A Two-Level Load Balancing Policy For Grid Computing

Said Fathy El-Zoghdy*
Computer Science Dep., College of Computers & Information Technology, Taif University,
Taif, KSA
Alzoghdy@tu.edu.sa

Sultan Aljahdali
Computer Science Dep., College of Computers & Information Technology, Taif University,
Taif, KSA
Aljahdali@tu.edu.sa

Abstract— This paper addresses the problem of scheduling and load balancing in heterogeneous computational grids. We proposed a two-level load balancing policy for the multi-cluster grid environment where computational resources are dispersed in different administrative domains or clusters which are located in different local area networks. The proposed load balancing policy takes into account the heterogeneity of the computational resources. It distributes the system workload based on the processing elements capacity which leads to minimize the overall job mean response time and maximize the system utilization and throughput at the steady state. To evaluate the performance of the proposed load balancing policy, an analytical model is developed. The results obtained analytically are validated by simulating the model using Arena simulation package. The results show that the overall mean job response time obtained by simulation is very close to that obtained analytically. Also, the simulation results show that the performance of the proposed load balancing policy outperforms that of the random and uniform distribution load balancing policies in terms of mean job response time. The improvement ratio decreases as the system workload increases.

Keywords- grid computing; resource management; load balancing; performance evaluation; queuing theory; simulation models

I. INTRODUCTION

The rapid development in computing resources has enhanced the performance of computers and reduced their costs. This availability of low cost powerful computers coupled with the advances and popularity of the Internet and high speed networks has led the computing environment to be mapped from the traditionally distributed systems and clusters to the computing Grid environments. Grid computing is a form of distributed computing that involves coordinating and sharing computational power, data storage and network resources across dynamic and geographically widely dispersed organizations [1]. It allows the management of heterogeneous, geographically distributed and dynamically available computational resources which may belong to different individuals and institutions to solve large-scale scientific applications.

Due to uneven task arrival patterns and unequal computing capacities and capabilities, the computers in one grid site may be heavily loaded while others in a different grid site may be lightly loaded or even idle. It is therefore desirable to transfer some jobs from the heavily loaded computers to the idle or lightly loaded ones in the grid environment aiming to efficiently utilize the grid resources and minimize the average job response time. The process of load redistribution is known as load balancing [3,4,5].

Grid scheduling and load balancing is NP-complete, so there is no best scheduling algorithm for all grid computing systems[1]. An alternative is to select an appropriate load balancing algorithm to use in a given grid environment because of the Grid characteristics of the tasks, machines and network connectivity, see figure 1.

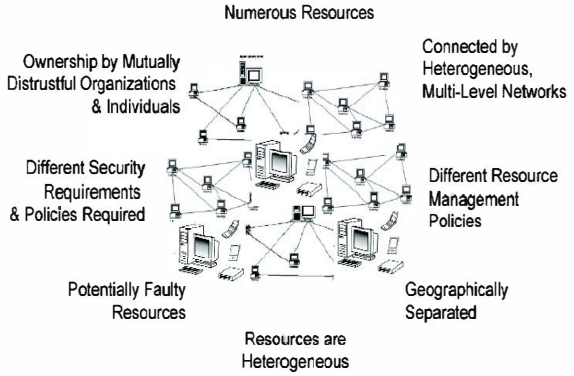

Figure 1. Characteristics of Grids

Although load balancing problem in traditional distributed systems has been intensively studied [5-13], new challenges in Grid computing still make it an interesting topic, and many research projects are interested in this problem.

In this paper, we present a decentralized load balancing policy for the grid computing environment. The proposed policy tends to improve grid resources utilization and hence maximizes throughput. We focus on the steady-state mode, where the number of jobs submitted to the grid is sufficiently large and the arrival rate of jobs does not exceed the grid overall processing capacity [10]. As in [10], steady-state mode will help us to derive optimality for the proposed load balancing policy. The class of problems addressed by the proposed load balancing policy is the computation-intensive and totally independent jobs with no communication between them. An analytical model is presented. This model is based

978-1-4673-1520-3/12/$31.00 ©2012 IEEE

on queuing theory. We are interested in computing the overall mean job response time of the grid system. The results obtained analytically are validated by simulating the model using Arena simulation package.

The rest of this paper is organized as follows: Section II presents related work. Section III describes the structure of grid computing service model. Section IV introduces the proposed grid load balancing policy. Section V presents the analytical queuing model. In section VI, we present performance evaluation of the proposed load balancing policy. Finally, Section VII summarizes this paper.

II. RELATED WORK AND MOTIVATIONS

Many papers have been published recently to address the problem of load balancing in Grid computing environments. Some of the proposed grid computing load balancing policies are modifications or extensions to the traditional distributed systems load balancing policies. In [11], a decentralized model for heterogeneous grid has been proposed as a collection of clusters. In [2], the authors presented a tree-based model to represent any Grid architecture into a tree structure. The model takes into account the heterogeneity of resources and it is completely independent from any physical Grid architecture. However, they did not provide any job allocation procedure. Their resource management policy is based on a periodic collection of resource information by a central entity, which might be communication consuming and also a bottleneck for the system. In [15], the authors proposed a ring topology for the Grid managers which are responsible for managing a dynamic pool of processing elements (computers or processors).The load balancing algorithm was based on the real computers workload. In [12], the authors proposed a hierarchical structure for grid managers rather than ring topology to improve scalability of the grid computing system. They also proposed a job allocation policy which automatically regulates the job flow rate directed to a given grid manager.

In this paper we propose a decentralized load balancing policy that can cater for the following unique characteristics of practical Grid Computing environment:

- **Large-scale.** As a grid can encompass a large number of high performance computing resources that are located across different domains and continents, it is difficult for centralized model to address communication overhead and administration of remote workstations.
- **Heterogeneous grid sites.** There might be different hardware architectures, operating systems, computing power and resource capacity among different sites.
- **Effects from considerable transfer delay.** The communication overhead involved in capturing load information of sites before making a dispatching decision can be a major issue negating the advantages of job migration. We should not ignore the considerable dynamic transfer delay in disseminating load updates on the Internet.

III. GRID COMPUTING SERVICE STRUCTURE

The grid computing model which we consider is a large-scale computing service model that is based on a hierarchical geographical decomposition structure. Every user submits his computing jobs and their hardware requirements to the Grid Computing Service (GCS). The GCS will reply to the user by sending the results when it finishes the execution of the jobs. In the GCS, jobs pass through four phases which can be summarized as follows:

A. Task submission phase

Grid users can submit their jobs through the available web sites browsers. This makes the job submission process easy and accessible to any number of clients.

B. Task allocation phase

Once the GCS receives a job, it looks for the available resources (computers or processors) and allocates the suitable resources to the task.

C. Task execution phase

Once the needed resources are allocated to the task, it is scheduled for execution on that computing site.

D. Results collection phase

When the execution of the jobs is finished, the GCS notify the users by the results of their jobs. Three-level Top-Down view of the considered grid computing model is shown in figure 2 and can be explained as follows:

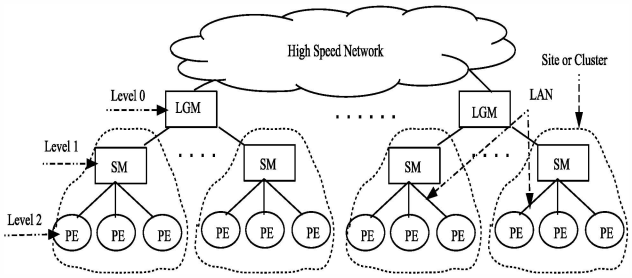

Figure 2. Grid Computing Model Structure

Level 0: Local Grid Manager (LGM)

Any LGM manages a pool of Site Managers (SMs) in its geographical area. The role of LGM is to collect information about the active resources managed by its corresponding SMs. LGMs are also involved in the task allocation and load balancing process in the grid. New SMs can join the GCS by sending a join request to register themselves at the nearest parent LGM.

Level 1: Site Manager (SM)

Every SM is responsible for managing a pool of processing elements (computers or processors) which is dynamically configured (i.e., processing elements may join or leave the pool at any time). A new joining computing element to the site should register itself within the SM. The role of the SM is to

collect information about active processing elements in its pool. The collected information mainly includes CPU speed, and other hardware specifications. Also, any SM has the responsibility of allocating the incoming jobs to any processing element in its pool according to a specified load balancing algorithm.

Level 2: Processing Elements (PE)

Any private or public PC or workstation can join the grid system by registering within any SM and offer its computing resources to be used by the grid users. When a computing element joins the grid, it starts the GCS system which will report to the SM some information about its resources such as CPU speed. Within this hierarchy, adding or removing SMs or PEs becomes very flexible and serves both the openness and the scalability of proposed grid computing service model.

The LGMs represent the entry points of computing jobs in the proposed grid computing model. Any LGM acts as a web server for the grid model. Clients (users) submit their computing jobs to the associated LGM using the web browser. According to the available load balancing information, the LGM will pass the submitted jobs to the appropriate SM. The SM in turn distributes these computing jobs according to the available site load balancing information to a chosen processing element for execution. LGMs all over the world may be interconnected using a high-speed network as shown in figure. 2.

As explained earlier, the information of any processing element joining or leaving the grid system is collected at the associated SM which in turn transmits it to its parent LGM. This means that a communication is needed only if a processing element joins or leaves its site. All of the collected information is used in balancing the system workload between the processing elements to efficiently utilize the whole system resources aiming to minimize user jobs response time. This policy minimizes the communication overhead involved in capturing system information before making a load balancing decision which improves the system performance. .

IV. GRID LOAD BALANCING POLICY

We proposed a two-level load balancing policy for the multi-cluster grid environment where clusters are located in different local area networks. The proposed load balancing policy takes into account the heterogeneity of the computational resources. It distributes the system workload based on the processing elements capacity. We assume that the jobs submitted to the grid system are totally independent jobs with no inter-process communication between them, and that they are computation intensive jobs.

To formalize the load balancing policy, we define the following parameters for grid computing service model:

a. Job: Every job is represented by a job Id, number of job instructions NJI, and a job size in bytes JS.

b. Processing Element Capacity (PEC_{ij}): Number of jobs that can be executed by j^{th} PE at full load in i^{th} site per second. The PEC can be calculated using the PEs CPU speed and assuming an Average Number of job Instructions ANJI.

c. Site Processing Capacity (SPC_i): Number of jobs that can be executed by i^{th} site per second. Hence, the SPC_i is calculated by summing the PEC_{ij} of all j^{th} PEs at i^{th} site.

d. Local grid manager Processing Capacity (LPC): Number of jobs that can be executed under the responsibility of the LGM per second. The LPC can be calculated by summing all the SPCs for all the sites managed by the LGM.

The proposed load balancing policy is a multi-level one as it could be seen form figure 3. This policy is explained at each level of the grid architecture as follows:

A. Local Grid Manager Load Balancing Level

Consider a Local Grid Manager (LGM) which is responsible of a group of site managers (SMs). As mentioned earlier, the LGM maintains information about all of its SMs in terms of processing capacity SPCs. The total processing capacity of a LGM is LPC which is the sum of all the SPCs for all the sites managed by that LGM. Based on the total processing capacity of every site SPC, the LGM scheduler distributes the workload among his sites group members (SMs). Let N denotes the number of jobs arrived at a LGM in the steady state. Hence, the i^{th} site workload (S_iWL) which is the number of jobs to be allocated to i^{th} site manager is obtained as follows:

$$S_i WL = N \times \frac{SPC_i}{LPC} \qquad (1)$$

B. Site Manager Load Balancing Level

As it is explained earlier every SM manages a dynamic pool of processing elements (workstations or processors). Hence, it has information about the PECs of all the processing elements in its pool. The total site processing capacity SPC is obtained by summing all the PECs of all the processing elements in that site. Let M be the number of jobs arrived at a SM in the steady state. The SM scheduler will use a load balancing policy similar to that used by the LGM scheduler. This means that the site workload will be distributed among his group of processing elements based on their processing capacity. Using this policy, the throughput of every processing element will be maximized and also its resource utilization will be improved. Hence, the i^{th} PE workload (PE_iWL) which is the number of jobs to be allocated to i^{th} PE is obtained as follows:

$$PE_i WL = M \times \frac{PEC_i}{SPC} \qquad (2)$$

Example: Let N =2000 j/s (job/second) arrive at a LGM with five SMs having the following processing capacities:

SPC_1=450 j/s, SPC_2=600 j/s, SPC_3=475 j/s, SPC_4=625 j/s, and SPC_5=350 j/s.

Hence, LPC= 440+600+475+625+250=2500 j/s. So, the workload for every site will be computed according to equation 1 as follows:

$$S_1 WL = 2000 \times \frac{450}{2500} = 360 \ \ j/s$$

$$S_2 WL = 2000 \times \frac{600}{2500} = 480 \ \ j/s$$

$$S_3 WL = 2000 \times \frac{475}{2500} = 380 \ \ j/s$$

$$S_4 WL = 2000 \times \frac{625}{2500} = 500 \ \ j/s$$

$$S_5 WL = 2000 \times \frac{350}{2500} = 280 \ \ j/s$$

Then workload of every site will be allocated to the processing elements managed by that site based on equation 2. As an example, suppose that the 3rd site contains three PEs having the processing capacities of 120j/s, 250j/s, and 230j/s respectively. Hence the SPC= 240+210+150= 600 t/s. Remember that this site workload equals to 380 t/s as computed previously. So, the workload for every PE will be computed according to equation 2 as follows:

$$PE_1 WL = 380 \times \frac{240}{600} = 152 \ \ j/s$$

$$PE_2 WL = 380 \times \frac{210}{600} = 133 \ \ j/s$$

$$PE_3 WL = 380 \times \frac{150}{600} = 95 \ \ j/s$$

From this simple numerical example, one can see that the proposed load balancing policy allocates more workload to the faster PEs which improves the system utilization and maximizes system throughput.

V. ANALYTICAL MODEL

To compute the mean job response time analytically, we consider a LGM section as simplified grid model. In this model, we will concentrate on the time spent by a job in the processing elements. Consider the following system parameters:

- λ is the external job arrival rate from grid clients to a LGM.
- λ_i is the job flow rate from the LGM to the i^{th} SM which is managed by that LGM.
- λ_{ij} is the job flow rate from the i^{th} SM to the j^{th} PE managed by that SM.
- μ is the LGM processing capacity.
- μ_i is processing capacity of the i^{th} SM.
- μ_{ij} is the processing capacity of the j^{th} PE which is managed by i^{th} SM.
- $\rho=\lambda/\mu$ is the system traffic intensity. For the system to be stable ρ must be less than 1.
- $\rho_i = \frac{\lambda_i}{\mu_i}$ is traffic intensity of the i^{th} SM .
- $\rho_{ij} = \frac{\lambda_{ij}}{\mu_{ij}}$ is traffic intensity of the j^{th} PE which is managed by i^{th} SM.

We assume that the jobs arrive from clients to the LGM according to a time-invariant Poisson process. Jobs arrive at the LGM sequentially, with inter-arrival times which are independent, identically, and exponentially distributed with the arrival rate λ j/s. Simultaneous arrivals are excluded. Every PE in the dynamic site pool will be modeled by an M/M/1 queue. Since jobs that arrive to the LGM will be automatically distributed on the sites managed by that LGM with a routing probability $\Pr S_i = \frac{SPC_i}{LPC}$ according to the load balancing policy (LBP), where i is the site number, hence $\lambda_i = \lambda \times \Pr S_i = \lambda \times \frac{SPC_i}{LPC}$. Again the site i arrivals will also automatically be distributed on the PEs managed by that site with a routing probability $PrE_{ij} = \frac{PEC_{ij}}{SPC_i}$ based on the LBP, where j is the PE number and i is the site number. Hence, $\lambda_{ij} = \lambda_i \times PrE_j = \lambda_i \times \frac{PEC_{ij}}{SPC_i}$. Since the arrivals to LGM are assumed to follow a Poisson process, then the arrivals to the PEs will also follow a Poisson process. We also assume that the service times at the j^{th} PE in the i^{th} SM site is exponentially distributed with fixed service rate μ_{ij} j/s. Note that μ_{ij} represents the PE's processing capacity (PEC) in our load balancing policy. The service discipline is First Come First Serviced. This grid queueing model is illustrated in figure 3.

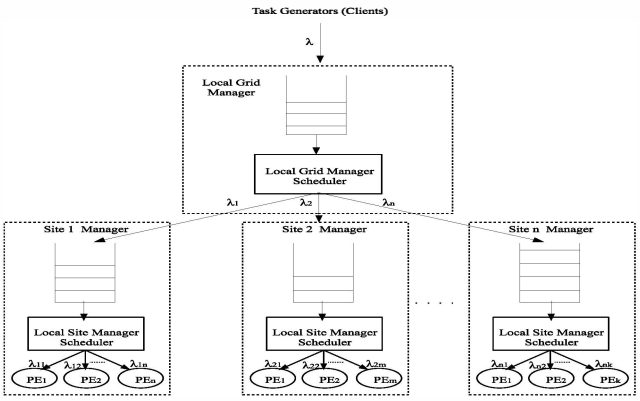

Figure 3. Grid Computing Queueing Model

The state transition diagram of the j^{th} PE in i^{th} site manager is shown in figure 4.

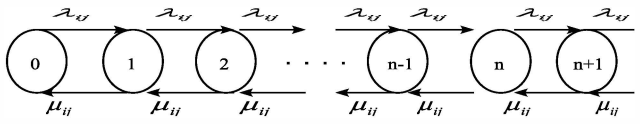

Figure 4. A state transition diagram of j^{th} PE in i^{th} site manager.

As mentioned earlier, we are interested in studying the system at the steady state that is the traffic intensity is less than one i.e., ρ<1. To compute the expected mean job response time, the Little's formula will be used. Let E[T_g] denotes the mean time spent by a job at the grid to the arrival rate λ and E[Ng] denotes the number of jobs in the system. Hence by Little formula, the mean time spent by a job at the grid will be given by equation 3 as follows:

$$E[N_g] = \lambda \times E[T_g] \qquad (3)$$

$E[N_g]$ can be computed by summing the mean number of jobs in every PE at all the grid sites. So, $E[N_g] = \sum_{i=1}^{m} \sum_{j=1}^{n} E[N_{PE}^{ij}]$, where i=1,2,..m, is the number of site managers managed by a LGM and j=1,2,…,n is the number of processing elements managed by a SM and $E[N_{PE}^{ij}]$ is the mean number of jobs in a processing element number j at site number i. Since every PE is modeled as an M/M/1 queue, then $E[N_{PE}^{ij}] = \frac{\rho_{ij}}{1-\rho_{ij}}$, where $\rho_{ij} = \frac{\lambda_{ij}}{\mu_{ij}}$, μ_{ij} =PEC_{ij} for PE number j at site number i. From equation 3, the expected mean job response time is given by:

$$E[T_g] = \frac{1}{\lambda} \times E[N_g] = \frac{1}{\lambda} \times \sum_{i=1}^{m} \sum_{j=1}^{n} E[N_{PE}^{ij}]$$

Note that the stability condition for PE_{ij} is $\rho_{ij} \prec 1$.

VI. Results and Discussion

A. Experimental Environment

The simulation was carried out using the great discrete event system simulator Arena. This simulator allows modeling and simulation of entities in grid computing systems-users, applications, resources, and resource load balancers for design and evaluation of load balancing algorithms.

To evaluate the performance of grid computing system under the proposed load balancing policy, a simulation model is built using Arena simulator. This simulation model consists of one LGM which manages a number of SMs which in turn manages a number of PEs (Workstations or Processors). All simulations are performed on a PC (Core 2 Processor, 2.73GHz, 1GB RAM) using Windows xp OS.

B. Simulation Results and Analysis

We assume that the external jobs arrive to the LGM sequentially, with inter-arrival times which are independent, identically, and exponentially distributed with mean 1/λ j/s. Simultaneous arrivals are excluded. We also assume that the service times of LGMs are independent and exponentially distributed with mean μ j/s.

The performance of the grid computing system under the proposed load balancing policy is compared with two other policies namely; Random distribution load balancing policy and Uniform distribution load balancing policy.

In the Uniform distribution load balancing policy the job flow rate (routing probability) from LGM to its SMs is fixed to the value $\frac{1}{n_s}$, where n_s is the number of SMs in the grid computing service model. Also the job flow rate (routing probability) from any SM to its PEs is fixed to the value $\frac{1}{n_{PE}}$, where n_{PE} is the number of PEs which are managed by that site.

In the Random distribution load balancing policy a resource for job execution is selected randomly without considering any performance metrics to that resource or to the system. This policy is explained in [15]. However, in the proposed load balancing policy all the arriving jobs from clients to the LGMs are distributed on the SMs based on their processing capacity to improve utilization aiming to minimize mean job response time.

The grid system built in our simulation experiment has 1 LGM, 3 SMs having 4, 3, and 5 PEs respectively. We fixed the total grid system processing capacity μ=LPC=1700 j/s. First, the mean job response time under the proposed load balancing policy is computed analytically and by simulation as shown in figure 5. From that figure, we can see that the response times obtained by the simulation approximate that obtained analytically. The obtained simulation results satisfy 95% confidence level.

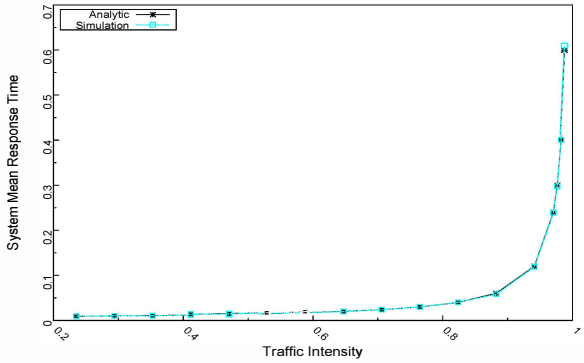

Figure 5. Analytic versus simulation mean job response times

Also, from figure 5, we can notice that the proposed load balancing policy is asymptotically optimal because its saturation point (λ/μ)≈1 (traffic intensity) is very close to the saturation level of the grid computing model.

Using the same grid model parameters setting of our simulation experiment, the performance of the proposed load balancing policy is compared with that of the Uniform distribution, and Random distribution as shown in figure 6. From that figure we can see that proposed LBP outperforms the Random distribution and Uniform distribution LBPs in terms of system mean job response time. It is also noticed that the system mean response time obtained by the uniform LBP lies between that of the proposed and random distribution LBPs.

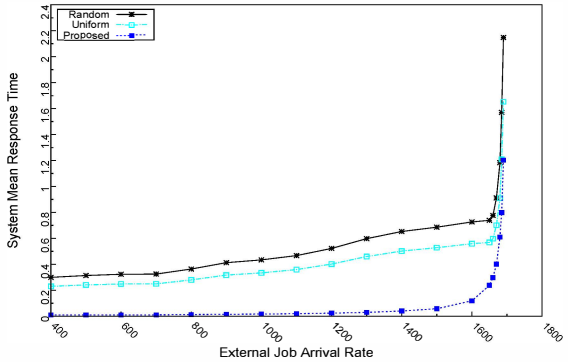

Figure 6. System mean job response time versus job arrival rate

To evaluate how much improvement obtained in the system mean job response time as a result of applying the proposed LBP, we computed the improvement ratio $(T_U - T_p)/T_U$, where T_U is the system mean job response time under uniform distribution LBP and T_p is the system mean job response time under proposed LBP, see figure 7. From that figure, we can see that the improvement ratio decreases as the system workload increases since the system gradually approaches its saturation point. This result was anticipated since the proposed LBP distributes the system workload based on the processing elements capacity which leads to maximizing system resources utilization ratio and as a result system mean job response time is minimized. In contrast, the Random distribution policy distributes the system workload randomly on the system PE without putting any performance metric in mind which may lead to unbalanced system workload distribution which leads to poor resources utilization and hence, the system performance is affected. This situation appears clearly as the system workload increases. Also, the Uniform distribution policy distributes the system workload equally on the PEs without putting their processing capacity or any workload information in mind which repeats the same situation as the random distribution LBP. To be fair, we must say that according to the obtained simulation results, the performance of the Uniform distribution LBP is much better that that of the Random distribution LBP.

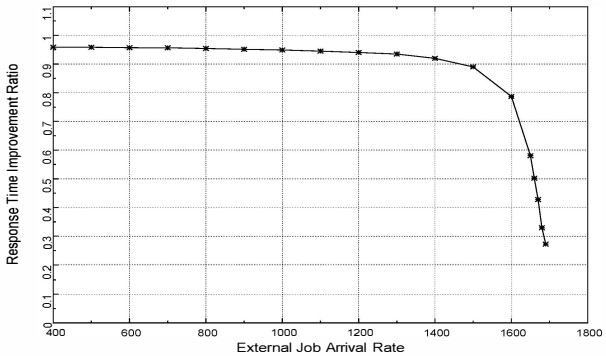

Figure 7. System mean response time improvement ratio

VII. CONCLUSION

A two-level load balancing policy for the grid computing is proposed in this paper. It distributes the grid workload based on the processing elements capacity which leads to minimize the overall job mean response time and maximize the system utilization and throughput at the steady state. An analytical model is developed to compute the expected mean job response time in the grid system. A simulation model is built using Arena simulator to evaluate the performance of the proposed load balancing policy and validate the analytic results. The results show that the overall mean job response time obtained analytically is very close to that obtained by the simulation. Also, it shows that the performance of the proposed load balancing outperforms that of the Random and Uniform distribution load balancing policies in terms of mean job response time. It improves the overall job mean response time. The improvement ratio gradually decreases as the system workload increases because the system gradually approaches its saturation point.

REFERENCES

[1] Raksha S., Vishnu K. S., Manoj K. M., and Prachet B., "A Survey of Job Scheduling and Resource Management in Grid Computing", World Academy of Science, Engineering and Technology, Vol. 64, pp. 461-466, 2010.

[2] B. Yagoubi and Y. Slimani, "Task Load Balancing Strategy for Grid Computing", Journal of Computer Science, vol. 3, no. 3: pp. 186-194, 2007.

[3] K. Li, "Optimal load distribution in nondedicated heterogeneous cluster and grid computing environments", Journal of Systems Architecture, vol. 54, pp. 111–123, 2008.

[4] Y. Li, Y. Yang, M. Ma, and L. Zhou, "A hybrid load balancing strategy of sequential jobs for grid computing Environments", Future Generation Computer Systems, vol. 25, pp.) 819_828, 2009.

[5] H. Kameda, J. Li, C. Kim, and Y. Zhang, "Optimal Load Balancing in Distributed Computer Systems", Springer, London, 1997.

[6] S. F. El-Zoghdy, Studies on Braess-Like Paradoxes for Non-Cooperative Dynamic Load Balancing in Distributed Computer Systems, Proc. of the IASTED Inter. Conf. on Parallel and Distributed Computing and Networks, pp. 238-243, 2006.

[7] S. F. El-Zoghdy, H. Kameda, and J. Li, A comparative study of static and dynamic individually optimal load balancing policies, Proc. of the IASTED Inter. Conf. on Networks, Parallel and Distributed Processing and Applications, pp. 200-205. 2002.

[8] J. Li and H. Kameda, "A Decomposition Algorithm for Optimal Static Load Balancing in Tree Hierarchy Network Configurations", IEEE Trans. Parallel and Distributed Systems, vol. 5, no. 5, pp.540-548, 1994.

[9] J. Li and H. Kameda, "Load Balancing Problems for Multiclass Jobs in Distributed/Parallel Computer Systems", IEEE Trans. Comput., vol. 47, no. 3, pp322-332, 1998.

[10] R. Mirchandaney, D. Towsley, and J. A. Stankovic, "Adaptive Load Sharing in Heterogeneous Distributed Systems", J. Parallel and Distributed Computing", vo. 9, pp.331-346, 1990.

[11] O. Beaumont, A. Legrand, L. Marchal and Y. Robert. Steady-State Scheduling on Heterogeneous Clusters. Int. J. of Foundations of Computer Science, Vol. 16, No.2,pp. 163-194, 2005.

[12] A. Touzene, H. Al Maqbali, "Analytical Model for Performance Evaluation of Load Balancing Algorithm for Grid Computing", Proc. of the 25th IASTED Inter. Multi-Conference: Parallel and Distributed Computing and Networks, pp. 98-102, 2007.

[13] J. Balasangameshwara, N. Raju, " A Decentralized Recent Neighbour Load Balancing Algorithm for Computational Grid", Int. J. of ACM Jordan, vol. 1,no. 3, pp. 128-133, 2010.

[14] A. Touzene, S. Al Yahia, K.Day, B. Arafeh, "Load Balancing Grid Computing Middleware", IASTED Inter. Conf. on Web Technologies, Applications, and Services, 2005.

[15] Zikos, S., Karatza, H.D., Resource allocation strategies in a 2-level hierarchical grid system. In: Proceedings of the 41st Annual Simulation Symposium (ANSS), April 13–16, 2008. IEEE Computer Society Press, SCS, pp. 157–164.