ISSN 2277-3061
Volume 11 Number 9
INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY

An Intelligent AntNet-Based Algorithm for Load Balancing in Grid Computing

S. F. El-Zoghdy[a, b]
[a]Department of Mathematics & Computer Science, College of Science, Menoufia University, Egypt.
[b]Department of Information Technology, College of Computers & Information Technology,
Taif University, Taif, KSA
elzoghdy@yahoo.com

ABSTRACT

Computational grids have a huge number of diverse and scattered resources that are used in handling complex problems. A decent load balancing methodology is needed to utilize grid resource by efficiently distributing tasks, for execution, on available computing nodes.

Ant colony is a major and popular method for approximate optimization. It works by simulating the actual ant's demeanor in detecting the best path for the resources of food. This research paper employs ant colony optimization in proposing a load balancing technique for computational grids. The performance of the suggested technique is computed, evaluated and compared with that of a Random Distributed Load Balancing technique using simulation. The achieved results reveal that the suggested technique enhances the task average response time. It reveals also that the enhancement ratio progressively rises up as the system's load rises up till the load come to be mild where the best enhancement ratio is achieved. Immediately after that, the enhancement ratio declines steadily as the system's load rises up till the system becomes saturated.

Keywords:

Computational Grids; Load Balancing; Ant Colony; Performance Optimization.

1. INTRODUCTION

Computational grid is an integrated environment of software and hardware that supports users by consistent, dependable, pervasive and cheap access to a huge set of computing resources. Such resources may include but not limited to computers, storage space, software applications and data [1]. These resources can be shared and coordinated by grid users without taking into account their type and location in the virtual organizations (VOs) to solve intensive computing tasks. VOs are composed of individuals, foundations and resources. In grid computing, a joint interface is utilized for linking LANs and clusters together. Any user or VOs can share the computing clusters. For reliability and authentications issues, each cluster applies a local security policy which identifies the access rights for every user. This policy is applied through a local resource management system. Fig.1. demonstrates the clustering process of grid resources.

Grid computing is mainly motivated by providing its users and applications with a widespread and smooth access to a huge set of advanced computing resources. To do this task, an illusion of a single system image is created. Consequently, such computational environments are implemented in a way so that their clients should not have to worry about where their tasks are executed [2-6,15].

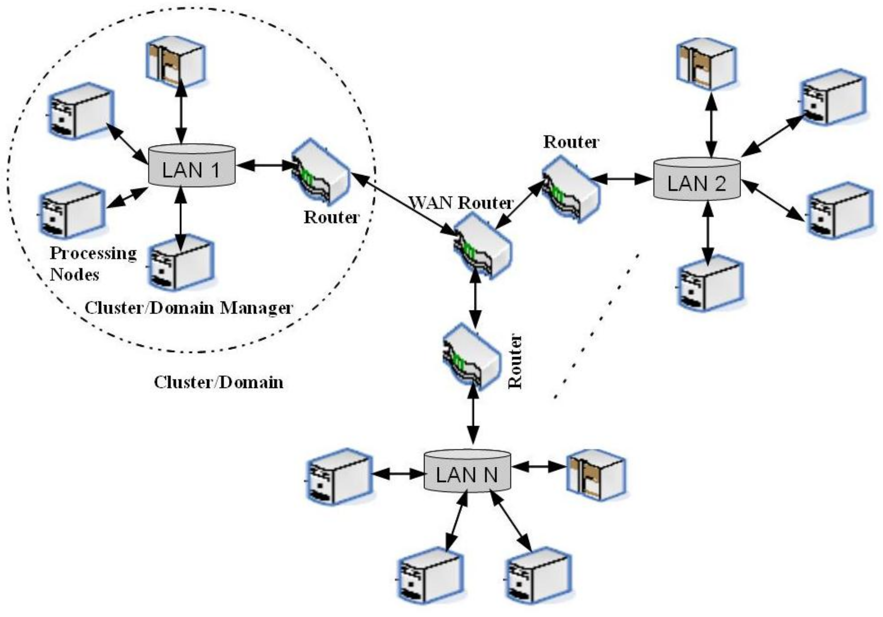

Fig. 1: Clustered distribution of Grid resources

Various services are offered by grid computing systems to their users like application, computation, information, knowledge and data services. Such services are conducted by the available servers and computing resources in grid system. The computing resources and servers are diverse by their nature as they have dissimilar storage space, memory sizes, CPU speeds and I/O bandwidths [2,3,7].

The diversity of the grid resources connected with the unequal task arrival forms could result in a situation that some computing resources in a cluster come to be over-loaded while others in another cluster are under-loaded or even idle. Consequently, there is a desire to shift some jobs from overloaded computing nodes to be processed on the under loaded ones targeting to improve utilization of existing resources. Redistributing system's workload is recognized as load balancing [8-13].

To efficiently utilize the huge set of available computing resources in grid, the grid infrastructure service level should utilize efficient load balancing and scheduling algorithms [1-8]. Such algorithms tend to minimize the task response time by maximizing utilization of existing resources. Their core objective is to avoid the possibility of having some burdened processing nodes and others under-loaded or idle at the same time [2-13].

Ant colony optimization (ACO) [16] is a major and recent technique among approximate optimization methodologies. The rousing origin of ACO techniques is the actual colonies of Ants. To be more precisely, ACO is originated mainly by the foraging demeanor of ants. The ant puts a definite quantity of pheromone during its walk. Ants tend to select a path positively correlated to the density of pheromone of found trials. Over time, the effect of pheromone trail vanishes. When several ants select a definite path and put their pheromones, this leads to increasing trail density. Therefore, such trail entices other ants; this manner leads to a highway of ants using shortest path. Ants are also able to dynamically adjust their behavior based on changes in the environment. For example, they are able to discover a new path when the old path is not valid anymore as a result of appearing a new barrier. In the essence of their demeanor, ants are able to communicate indirectly by using chemical pheromone trails which gives them the ability to discover the shortest paths between the ant's nest and a food source. Such fabulous features of actual ant's societies are utilized in ACO techniques for solving various scientific problems. Recently, ACO techniques are used for balancing workload of tasks in computational grids [18- 20].

This research utilizes ACO technology in developing a load balancing algorithm for computational grids. It considers the diversity of existing computing resource in grid. The proposed algorithm selects a resource to execute a task according to the assessed task transfer time and anticipated task processing time when it is allocated to such resource. It balances the grid workload using a local, and global pheromone update procedures. The local pheromone update process updates the status of the designated resource directly after assigning a task to it. On the other hand, the status of every resource is updated for all tasks directly after finishing any task by using global pheromone update process. This procedure gives grid scheduler the latest information about all resources to be utilized in the next task allocation round which lead to effectively utilizing the existing grid resources.

This policy leads to maximizing system utilization and improving load balancing level. Hence the mean job response time is minimized. A simulation model is built for assessing the performance of the suggested algorithm. The results reveal that the suggested technique enhances the average job response time compared to random distribution load balancing algorithm (RDLBA) in all studied cases. The enhancement ratio rises up steadily as the system traffic intensity rises up till the system load come to be moderate at this point the highest enhancement ratio is attained. After that, the enhancement ratio declines steadily as the grid load rises up reaching to the system saturation point.

The remainder of the paper is structured as follows: Related work is introduced in section II. Studied computational grid system is presented in section III. Section IV gives the suggested ant colony load balancing algorithm. Section V discusses the simulated model and explains results. At the end, section VI concludes this research paper and gives some of our future research directions.

2. RELATED WORK

Lately, the problems in all sciences become very difficult and complicated. They require enormous processing power and large storing space. The old systems like parallel or cluster computing ones are improper for solving such complicated problems. At the same time, the increasing popularity of the Internet connected with the availability of low-cost advanced computers and very high-speed networks altered the method we utilize computers systems today. These technological chances enable user from using scattered and multi-owner resources in solving various large-scale and complicated scientific problems. Latest research on these areas resulted in developing a new computing technology called grid computing [1].

Effectively utilizing the huge and diverse grid resources is a big challenge to grid designers and software implementers. To achieve this goal, the service level of the grid infrastructure should utilize efficient and effective load balancing and resource management algorithms [1-8]. These algorithms can be categorized into static and dynamic ones. For more information about such categorization and the features of each category, the reader is directed to [10-12].

A large number of load balancing algorithms for traditional distributed and parallel systems have been developed [8-13]. Unfortunately, the load balancing algorithms designed for traditional parallel and distributed systems which usually run on heterogeneous and dedicated resources cannot work directly in grid environments. Therefore, it is essential to consider the impact of various dynamic characteristics of grid in designing and analyzing load balancing algorithms [1-3].

Lately, a number of scholars have utilized ACO technology for studying load balancing problem in computational grids [18-21]. In [18], the authors explained the basic ideas of ACO and their applications in general. They gave some illustrative examples. In [19], the authors presented an ACO policy for computational grids. The scheduler in their policy assigns the task to the best match processing node selected form the existing processing nodes group. The authors of this study have performed a variety of exhaustive experiments using different simulation settings. Their results revealed that the suggested technique can certainly be applied practically and its performance is much better than that of other three earlier techniques. In [20], the authors introduced an ACO policy for balancing load in computational grids. Their algorithm utilizes the capacity of the existing resources in selecting the best processing node to execute a task and it balances the workload for all of the existing processing resources. The major goal for this algorithm is to enhance the system throughput and consequently the total grid performance will be improved. The authors in [21], developed a heuristic approach to obtain optimal solution for resource allocation problem in grid computing. They conducted many experiments using various data sets and settings. The attained results reveal that the performance of their technique is better than some of existing ant techniques. Also in [22], the authors introduced an ACO algorithm for load balancing in grid computing. Their main contributions are balancing the entire system load while trying to minimize the mean response time of a given set of jobs. Compared with the other job scheduling algorithms, according to the experimental results, the algorithm can outperform them. In [23], the authors presented a new security constraint model by formulating the scheduling problem for work-flow requests in the scattered and data-intensive systems. They introduced various meta-heuristic modifications to the main techniques of swarm optimization for treating effective schedules formulation and they introduced an adaptable neighborhood swarm optimization technique. The performance of their technique is computed and compared with that of multi-start genetic and multi-start swarm optimization techniques. The results reveal that their proposed meta-heuristic techniques always give analogous results for scheduling work-flow requests.

3. GRID COMPTING MODEL

The grid computing model considered in this paper is shown in Fig. 2. It has six main components: User, Portal, Grid Information Server (GIS), Domains, Grid Scheduler (GS) and Processing Nodes (PNs).

1. **User** is a person or program that submits jobs for execution to the grid.
2. **Portal** provides grid applications to grid users.
3. **GIS** is responsible for collecting grid information such as grid workload, network traffic,.. etc. periodically.
4. **Domain** is an independent object consisting of one or more computing nodes, and a Domain Manager (DM).
5. **GS** receives jobs, selects feasible domain for executing them based on the acquired information from the GIS and finally generates job-to-domain mappings according to the proposed load balancing algorithm.
6. **PNs** machines responsible for executing user jobs.

Every DM has unlimited storage capacity to hold all of the coming jobs from both exterior grid users and domain's local users. The processing nodes don't have capacity to hold any jobs (i.e., zero buffer capacity). They are only for execution. The dynamic nature and heterogeneity of the Grid resources makes the status information about available computing resources essential for GS in taking the scheduling decisions properly. The main function of GIS is to provide this information to GS. It collects the state information from all domains, such as entire domain processing capacities (equals summation of all CPUs capacities of processing nodes in the domain), network bandwidth, memory size, software accessibilities and burden of a domain in a certain period for every DM. Every DM is in charge of:

1. Supervising a dynamically changing group of computing nodes that is any member can join or leave the group at any time.
2. Recording newly joining computing nodes to its domain.
3. Gathering all needed information about active computing nodes in its domain and periodically updates GIS with such information. This information may include but not limited to computing node's processing capacities, available memory size, hardware specifications and existing software.
4. Taking in domain local scheduling decisions for all tasks submitted from domain's local users and external grid users to be executed in its domain.
5. Sending back the execution results to corresponding users.

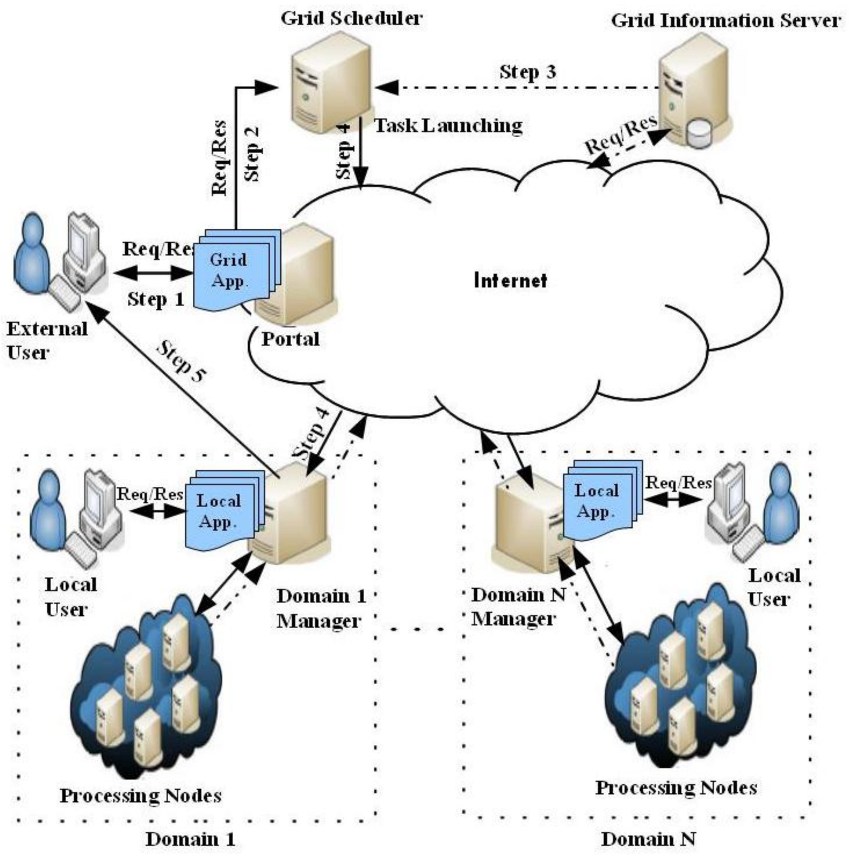

Fig. 2. Grid computing model architecture: dotted lines show information flow and solid lines show job or job scheduling command flow.

It is known that the most common factor affecting on the job response time in net computing is the communication time. For this reason the proposed load balancing algorithm concentrates on selecting the fastest available DM to execute the job based on the available state information about every domain. This information includes total domain processing capacities, available memory size, hardware specifications, network bandwidth, software availabilities and current load information of the domain. It is collected by the GIS periodically about every DM and is used to estimate job transfer time and expected processing time if it is allocated to that DM.

4. PROPOSED ANT COLONY LOAD BALANCING ALGORITHM (PACLBA)

The proposed ant colony load balancing algorithm (PACLBA) utilizes the main concepts of ACO techniques to minimize the response times of tasks in computational grids. This policy takes into consideration the current load information of each DM in taking load distribution decisions. In PACLBA the density of pheromone is updated based on the DM status information. The pheromone update process is conducted by executing a local and a global pheromone update functions. It aims to achieve minimum response time for every task by redistributing the workload in a way that efficiently utilizes all of the available grid resources. It is known that, the FCFS scheduling policy guarantees an assured fairness level, it does not need any information about task processing time in advance, its overhead is low and it can be implemented easily. Therefore, each DM utilizes such policy as a local scheduling one. With the FCFS policy, every DM in its local scheduling policy utilizes the fastest available processing node technique in case of having various free processing nodes at the time of selecting a processing node to execute a job.

In order to map the proposed ant colony model to the grid computing one, their relationships are explained as below:

1. **An ant**: Tasks in grid computing model are represented by ants in the ant colony model.
2. **Pheromone**: The weight of a DM in the grid computing model is represented by the value of the pheromone on a path in ant colony model. The GS computes the weight value for every DM using its corresponding data collected by GIS. Using this method, a DM having best computing power is realized by having the smallest weight value. The computed weight value for every DM is kept by GS and the GS employs it in the PACLBA as a major scheduling parameter in taking load balancing decisions.

Every job is assigned the initial weight (pheromone value) for each DM as a pheromone measure. This measure is computed by summing assessed transfer time and expected processing time of the job when it is allocated to a DM for execution. The expected transfer time is computed by $M_j/bandwidth_i$ where M_j is the j^{th} job size and $bandwidth_i$ is the bandwidth of available communication link between the GS and the i^{th} DM. The job processing time is hard to predict.

Depending on the type of programs many methods can be used in estimating the program processing time [27]. With that, the pheromone indicator is defined by:

Pheromone indicator:

$$Ph_{ij} = \left[\frac{M_j}{Bandwidth_i} + \frac{T_j}{CPU\_speed_i * (1 - Load_i)} \right] \qquad (1)$$

Where Ph_{ij} is the pheromone measure for the jth job which is assigned to the ith DM, M_j is the jth job size, T_j is the requested CPU time for processing jth job, $load_i$,CPU_speed_i and $bandwidth_i$ are the current status information of ith DM.

Based on equation (1), when a job is assigned to a DM, the DM status, the size of jobs and the anticipated program execution time are considered by the GS in the process of selecting the DM for execution. The smaller the value of Ph_{ij} is, the more efficient it is for the ith DM to execute job j. Assume there are m DMs and n jobs, hence the Pheromone (Ph) matrix is defined as follows:

$$Ph_{ij} = \begin{array}{c} \\ DM_1 \\ DM_2 \\ . \\ . \\ . \\ DM_m \end{array} \begin{array}{c} \begin{array}{cccccc} J_1 & J_2 & . & . & . & J_n \end{array} \\ \begin{bmatrix} Ph_{11} & Ph_{12} & . & . & . & Ph_{1n} \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ Ph_{m1} & Ph_{m2} & . & . & . & Ph_{mn} \end{bmatrix} \end{array} \qquad (2)$$

In each round, the smallest value in the Ph matrix is selected. Assuming Ph_{ij} is the selected value then jth job is assigned to the ith DM for processing there. After assigning a job to a DM, equation (1) is applied to that DM for each unallocated jobs in Ph matrix. This process is conducted to update local (row) pheromone. The whole Ph matrix entries are recomputed immediately after any job completion in a process called global pheromone update. After that, the row corresponding to the DM that just completes executing this job is further multiplied by $(1-\rho_i)$, where $0 \le \rho_i < 1$. ρ_i represents overhead incurred in the ith DM after finishing execution of jth job.

Performing global pheromone update reflects the changes of network condition and DM status after a job is completed. It incorporates the dynamic nature of the grid into the scheduling algorithm such that a better load balancing decision can be taken by the GS at the next turn. For n jobs and m DM, assigning the first job needs to calculate the Ph matrix of m×n entries. For the second job, only m×(n−1) entries remained in the Ph matrix ... etc. Hence, the total number of computed matrix entries equals $\mathbf{m} \times \sum_{\mathbf{i=1}}^{n} i = \frac{\mathbf{m \times n \times (n+1)}}{\mathbf{2}}$. This means that the proposed scheduling algorithm has good scalability even if n or m grows very large.

As it is illustrated above, the suggested technique considers the grid computing resources heterogeneity. It balances the grid load using the two pheromone update procedures explained earlier. The status of the designated DM is updated immediately after allocating jobs by the local update procedure. On the other hand, the global update procedure is used to update the status of all DMs with respect to all jobs immediately after a job completion takes place. It supports GS by the latest information about all DMs which in turn utilizes such information in taking balancing decisions for next task allocation round aiming to effectively utilizing the available grid resources. This policy leads to maximizing system utilization and improving load balancing level. Therefore, the task mean response time is improved.

The following example illustrates how a DM is selected to execute a job based on the pheromone level.

4.1 Example

Assume that the grid has five jobs (J1, J_2, J_3, J_4, and J_5) and five DMs (DM_1, DM_2, DM_3, DM_4 and DM_5). Also, assume that the sizes of the five jobs are 5MB, 15MB, 10MB, 4MB and 3MB respectively and that the initial status of every DM is as given in Table 1. The numbers of CPU iterations required for every job are 4M, 3M, 4.5M, 5M and 3.5M respectively.

Table 1: Initial status for every DM in the grid

	DM_1	DM_2	DM_3	DM_4	DM_5
Total CPU Speed (MHz)	1500	2000	2500	3000	3500
Load	0.25	0.1	0.3	0.2	0.35
Bandwidth (MB/s)	12.6	25.2	15.9	18.4	22.1

Applying (1), the initial pheromone level for every element in the pheromone matrix Ph is computed as follows:

$$Ph = \begin{array}{c} \\ DM_1 \\ DM_2 \\ DM_3 \\ DM_4 \\ DM_5 \end{array} \begin{array}{c} \begin{array}{ccccc} J_1 & J_2 & J_3 & J_4 & J_5 \end{array} \\ \begin{bmatrix} 0.400381 & 1.193143 & 0.797651 & 0.321905 & 0.241206 \\ 0.200635 & 0.596905 & 0.399325 & 0.161508 & \boxed{0.120992} \\ 0.316751 & 0.945111 & 0.631502 & 0.254429 & 0.190679 \\ 0.273406 & 0.816467 & 0.545353 & 0.219475 & 0.164502 \\ 0.228003 & 0.680052 & 0.454467 & 0.183193 & 0.137285 \end{bmatrix} \end{array}$$

When the job is dispatched, the GS determines the minimum pheromone level in the Ph matrix, that is Ph_{25}=0.120992. So J_5 is scheduled to DM_2 for execution. Hence, a local update (row update) to second row in the Ph matrix is performed for all jobs except J_5. Since J_5 is scheduled, column 5 in the Ph matrix is no longer needed. Now, assume that as a result of assigning J_5 to DM_2, DM_2 load becomes 22%. The new Ph matrix after executing local update is as follows:

$$Ph = \begin{array}{c} \\ DM_1 \\ DM_2 \\ DM_3 \\ DM_4 \\ DM_5 \end{array} \begin{array}{c} \begin{array}{cccc} J_1 & J_2 & J_3 & J_4 \end{array} \\ \begin{bmatrix} 0.400381 & 1.193143 & 0.797651 & 0.321905 \\ 0.200977 & 0.597161 & 0.39971 & 0.161935 \\ 0.316751 & 0.945111 & 0.631502 & 0.254429 \\ 0.273406 & 0.816467 & 0.545353 & 0.219475 \\ 0.228003 & 0.680052 & 0.454467 & 0.183193 \end{bmatrix} \end{array}$$

If the execution of J_5 is completed before the scheduler dispatches the next task, all elements of the Ph matrix will be updated by the global update process to estimate new values of pheromone indicators. These values are used in taking the new decision for allocating next task.

Suppose that, the DMs new status after the completing J_5 is as listed in Table 2, and the overhead incurred in DM_2 as a result of executing J_5 is 0.1 (i.e., ρ_2=0.1). Note that ρ_i=0 for all other DMs (i.e., $\forall i \neq 2$) because no jobs are allocated to them for processing yet.

Table 2: New system status after the execution of J_3

	DM_1	**DM_2**	**DM_3**	**DM_4**	**DM_5**
Total CPU Speed (MHz)	1500	2000	2500	3000	3500
Load	0.15	0.2	0.18	0.28	0.4
Bandwidth (MB/s)	10.5	20.2	12.9	15.4	19.1
Overhead (ρ)	0	0.1	0	0	0

The new Ph matrix after executing global update is as follows

$$\mathbf{Ph} = \begin{array}{c} \\ \mathbf{DM_1} \\ \mathbf{DM_2} \\ \mathbf{DM_3} \\ \mathbf{DM_4} \\ \mathbf{DM_5} \end{array} \begin{array}{c} \begin{array}{cccc} \mathbf{J_1} & \mathbf{J_2} & \mathbf{J_3} & \mathbf{J_4} \end{array} \\ \begin{bmatrix} \mathbf{0.477759} & \mathbf{1.429748} & \mathbf{0.954146} & \mathbf{0.382913} \\ \mathbf{0.224179} & \mathbf{0.669372} & \mathbf{0.447127} & \mathbf{0.179976} \\ \mathbf{0.389339} & \mathbf{1.164097} & \mathbf{0.777154} & \mathbf{0.312255} \\ \mathbf{0.326898} & \mathbf{0.975693} & \mathbf{0.651851} & \mathbf{0.262518} \\ \mathbf{0.263461} & \mathbf{0.786601} & \mathbf{0.525451} & \mathbf{0.211525} \end{bmatrix} \end{array}$$

This scheduling process is repeated for the other unassigned jobs.

4.2 Performance Metrics

Various performance metrics could be used in describing load balancing and grid resource management systems. In this study, the most commonly used three performance metrics in evaluating load balancing algorithm are utilized as follows:

1. Average Job Response Time

The response time of a job is defined as the interval of time between job arrival instant to grid and the job leaving instant from grid after finishing all of its computations and communications. Suppose r_i represents the i^{th} job response time.

Therefore, the mean job response time is computed by $RT = \frac{1}{N}\sum_{i=1}^{N} r_i$, given *N* as the entire number of jobs in the system.

2. Average node Utilization Rate

The utilization rate U_i of ith processing node P_i is obtained by dividing the completion time of task at Pi to the highest task completion time obtained from all computing nodes in the whole grid (Makespan), as follows:

$$U_i = \frac{P_i\,(\text{completion Time})}{\text{Makespan}} \times 100\% \qquad (3)$$

Hence, the average utilization rate U of all processing nodes is computed by:

$$U = \frac{\sum_{i=1}^{M} U_i}{M} \qquad (4)$$

Where M represents entire number of processing nodes in the system and **U** is in the range 0-1.

3. Load Balancing level

It is known that higher average resource utilization does not guarantee a good load balancing policy [24]. As a result of that, the mean square deviation *d* of processing nodes utilization rate U_i will be used as a measure of load balancing level. It is defined by:

$$d = \sqrt{\frac{\sum_{i=1}^{M} (U - U_i)^2}{M}} \qquad (5)$$

Based on equation (5), the lower the value of d is, the more efficient load balancing accomplished. Hence, the relative deviation α of *d* with respect to *U* which explains the level of grid load balancing is given by:

$$\alpha = \left(1 - \frac{d}{U}\right) \times 100\% \qquad (6)$$

The small values of the mean square deviation *d* lead to higher relative deviation which tells that the entire system workload is balanced between processing nodes (i.e. a good load balancing level). The best level of load balancing is attained in case of *d* equals zero which leads to α equals 100%.

The previously explained three performance metrics can be applied to the grid environment and they are correlated. For example, if the grid workload is balanced between the processing nodes, then the resource utilization rate will be high and consequently, the response time of tasks will be improved.

5. RESULTS AND DISCUSSION

5.1 Simulation Tool and Environment

Various simulation tools are available to simulate the proposed algorithm for balancing workload in grid computing systems. The reader is referred to [25] for more details. Among these simulation tools, GridSim v4.0 simulator [26] is utilized in our experiments because it is easily able to simulate various objects in grid computing systems through its offered facilities. These objects include users, heterogeneous resources, software applications, workload balancers of resource which are utilized in assessing performance of workload balancing methodologies. During the experiments, a heterogeneous grid model was constructed with diverse specifications for its resources to assess the performance of PACLBA. Gridlet objects are used to simulate tasks because it has all needed information associated with task and processing administration specifics. On the other hand, the Grid Information Service object has all of the requested information about the existed grid computing resources.

The simulation experimentations are conducted on a 3.6 GHz Core I3 Processor's PC having 8GB RAM and it is equipped by windows 7 OS.

5.2 Experimental Setup

The simulated grid environment contains 3 domains (sites) having 60 processing elements in total with different characteristics, configurations, and capabilities. Every domain has one job waiting queue. Domain local scheduling policy is M/M/n FCFS with fastest available processing node policy. That is, it selects the fastest PNs to execute a job in case of having many free PNs at the decision making time. The local and global bandwidths are 1000Mbps and 100Mbps respectively. All time units are in seconds.

The following assumptions are made for the simulations:

1. Tasks arrive sequentially and randomly to the system following a Poisson process with rate λ.
2. Times between arrivals are autonomous and follow the exponential distribution.
3. Instantaneous arriving of tasks is prohibited.
4. The task's processing times are assumed to follow the exponential distribution with mean μ.
5. Tasks are assumed to be mutually independent that is, there are no dependences or communication between them.
6. Any computing node can be used in executing tasks and every CPU can perform only one task at a certain point of time.
7. Tasks are not preemptable that is, the task processing could not be interrupted or shifted to any other computing node during its processing.
8. Task length is a uniformly distributed random number in the range of (0.1…0.5) Million Instructions (MI) unit.
9. Total CPUs speed ranging from 0 to 4 Million of Instructions per second (MIPs) are randomly assigned to the processing elements.
10. Every result listed in this paper is the mean value achieved from five simulation rounds starting with various seeds for generating random numbers.

Set ρ to represent the mean system traffic intensity parameter in the simulated mode. It is computed by dividing the mean arrival rate to the mean processing rate of tasks. Based on this definition, the tasks processing times μ are adjusted to obtain the requested traffic intensity ρ.

The job response time, mean node utilization and load balancing level are the three performance measurements used in evaluating the PACLBA. During the simulations, the average system traffic intensity factor is varied and results are collected to assess the performance of PACLBA under various system parameters setting. The final results of the simulations are presented on an average basis.

5.3 Experimental Results

This section presents an evaluation for the performance of the PACLBA and compares it with the performance of the Random Distribution Load Balancing Algorithm (RDLBA). In RDLBA the task processing domain is selected randomly. This performance comparison is conducted based on three performance measures: average job response time, average node utilization and load balancing level that indicates how much load balancing is achieved. In Fig. 3, the average job response time of the two algorithms is compared. From that figure, one can notice that average job response time of the two algorithms rises up as the system traffic intensity rises up. This is normal because increasing the traffic intensity means that there are many jobs need to be handled. One more point is that the PACLBA outperforms the RDLBA in all cases. This result was anticipated because the PACLBA selects a DM to execute a job according to the assessed task communication time and expected task processing time if it is allocated to that DM. Taking these parameters in consideration leads to effectively utilizing available resources which in turn minimize the grid mean job response time. On the other hand, the RDLBA selects randomly a DM to execute a job without taking into account any performance indicators and that lead to unbalance the distribution of system's load. As a direct result, the available grid resources are poorly utilized and consequently, the system performance is degenerated.

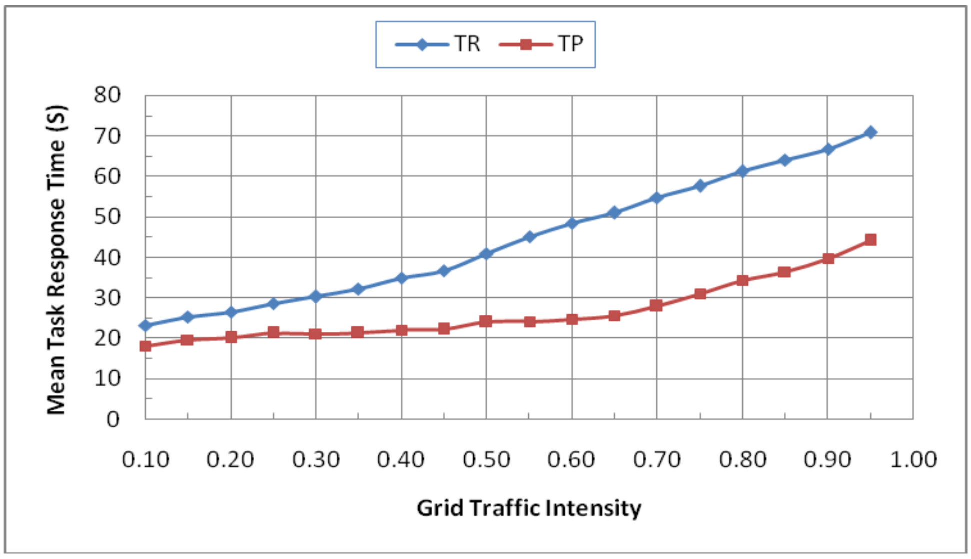

Fig. 3. Mean job response time of PACLBA vs RDLBA

To estimate the enhancement ratio achieved in task mean response time, we computed the mean task response time improvement ratio (TR-TP)/TR, where TR and TP are the mean task response time obtained using the RDLBA and PACLB algorithm respectively. Fig. 4 presents the improvement ratio in the mean job response time. From it, one can notice that the enhancement ratio rises up steadily as the load (traffic intensity) rises up. This increase continues till the system load come to be intermediate where the extreme enhancement ratio is achieved. After that the enhancement ratio declines steadily as the system load rises up till the system's saturation point reached.

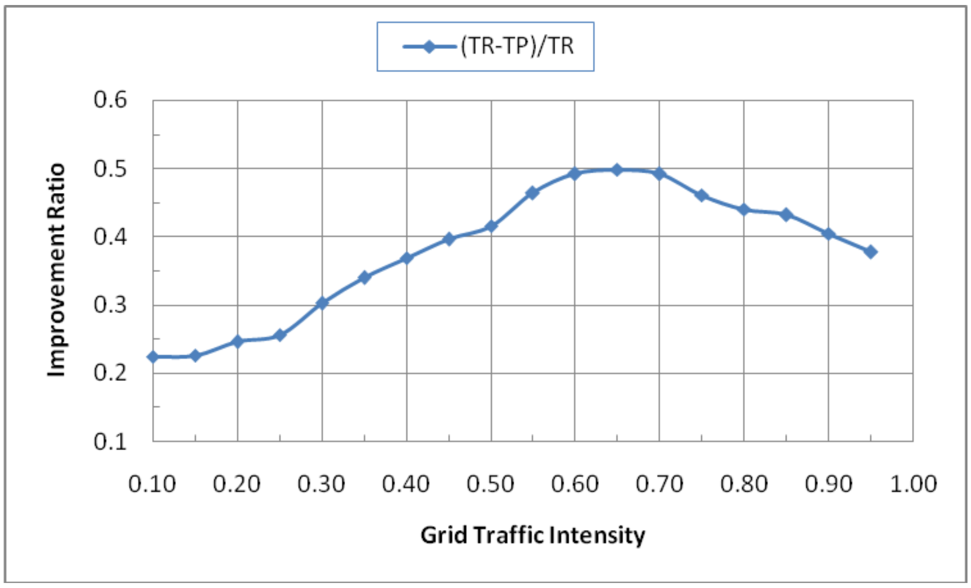

Fig. 4. Improvement ratio in mean job response time

Figs. 5 and 6 illustrate the mean utilization and mean square deviation of processing nodes for various grid workload using RDLBA and PACLBA respectively. From these figures, one can notice that the average processing nodes utilization (mean square deviation) obtained using the two algorithms increases (decreases) as the grid workload increases. However, the utilization (mean square deviation) of processing nodes under the PACLBA is always higher (lower) than that of the RDLBA which means that, the performance of the PACLBA is better than that of the RDLBA. Since, a low value of mean square deviation means a good load balancing level is obtained [24]. This ensures the results presented earlier in Figs. 3 and 4.

Fig. 7 presents the load balancing level for various grid workload using RDLBA and PACLBA. From that figure, it is noticed that the load balancing level obtained using the PACLBA is always higher than that of the RDLBA in all cases which again ensures the previously presented results. By carefully examining all of the presented results, we can say that the PACLBA performs more robustly than the RDLBA.

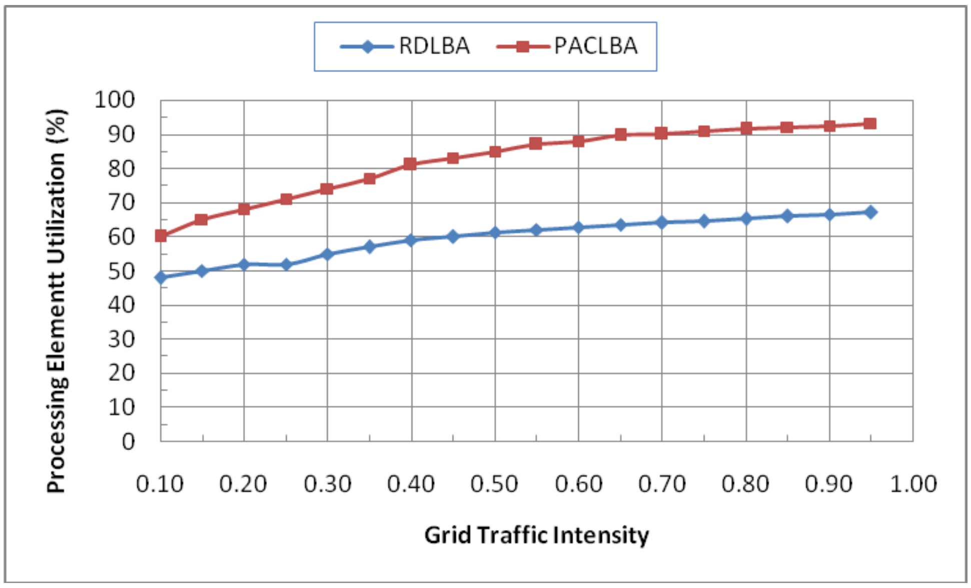

Fig. 5. Average Processing Node Utilization

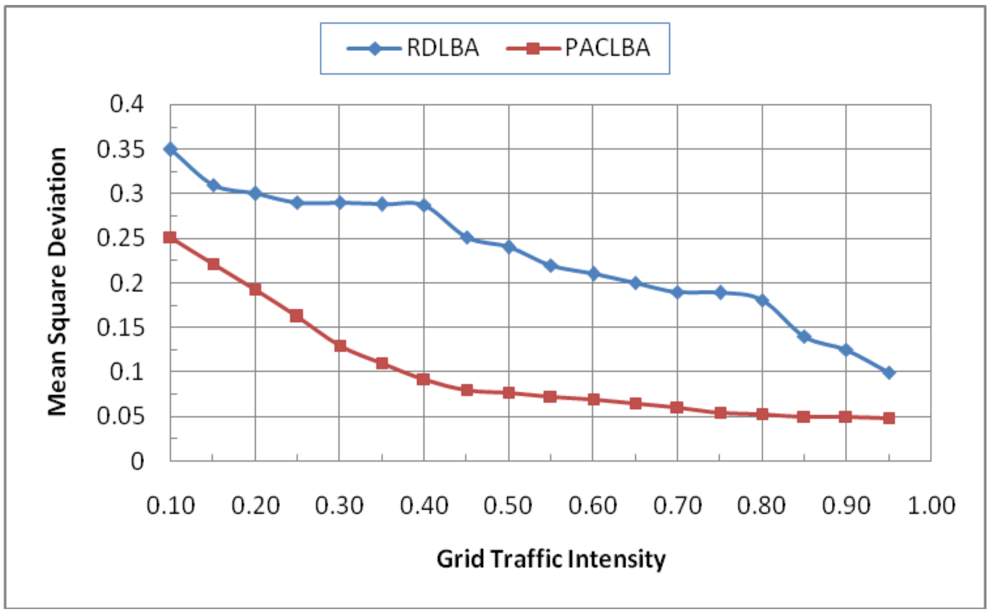

Fig. 6. Mean Square Deviation

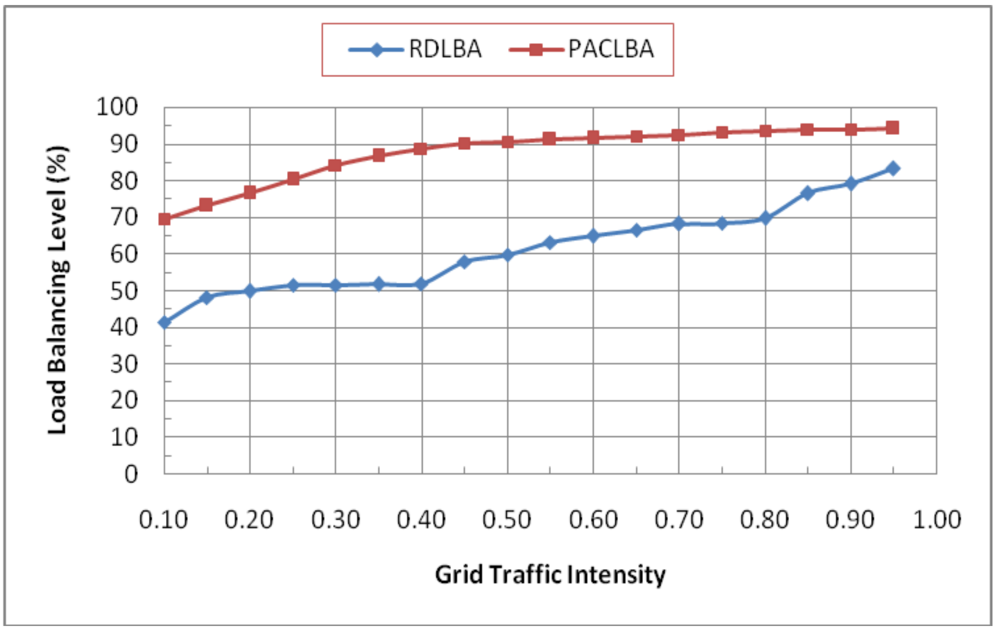

Fig. 7. Load Balancing Level

6. CONCLUSIONS AND FUTURE WORK

This paper presents an ant colony load balancing technique that selects a suitable domain manager for executing jobs in the computational grids infrastructure. The suggested algorithm takes into considerations the computing resources heterogeneity. It selects a domain manager to execute a job according to the assessed task transfer time and anticipated processing time of the task when it is allocated to that domain manager. The PACLBA balances the grid workload using a global and local pheromone update procedures.

To evaluate the performance of the PACLBA, a simulation model is built using GridSim simulator. The performance of proposed technique is compared with that of the RDLBA. The obtained results indicate that the PACLBA enhances the average task response time in all cases. The enhancement ratio rises up steadily as the system load rises up. Such increase continues till the system load come to be mild where the highest enhancement ratio is attained and then the enhancement ratio steadily degenerates as the system load rises up till the system's saturation point is reached.

In the future, we will study the reliability of PACLBA by studying some fault tolerance metrics. Also, the ability to extend the PACLBA to be able to deal with dependent tasks by adding a synchronization mechanism to it could be studied because the proposed algorithm deals only with independent tasks.

REFERENCES

1. B. Yagoubi and Y. Slimani, “Task load balancing strategy for grid computing “, Journal of Computer Science, 3(3), p. 186-194, 2007.
2. B. Yagoubi and Y. Slimani, “ Load balancing strategy for Grid Environment”, Journal of Information Technology and applications 1(4), p. 285-296, 2007.
3. B. Yagoubi and Y. Slimani, “Dynamic load balancing strategy for Grid computing”, World academy of science, Engineering technology, 19, p. 9095, 2006.
4. F. Berman, G. Fox and Y. Hey. Grid Computing: Making the Global Infrastructure a Reality. Wiley Series in Communications Networking & Distributed Systems, 2003.
5. R. Buyya, D. Abramson, J. Giddy and H. Stockinger. ‘Economic models for resource management and scheduling in Grid computing’, Journal of Concurrency and Computation: Practice and Experience, 14(13-15): pp.1507-1542, December 2002.
6. J. Cao, Daniel P. Spooner, Stephen A. Jarvis, S. Saini and Graham R. Nudd. Agent-Based Grid load Balancing Using Performance-Driven Task Scheduling.: Proceedings of the 17th International Symposium on Parallel and Distributed Processing, pages 49.2, 2003.
7. T. L. Casavant and J.G. Khul. ‘A taxonomy of scheduling in general purpose distributed computing systems’. IEEE Transactions on Soft. Engineering, 14(2):pp.141-153, 1994.
8. S. F. El-Zoghdy, Kameda, H. and Li J., A comparative study of static and dynamic individually optimal load balancing policies, Proc. of the IASTED International Conference on Networks, Parallel and Distributed Processing and Applications, p. 200-205, 2002.
9. S. F. El-Zoghdy, Hisao Kameda, and Jie Li, “A Performance Comparison of Dynamic Vs. Static Load Balancing Policies in a Mainframe-Personal Computer Network Model,” INFORMATION, Vol. 5, No. 4, pp. 431-446, October 2002
10. Tantawi A. N. and Towsley D. Optimal static load balancing in distributed computer systems. Journal of the Association for Computing Machinery, 32(2):445–465, April 1985.
11. Li J. and Kameda H. Load balancing problems for multiclass jobs in distributed/parallel computer systems. IEEE Trans. Computer, 47(3):322–332, 1998.
12. Kameda H., Li J., Kim C., and Zhang Y. Optimal Load Balancing in Distributed Computer Systems. Springer, Tokyo, 1997.
13. J. Zhang, J.H. He, and Y.Fu (Eds): “A Survey of load balancing in Grid computing”, CIS 2004, LNCS 3314, p. 280-285, 2004.
14. Ali Afzal, A. Stephen McGough, John Darlington, “Capacity planning and scheduling in Grid computing environments”, Future generation computer systems, 24, p. 404-414, 2008.
15. R. Sharma, V. Kant Soni, M. Kumar Mishra, P. Bhuyan, “A survey of job scheduling and resource management in Grid Computing”, World academy of science, Engineering technology, 64, p. 461-466, 2010.
16. Jagdish Chandra Patni, Dr. M.S.Aswal, Om Prakash Paland Ashish Gupta, “Load balancing Strategies for Grid Computing” presented at 3rd international conference on electronics computer technology (ICECT), vol.3, pp. 239-243, 2011.
17. Fangpeng Dong and Selim G. Akl, "Scheduling Algorithms for Grid Computing: State of the Art and Open Problems", Technical report, School of Computing, Queen’s University Kingston, Ontario January 2006
18. M. Dorigo and T. Stützle, Ant colony optimization, Cambridge,Massachusetts, London, England: MIT Press, 2004.
19. S. Fidanova and M. Durchova, "Ant algorithm for grid scheduling problem," Lecture Notes in Computer Science, vol. 3743, pp. 405- 412, 2006.
20. S. Suryadevera, J. Chourasia, S. Rathore and A. Jhummarwala", Load Balancing in Computational Grids Using Ant Colony Optimization Algorithm", International Journal of Computer & Communication Technology (IJCCT), 3(3), pp. 20-23, 2012
21. L. M. Nithya, and A. Shanmugam, " Scheduling in Computational Grid with a New Hybrid Ant Colony Optimization Algorithm", European Journal of Scientific Research,Vol.62 No.2, pp. 273-281, 2011.
22. Ruay-Shiung Chang, Jih-Sheng Chang, and Po-Sheng Lin," An ant algorithm for balanced job scheduling in grids", Future Generation Computer Systems 25, pp. 20–27, 2009.
23. Hongbo Liu, Ajith Abraham, Vclav Snšel, and Sen McLoone," Swarm scheduling approaches for work-flow applications with security constraints in distributed data-intensive computing environments", Information Sciences 192, pp. 228–243, 2012.

24. Yajun Li, Yuhang Yang, Maode M, Liang Zhou," A hybrid load balancing strategy of sequential tasks for grid computing environments", Future Generation Computer Systems , 25, pp. 819-828, 2009.

25. Y. ZHU, "A survey on grid scheduling systems", Technical report, Department of Computer Science, Hong Kong University of Science and Technology, 2003.

26. R. Buyya, "A grid simulation toolkit for resource modeling and application scheduling for parallel and distributed computing", www.buyya.com/gridsim/

27. Yuanyuan Zhang, Wei Sun, Yasushi Inoguchi, "Predict task running time in grid environments based on CPU load predictions", Future Generation Computer Systems 24 (6) (2008) 489–497.

Biography

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. He is currently the chairman of IT Dep. at the same place. His research interests are performance evaluation, load balancing in distributed/parallel computing systems, Grid computing, network security and cryptography.