# A Load Balancing Policy for Heterogeneous Computational Grids

Said Fathy El-Zoghdy
Mathematics and Computer Science Department
Faculty of Science, Menoufia University
Shebin El-Koom, Egypt.
Email: Elzoghdy@yahoo.com

***Abstract*—Computational grids have the potential computing power for solving large-scale scientific computing applications. To improve the global throughput of these applications, workload has to be effectively balanced among the available computational resources in the grid environment. 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 that existed physically in various LANs. The proposed load balancing policy reflects the heterogeneity of the computational resources in deciding load distributions decisions. It balances the system's load according to the computing nodes capacity. Therefore, system's overall job response time and utilization are minimized and maximized respectively. An analytical model is developed to gauge the performance of the proposed load balancing policy. 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 results revealed that the performance of the suggested load balancing strategy outperforms that of the random and uniform distribution load balancing strategies in terms of mean job response time. The improvement ratio increases as the system workload increases and the maximum improvement ratio obtained is about 72% within the studied system parameters values.**

***Keywords- Computational grids; resource management; load distribution; queuing theory; simulation model.***

## 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 Grid computing environments. The Grid computing has emerged as an attractive computing paradigm [1,2]. The Computing Grid, a kind of grid environments, aims to solve the massive computation problems. It can be defined as hardware and software infrastructure which provides dependable, consistent, pervasive and inexpensive access to geographically widely distributed computational resources. These resources may belong to various individuals and institutions to solve large-scale scientific applications. Such applications may contain Nano-materials, massive data, DNA research and simulated meteorology systems.

Basically, grid resources are physically distributed workstations or servers, which are gathered to works as an integrated processing system. The primary motivation of grid computing system is to support clients and programs with universal and continuous access to enormous set of high performance computational resources [1-4]. Computational grids offer many types of services. These services are provided by the servers in the grid computing system. The servers are generally heterogeneous as they may have different CPUs computing power, storage size, etc. [4].

As a consequence of the unequal task arrival rates and difference of 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 needed to shift some jobs from the heavily loaded computers to others from the lightly loaded set aiming to efficiently employ the resources and consequently minimize the average job response time. The load shifting process is recognized as load balancing (LB)[4,5,6].

In general, LB policies can be categorized into centralized or distributed. In centralized policies, the system has only one LB decision maker which has a global view of the system load information. In such polices, the system's incoming jobs are automatically forwarded to the decision maker, which balances the load among different processing nodes aiming to improve system average response time. These strategies are favorable if the communication cost is unneglectable or not important as in shared memory multiprocessor systems. Various scholars claim that, the centralized policies are not scalable as if the number of processing nodes in the system increases, the decision maker may fail [6-9,16].

In the distributed (decentralized) LB policies on contrary, all computers (nodes) in the system participate in taking the load distribution decisions. As a result, the decisions of load redistribution are not centralized in one node. Therefore, various scholars think that, the distributed LB strategies are better from the scalability and fault tolerance points of view than the centralized ones. But at the same time, it is very costly to enable every computer in a distributed system from collecting the state information of the entire system. As a

consequence, in the distributed load distribution strategies, every computing node receives its incoming tasks and after that, it decides to shift a part of its load based on the partial or complete information it has about the overall system's load distribution [17-19]. It appears that this policy is closely related to the individually optimal policy, in that each job (or its user) optimizes its own expected mean response time independently of the others [4-10].

Although the problem of balancing loads in conventional distributed environments has been studied massively [6-14], 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 distributed LB policy for the grid computing environment. The proposed policy tends to improve grid resources utilization and hence maximizes throughput. It concentrates on studying the proposed model in its steady state. In this state, the total number of admitted jobs to the computational grid is adequately large and the incoming jobs rate cannot surpass the entire processing capacity of system [15]. As in [15], steady-state mode will help us to derive optimality for the proposed LB policy. The suggested LB strategy addresses the problem's class of massive computation and entirely independent jobs that has no in between communications. An analytical model is presented. This model is based on queuing theory. We are interested in computing the overall mean job response time. The results obtained analytically are validated by simulating the model using Arena simulation package.

The structure of this paper's remaining sections is as follows: Section II gives major and recent related works. Section III presents the architecture of suggested computational grid model. Section IV introduces the proposed grid LB policy. Section V discusses the analytical queuing model. In Section VI, we assess the performance evaluation of the proposed LB policy. Lastly, Section VII concludes this paper.

## II. Related Work and Motivations

LB has been studied massively in the conventional distributed systems literature for more than two decades. Various policies and algorithms have been suggested, analyzed and implemented in a number of studies [6-14]. It is more challenging to achieve LB in Grid systems than in conventional distributed computing ones because of the heterogeneity and the complicated dynamic nature of the Grid systems. The problem of LB in grid architecture is addressed by assigning loads in a grid without neglecting the communication overhead in collecting the load information. It considers load index as a decision factor for scheduling of jobs in a cluster and among clusters.

Many papers have been published recently to address the problem of LB in Grid computing environments. Some of the proposed computational grids LB policies are modifications or extensions to the conventional distributed systems LB policies.

In [23], a decentralized model for heterogeneous grid has been proposed as a collection of clusters. In [1], the authors employed the tree structure in representing a computational grid model. Their suggested model considers the heterogeneity of system's computational nodes but it is entirely autonomous of any real grid structure. Though, they did not offer any job assigning algorithm. Their resource controlling strategy relies on the periodic gathering of node's information via manager node. Such strategy suffers from having massive communication overhead. Indeed, the manager node may represent a single point of failure to the system. The authors in [24] suggested utilizing ring topology in guiding managers of computational grids. These managers are in charge of controlling a dynamic set of computing nodes (computers or processors). The process of taking workload balancing decisions in their model relies on real load of computing nodes in the system. In [21], 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 LB policy that can cater for the next exclusive features of applied computational grids systems:

- **Large-scale.** As a grid can involve a huge set of advanced computational nodes that really existed in various distributed sites; where it is impossible for the centralized systems to deal with the problems of having enormous communication overhead and remotely administrating distant stations.
- **Heterogeneous grid sites.** There might be various hardware specifications, OS and processing speeds in 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 studied computational grid model is a large-scale service one and it relies on a geographical hierarchy decomposition arrangement. 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 clients can admit the jobs via any web explorer. This facilitates the job admission procedure and makes the system reachable to all users.

*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*

The GCS informs the user by his task's results immediately upon the execution is completed.

Three-level Top-Down view of the considered grid computing model is shown in Fig. 1 and can be explained as follows:

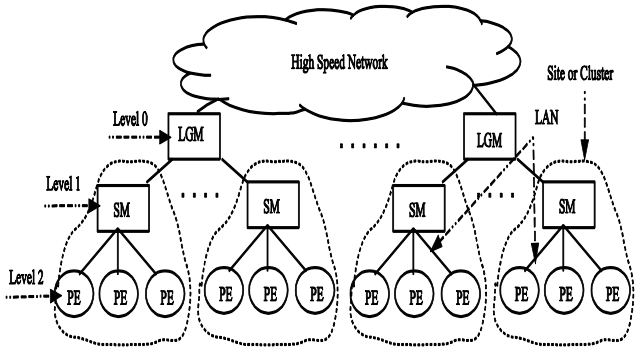


Figure 1. Grid Computing Model Structure

- **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 LB 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 in charge of controlling a set of computing nodes that are configured dynamically (i.e., any computing node can enter or disuse the system as desired). A new joining computing node 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 LB 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, the addition or removal of a SMs or PEs is an easy process and ensures scalability of suggested model of computational grids.

The LGMs represent the entry points of computing jobs in the proposed grid computing model. Any LGM works like a server in the web for the grid model. Any client can admit his jobs to the associated LGM using the web explorer. According to the available LB information, the LGM will pass the arrived jobs to the appropriate SM. The SM in turn distributes these computing jobs according to the available site LB information to a chosen processing element for execution. LGMs allover the world may be interconnected using a high-speed network as shown in Fig. 1.

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 entire system resources aiming to minimalize user's jobs response time. This policy minimizes the communication overhead involved in capturing system information before making a LB decision which improves the system performance. .

## IV. GRID LOAD BALANCING POLICY

We proposed a two-level LB policy for the multi-cluster grid environment where clusters are located in different local area networks. The proposed LB policy takes into account the heterogeneity of the computational resources. It balances the system's load according to capacity of computing nodes. We assume that the jobs admitted to the grid system are entirely independent ones with no inter-process communication in between and that they are massive computation jobs.

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

1. **Job:** Every job is represented by a job Id, number of job instructions NJI, and a job size in bytes JS.
2. **Processing Element Capacity ($PEC_{ij}$):** Number of jobs that can be executed by the $j^{th}$ PE at full load in the $i^{th}$ site per second. The PEC can be calculated using the PEs CPU speed and assuming an Average Number of job Instructions ANJI.
3. **Site Processing Capacity ($SPC_i$):** Number of jobs that can be executed by the $i^{th}$ site per second. Hence, the $SPC_i$ can be calculated by summing all the PECs for all the PEs managed the $i^{th}$ site**.**
4. **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 that LGM.

The proposed LB policy is a multi-level one as it could be seen form Fig 2. 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 ($S_i$WL) 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 LB 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 ($PE_i$WL) 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 =1500 *j/s* (job/second*)* arrive at a LGM with five SMs having the following processing capacities:

*$SPC_1$=440 j/s, $SPC_2$=260 j/s, $SPC_3$=320 j/s, $SPC_4$=580 j/s, and $SPC_5$=400 j/s.*

Hence, LPC= 440+260+320+580+400=2000 *j/s.* So, the workload for every site will be computed according to equation 1 as follows:

$$S_1 WL = 1500 \times \frac{440}{2000} = 330 \quad j/s$$

$$S_2 WL = 1500 \times \frac{260}{2000} = 195 \quad j/s$$

$$S_3 WL = 1500 \times \frac{320}{2000} = 240 \quad j/s$$

$$S_4 WL = 1500 \times \frac{580}{2000} = 435 \quad j/s$$

$$S_5 WL = 1500 \times \frac{400}{2000} = 300 \quad j/s$$

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 fifth site contains three PEs having the processing capacities of 90*j/s,* 200*j/s*, and *150j/s* respectively. Hence the SPC= 90+200+150= 440 *t/s.* Remember that this site workload equals to 300 *t/s* as computed previously. So, the workload for every PE will be computed according to equation 2 as follows:

$$PE_1 WL = 300 \times \frac{180}{400} = 135 \quad j/s$$

$$PE_2 WL = 300 \times \frac{120}{400} = 90 \quad j/s$$

$$PE_3 WL = 300 \times \frac{100}{400} = 75 \quad j/s$$

From this simple numerical example, one can see that the proposed LB 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 one LGM section as a 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:

- $\lambda$ is the external job arrival rate from grid clients to the LGM.
- $\lambda_i$ is the job flow rate from the LGM to the $i^{th}$ SM which is managed by that LGM.
- $\lambda_{ij}$ is the job flow rate from the $i^{th}$ SM to the $j^{th}$ PE managed by that SM.
- $\mu$ is the LGM processing capacity.
- $\mu_i$ is processing capacity of the $i^{th}$ SM.
- $\mu_{ij}$ is the processing capacity of the $j^{th}$ PE which is managed by the $i^{th}$ SM.
- $\rho=\lambda/\mu$ is the system traffic intensity. For the system to be stable $\rho$ 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 $\mathrm{PrS}_i = \frac{\mathrm{SPC}_i}{\mathrm{LPC}}$ according to the LB policy, where *i* is the site number, hence $\lambda_i = \lambda \times \Pr S_i = \lambda \times \frac{\mathrm{SPC}_i}{\mathrm{LPC}}$. Again the site *i* arrivals will also automatically be distributed on the PEs managed by that site with a routing probability $\mathrm{PrE}_{ij} = \frac{\mathrm{PEC}_{ij}}{\mathrm{SPC}_i}$ based on the LBP, where *j* is the PE number and *i* is the site number. Hence, $\lambda_{ij} = \lambda_i \times \mathrm{PrE}_j = \lambda_i \times \frac{\mathrm{PEC}_{ij}}{\mathrm{SPC}_i}$.

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 is exponentially distributed with fixed service rate $\mu_{ij}$ *j/s*. Note that $\mu_{ij}$ represents the PE's processing capacity (PEC) in our LB policy. The service discipline is First Come First Serviced. This grid queueing model is illustrated in Fig 2.

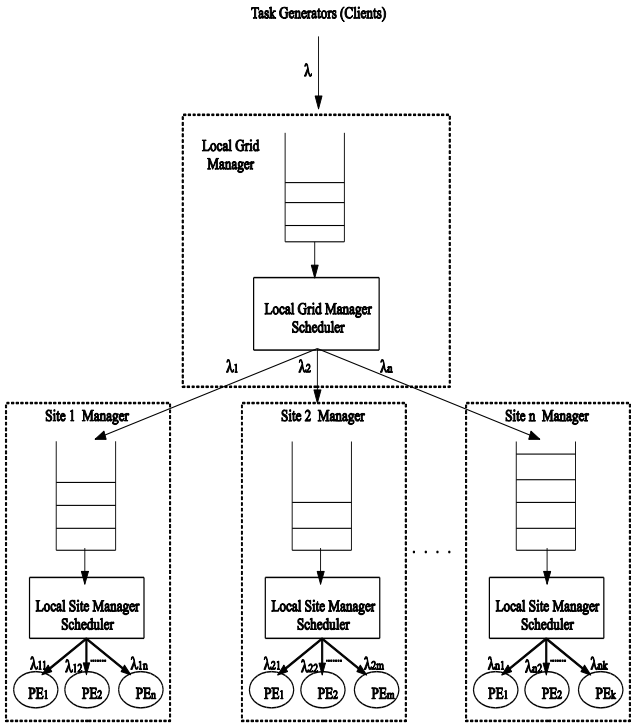


Figure 2. Grid Computing Queueing Model

The state transition diagram of the j[th] PE in i[th] site manager is shown in Fig. 3.

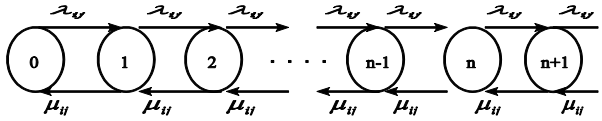


Figure 3. A state transition diagram of j[th] PE in i[th] site manager.

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., $\rho \prec 1$. 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 $E[N_g]$ 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, *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 $\rho_{ij} = \frac{\lambda_{ij}}{\mu_{ij}}$, $\mu_{ij}$ =$\mathrm{PEC}_{ij}$ 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 $\mathrm{PE}_{ij}$ 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 [25]. This simulator allows modeling and simulation of entities in grid computing systems users, applications, resources and resource load balancers for design and evaluation of LB algorithms.

To gauge the performance of grid computing system under the proposed LB 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 come to the LGM in a sequential fashion and their inter-arrival times are independent

and they follow the exponential distribution with mean 1/λ *j/s*. no Instantaneous arrivals is allowed. We also assume that the service times of LGMs follow the exponential distribution with mean 1/μ *j/s*.

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

In the Uniform distribution LB 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 LB 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 [26]. However, in the proposed LB 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 LB policy is computed analytically and by simulation as shown in Table 1. From that table, we can see that the response times obtained by the simulation approximate that obtained analytically. The obtained simulation results satisfy 95% confidence level.

Also, from table 1, we can notice that the proposed LB policy is asymptotically optimal because its saturation point (λ/μ)≈1 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 LB policy is compared with that of the Uniform distribution, and Random distribution as shown in Fig. 4. 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.

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 Fig. 5.

TABLE 1: COMPARISON BETWEEN ANALYTIC AND SIMULATION MEAN TASK RESPONSE TIMES USING THE PROPOSED LBP

| Arrival rate λ | Traffic Intensity ρ=λ/μ | Analytic Response Times | Simulation Response Times |
|---|---|---|---|
| 400 | 0.235294 | 0.009231 | 0.009431 |
| 500 | 0.294118 | 0.010000 | 0.010210 |
| 600 | 0.352941 | 0.010909 | 0.010709 |
| 700 | 0.411765 | 0.012000 | 0.012032 |
| 800 | 0.470588 | 0.013333 | 0.012833 |
| 900 | 0.529412 | 0.015000 | 0.015401 |
| 1000 | 0.588235 | 0.017143 | 0.017023 |
| 1100 | 0.647059 | 0.020000 | 0.019821 |
| 1200 | 0.705882 | 0.024000 | 0.024025 |
| 1300 | 0.764706 | 0.030000 | 0.029903 |
| 1400 | 0.823529 | 0.040000 | 0.040240 |
| 1500 | 0.882353 | 0.060000 | 0.058024 |
| 1600 | 0.941176 | 0.120000 | 0.119012 |
| 1650 | 0.970588 | 0.240000 | 0.238671 |
| 1660 | 0.976471 | 0.300000 | 0.297401 |
| 1670 | 0.982353 | 0.400000 | 0.401202 |
| 1680 | 0.988235 | 0.600000 | 0.610231 |
| 1685 | 0.991176 | 0.800000 | 0.798502 |
| 1690 | 0.994118 | 1.200000 | 1.201692 |

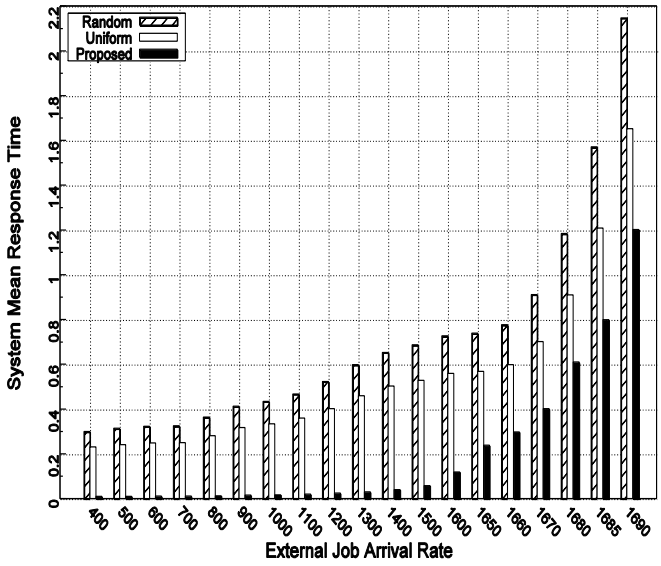


Figure 4. System mean job response time versus job arrival rate

From that figure, we can see that the improvement ratio increases as the system workload increases and it is about 72% in the range of parameter values examined. This result was anticipated since the proposed LBP balances the system's load according to the capacity of computing nodes 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.

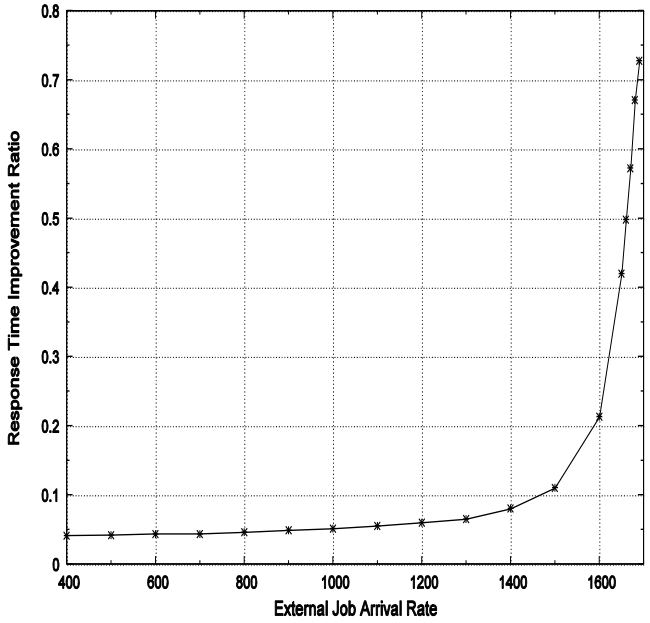


Figure 5. System mean job response time improvement ratio

## VII. CONCLUSION

This paper addresses the load balancing problem for computational grid environment. 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 strategy reflects the heterogeneity of the computing nodes. It balances system's load according to capacity of computing nodes. Consequently, the system's overall job response time, utilization are minimized and maximized respectively.

An analytical model is developed to compute the expected mean job response time in the grid system. To evaluate the performance of the proposed load balancing policy and validate the analytic results a simulation model is built using Arena simulator. The results show that the overall mean job response time obtained analytically is very close to that obtained by the simulation.

Also, the results showed 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 increases as the system workload increases and the maximum improvement ratio obtained is about 72% in the range of system parameter values examined.

## REFERENCES

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

[2] K. Lu, R. Subrata, and A. Y. Zomaya,"On The Performance-Driven Load Distribution For Heterogeneous Computational Grids," *Journal of Computer and System Science*, vol. 73, no. 8, pp. 1191-1206, 2007.

[3] S. Parsa and R. Entezari-Maleki," RASA: A New Task Scheduling Algorithm in Grid Environment," *World Applied Sciences Journal 7* (Special Issue of Computer & IT), pp. 152-160, 2009

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

[5] 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.

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

[7] S. F. El-Zoghdy, H. Kameda, and J. Li, "Numerical Studies on Paradoxes in Non-Cooperative Distributed Computer Systems," *Game Theory and Applications*, vol. 9, pp. 1-16, 2003.

[8] S. F. El-Zoghdy, H. Kameda, and J. Li, "Numerical Studies on a Paradox for Non-Cooperative Static Load Balancing in Distributed Computer Systems," *Computers and Operation Research*, vol. 33, pp. 345-355, 2006..

[9] 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.

[10] 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.

[11] A. N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," *J. ACM*, vol.32, no.2, pp.455-465, Apr 1985.

[12] 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.

[13] 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.

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

[15] 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.

[16] M. J. Zaki, W. Li, and S. Parthasarathy *"Customized dynamic load balancing for network of Workstations," In Proc. of the 5th IEEE Int. Symp. HDPC*: p. 282-291, 1996.

[17] A. Barak, O. La'adan, "The MOSIX multicomputer operating system for high performance cluster computing," *J. Future Gener. Comput. Systems*, vol. 13, no. (4-5), pp. 361–372, 1998.

[18] H.-U. Heiss, M. Schmitz, "Decentralized dynamic load balancing: The particles approach," *Inform. Sci.*, vol. 84, no. (1–2), pp. 115-128, 1995.

[19] M.H. Willebeek-LeMair, A.P. Reeves, "Strategies for dynamic load balancing on highly parallel computers," *IEEE Trans. Parallel Distrib. Systems*, vol. 4, no. 9, pp. 979–993, 1993.

[20] E. Saravanakumar and P. Gomathy," A novel load balancing algorithm for computational grid," *Int. J. of Computational Intelligence Techniques*, vol. 1, no. 1, 2010

[21] 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.

[22] Y. Wu, L. Liu, J. Mao, G. Yang, and W. Zheng, "Analytical Model for Performance Evaluation in a Computational Grid," *Proc. of the 3rd Asian Tech. Info. Program's (ATIP'S) on High performance computing*: solution approaches to impediment performance computing, pp. 145-151, 2007.

[23] 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.

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

[25] Arena simulator <http://www.ArenaSimulation.com>).

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

AUTHORS PROFILE

**Dr. Said Fathy El-Zoghdy** Was born in El-Menoufia, Egypt, in 1970. He received the BSc degree in pure Mathematics and Computer Sciences in 1993, and MSc degree for his work in computer science in 1997, all from the Faculty of Science, Menoufia, Shebin El-Koom, Egypt. In 2004, he received his Ph. D. in Computer Science from the Institute of Information Sciences and Electronics, University of Tsukuba, Japan. From 1994 to 1997, he was a demonstrator of computer science at the Faculty of Science, Menoufia University, Egypt. From December 1997 to March 2000, he was an assistant lecturer of computer science at the same place. From April 2000 to March 2004, he was a Ph. D. candidate at the Institute of Information Sciences and Electronics, University of Tsukuba, Japan., where he was conducting research on aspects of load balancing in distributed and parallel computer systems. From April 2004 to 2007, he worked as a lecturer of computer science, Faculty of Science, Menoufia University, Egypt. From 2007 until now, he is working as an assistant professor of computer science at the Faculty of Computers and Information Systems, Taif University, Kingdom of Saudi Arabia. His research interests are in load balancing in distributed/parallel systems, Grid computing, performance evaluation, network security and cryptography.