broker, on the other hand, receives jobs from a set of users and assigns them to the providers in the grid system; every time a job is received from a user, the broker decides which provider will process the job and sends the job to that provider. Ideally, there would be many more users than brokers in the system. As such, the jobs scheduled by the brokers are an aggregate from many users. Finally, a provider executes and processes jobs sent to it. Each provider has a queue that holds jobs to be executed. Each job is then processed on a First Come First Server (FCFS) basis. As shown in Fig. 1 the relationship between users, brokers, and providers.

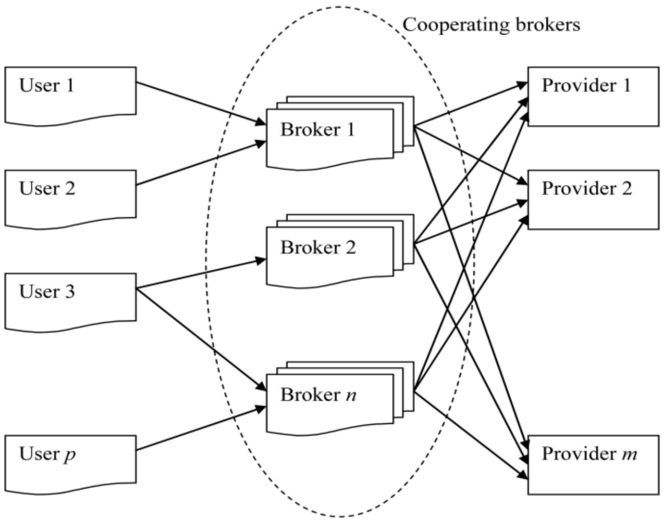

Fig. 1: Relationship between users, brokers and providers

Each user k is assumed to generate jobs with average rate βk (jobs per second) according to a Poisson process and independently of the other users. Jobs are then sent by the user to a broker that dispatches them to the providers. Each broker dispatches jobs to the providers at an average rate λi, which is an aggregation of jobs being sent by users to the broker. Depending on the computational power provided by the providers, each provider j executes jobs at an average rate μj (jobs per second).

In this model, the job execution time of the applications running on the system can take any distribution (having finite mean and variance). Each provider can therefore be modeled as an M/G/1 queuing system. The distribution of job for the application, once chosen, is consistent throughout the system. Communications between users, brokers, and providers are done through the underlying grid communication network, which is heterogeneous in nature. Certainly, the users, brokers, and providers are fully interconnected, meaning that there exists a communication path between any two entities in the grid system. Users generate jobs that are sent to the brokers, who in turn send them to the providers. For stability, however, we have the condition/constraint that jobs must not be generated faster than the system can process them (otherwise, the queues at the providers will build up to infinity):

$$\sum_{i=1}^{n} \lambda i < \sum_{j=1}^{m} \mu j \qquad (1)$$

Where, λi represents the average arrival rate of jobs (in jobs per second) at broker i and μj is the average processing rate of jobs at provider j. Another stability criterion is that the rate of jobs sent to a provider j must not exceed the rate at which jobs can be executed by the provider j.

$$\phi j < \mu j \qquad (2)$$

$$\phi j \geq 0 \qquad (3)$$

Where, ϕj represents the rate of jobs sent to processor j and obviously cannot be negative. All of the jobs generated by the system must also be executed by the providers, which leads to the following constraint:

$$\sum_{i=1}^{n} \lambda i = \sum_{j=1}^{m} \phi j \qquad (4)$$

Due to the cooperative nature of proposed algorithm, the algorithm is thought of as a centralized algorithm.

Pareto Optimal and Fair Job Allocation Algorithm: Grid scheduling is a process responsible for assigning users jobs onto available Grid resources as shown in Fig 2. The goal of this process is to maximize various optimization criteria such as machine usage, fairness and flow time or to guarantee non trivial QoS. Effective scheduling in the Grid environment is a complex problem which is not fully and efficiently solved in nowadays production systems. Finding of optimal solution is a complete problem which is practically intractable for larger sets of jobs.

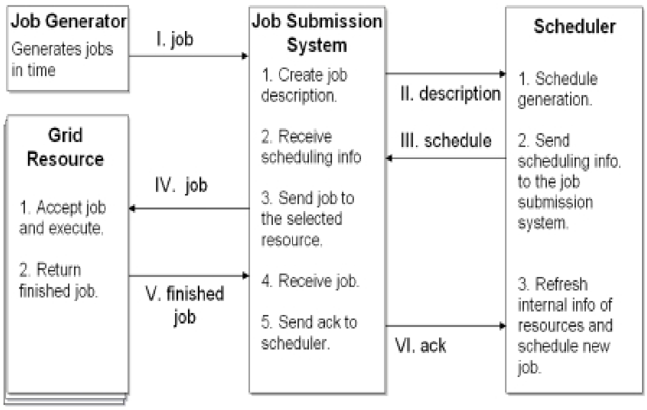
Job Generator
Generates jobs in time
I. job
Job Submission System
1. Create job description.
2. Receive scheduling info
3. Send job to the selected resource.
4. Receive job.
5. Send ack to scheduler.
II. description
III. schedule
VI. ack
Scheduler
1. Schedule generation.
2. Send scheduling info. to the job submission system.
3. Refresh internal info of resources and schedule new job.
Grid Resource
1. Accept job and execute.
2. Return finished job.
IV. job
V. finished job

Fig. 2: Grid Scheduling

For an M/G/1 queuing system [6], the average processing time of a job including waiting time at the queue at a provider j is given by

$$Fj = \overline{hj} + \frac{\overline{hj}^2 . \phi j}{2(1 - \overline{hj} . \phi j)} \qquad (5)$$

Where, hj = 1 / µj is the mean of the job execution distribution, hj^2 is the second moment of the distribution and ϕj is the average rate of jobs (in jobs per second) at provider j.

The expected transfer time of a job from any player to provider j is given by

$$Lj = b / Cj \qquad (6)$$

Where, b represents bits of data to be transferred and Cj represent certain bandwidth in bit per second available for grid jobs. Therefore, the average completion time of jobs for provider j involves the execution time of the job, the waiting time at the queue, and the transfer time of a job to the provider is given by

$$Dj = Fj + Lj \qquad (7)$$

A maximum expected service time Dj^o is calculated using ϕjmax which represents the maximum rate of jobs (in jobs per second) a provider j is willing to accept in order to maintain an acceptable level of QoS. For the cooperative game, the Nash bargaining solution is determined by solving the following optimization problem, given by $\max_{\phi}D$

$$= \sum_{j=1}^{m} ln\,(Djo - Dj) \qquad (8)$$

Cooperative Job Allocation

Finally ϕj is given by the following equation:

$$\phi j = \frac{1}{2\overline{hj}} + \frac{\phi jmax}{2} - \frac{\sqrt{\overline{hj} . \phi jmax - 1}\sqrt{\overline{hj}\,(\alpha . \phi jmax - 4) - \alpha}}{2\sqrt{\alpha . \overline{hj}}} \qquad (9)$$

Where, jmax is the maximum; α represents the Lagrange multiplier. The Lagrange multiplier is needed to choose the equality and inequality constraint. The above equation (9) determines the amount of jobs that should be allocated to each provider j in order to simultaneously maximize the QoS level of all providers. Once this is established, the amount of jobs that should be sent from each broker to each provider has to be determined. Another criterion for the system is set to determine the user's fairness criterion. This fairness criterion is concerned with the users and brokers and is different from the fairness axioms of the Nash bargaining solution, which is concerned with the system providers.

Fairness to users

The fairness is achieved when the average job completion time for each of the brokers is the same. If one broker has a lower average job completion time and another has a higher average job completion time, then the job allocation scheme can be considered as unfair as it gives some brokers an advantage while it gives some other brokers a disadvantage. Fairness is another important performance measure for job allocation schemes beside average job completion time and QoS. A fairness index is calculated by

$$FI = \frac{(\sum_{i=1}^{n} Ti)^2}{n \sum_{i=1}^{n} Ti^2} \qquad (10)$$

Where, Ti is the average job completion time of player i. If a job allocation scheme is 100 percent fair, then FI is 1.0. A fairness index close to 1.0 indicates a relatively fair job allocation scheme.

Experiments

In this experiment of Cooperative Game (CG) approach, the average system load is set to 60 percent. The number of participating providers versus system load is shown in Fig 3.

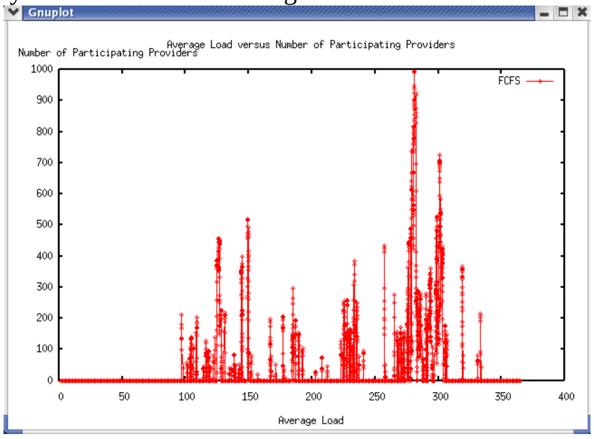

Fig. 3: Number of participating providers versus system load

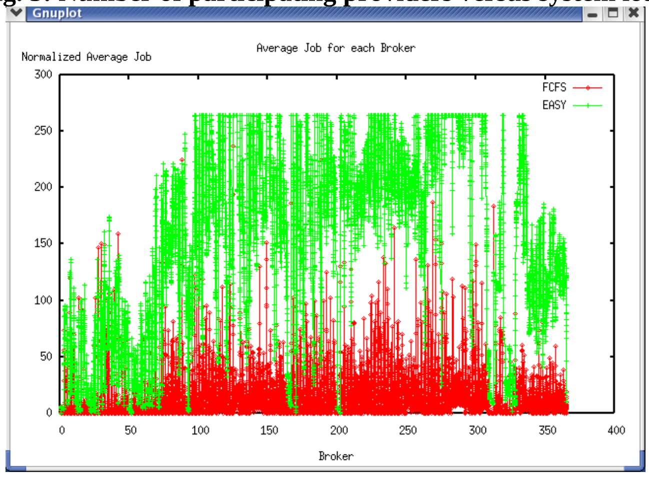

Fig. 4: Average job completion time for each broker at system load as 60%

The completion time of a job includes the waiting time at the queue and the execution time of the job itself. In Fig. 4, the average job completion time for each broker at system load 60%, for FCFS scheduling and EASY scheduling schemes are compared and is normalized by dividing each broker's value by the average job time of all the brokers for the each scheme. The EASY scheduler, which uses aggressive backfilling and small jobs are moved ahead to fill in the scheduler, provided the first job in the queue is not delayed. The comparison of the scheduler schemes are done with a general workload model and with specific workload traces.

A general trend that can be seen is that there is a rapid increase in the system wide average job completion time as the system nears full capacity. As the system nears full capacity, the average queue length at the providers will get longer, resulting in longer queue waiting time. At low system loads, only the fastest providers are fit to participate in job processing. The slower ones only slow down the whole system. However, as the system load increases, more and more providers are able to participate; at 60% load all of the providers in the system are able to participate

III. CONCLUSION

This paper discussed a cooperative game theoretic algorithm to solve the QoS sensitive grid job allocation problem. The proposed algorithm is fair to all users and represents a Pareto optimal solution to the QoS objective. In terms of average job completion time, the proposed algorithm (CG) generally gives better performance than either the noncooperative algorithm (NG) or the proportional-sharing algorithm (PS). While the proposed algorithm gives a Pareto optimal solution to the QoS objective in a grid environment wherein the multitude of sites have different ownerships with their own goals, interests, and priorities, the kind of algorithm that is most appropriate would be dependent on the application and the goals of the users, brokers, and providers.

REFERENCES

[1] Riky Subtrata, Albert Y.Zomaya, Bjorn Landfeldt, "A Cooperative Game Framework for QoS Guided Job Allocation Schemes in Grids", *IEEE Transactions on Computers,* Vol. 57, No. 10, October 2008.

[2] T. Roughgarden and E.Tardos, "How Bad is Selfish Routing", *J.ACM*, Vol 49, pp. 236-259, 2002.

[3] D. Grosu and A.T. Chronopoulos, "Non-cooperative Load Balancing in Distributed Systems," *J. Parallel and Distributed Computing,* vol. 65, pp. 1022-1034, 2005.

[4] D. Grosu, A.T. Chronopoulos, and M.Y. Leung, "Load Balancing in Distributed Systems: An Approach Using Cooperative Games," *Proc. 16th Int'l Parallel and Distributed Processing Symp.*, pp. 501- 510, 2002.

[5] A. Akella, R. Karp, C. Papadimitriou, S. Seshan, and S. Shenker, "Selfish Behavior and Stability of the Internet: A Game-Theoretic Analysis of TCP," Proc. ACM SIGCOMM '02, pp. 117-130, 2002.