A Multi-Class Task Scheduling Strategy for Heterogeneous Distributed Computing Systems

S. F. El-Zoghdy[1] and Ahmed Ghoneim[2,1]
[1] Department of Mathematics and Computer Science, Faculty of Science, Menoufia University, Egypt.
[e-mail: elzoghdy@yahoo.com]
[2] King Saud University, Department of Software Engineering,
College of Computer and Information Sciences
Riyadh 11543, Saudi Arabia
[e-mail: ghoneim@ksu.edu.sa]

Received November 13, 2014; accepted October 12, 2015; published January 31, 2016

Abstract

Performance improvement is a major issue in high performance distributed computing systems. In such computing systems, users can admit their jobs to a set of dynamic resources at anytime and wherever. Jobs arrival and processes execution times are stochastic. The performance of a distributed computing system could be enhanced by using a good load balancing strategy to redistribute the user tasks among computing resources for efficient utilization. This paper presents a multi-class load balancing strategy that balances different classes of user tasks on multiple heterogeneous computing nodes to minimize the per-class mean response time. For a wide range of system parameters, the performance suggested multi-class load balancing strategy is compared with that of the random distribution load balancing, and uniform distribution load balancing strategies using simulation. The results show that, the proposed strategy outperforms the other two studied strategies in terms of average task response time, and average computing nodes utilization.

Keywords: Resource Management; Load Balancing; Distributed Computing; Queuing Theory.

1. Introduction

Recently, many scientific problems come to be very difficult and complex. These problems require enormous processing power and packing space. Most of the previous used systems such as traditional parallel/ distributed computing ones are unable to solve such problems. On the other hand, the availability of cheep robust computers and fast networks are dramatically changing which leads nowadays to almost all computers are connected to the Internet. The connected computers can construct a workstations cluster or a distributed model. This technology advances enhanced the opportunity of utilizing physically scattered and multi-owner computers in solving complicated problems in many fields such as science, engineering, and commerce.

As a result, new computing paradigms such as Cluster, Grid, and Cloud computing have been emerged [1-3, 14]. These newly emerged computing environments are referred to as High Performance Distributed Computing Systems (HPDCS). These Computing environments provide reliable, harmonic, pervasive, and cheep access to powerful computing resources including packing store, data, computers, and softwares. They enable users to share and organize their utilization of resources autonomously of their location and type in dynamic virtual organizations (VOs) comprising of organizations, individuals, and resources to solve computationally expensive applications. Also, these computing systems employ public interface in connecting clusters or LANs with each other.

Many users or VOs can share these clusters and through a local resource management system, each cluster or LAN applies a local policy that defines their access rights. The primary motivation of these newly emerged computing systems is to support clients and applications by widespread and smooth access to HPDCS resources by making an illusion image of a single system. Therefore, HPDCS are constructed so that clients do not have to worry about place where their tasks will be processed [1-3,5-8,11,13-19].

The servers or processing nodes in the HPDCS offer various services including data, knowledge, computation, information, and application services. They usually are heterogeneous as they have various processing speeds, memory sizes, and I/O bandwidths [1-3,7,14-19,21-29].

In such computing systems, users admit their jobs anywhere and anytime to a group of dynamically changing resources. Arrival and execution times of processes are random. The available resources heterogeneity, the irregular job arriving and stochastic distribution of jobs execution times could lead to a case where some nodes come to be overloaded while others might be under-loaded or idle.

It is consequently necessary to move some jobs from over-loaded computers to idle or under-loaded ones hoping to effectively use computing resources. Redistributing system's workload among available processing nodes is recognized as load balancing. To satisfy the objectives of enormous distributed computing resources, efficient and effective load balancing and resource management algorithms should be utilized [1-10].

Load balancing methodologies could be categorized into static or dynamic [4,8,9,40]. Static load balancing techniques decide probabilistically or deterministically their choices and keep them during runtime. Therefore, such techniques could not be affected by the current system state information. On the other hand, dynamic load balancing techniques utilize current system state information in taking their decisions, see [4,8,9,10,11,40] for more details.

Various load balancing methodologies for traditional parallel and distributed systems have been suggested [4, 9,12,20,40]. Unfortunately, these load balancing algorithms cannot work directly in recent distributed computing environments such as Cluster, Grid, and Cloud computing systems. Consequently, it is necessary to put in mind the effect of several dynamic features for HPDCS in planning and implementing load balancing techniques [1-3,14,29] for such computing systems.

Lately, scholars published various techniques to deal with the load balancing problem in HPDCS [14,15-18, 41]. Almost all of these papers considered jobs as one class. They ignored the fact that distributed system's jobs varies widely by their nature. These jobs range from batch processes and CPU-bounded jobs to real time and even traditional jobs. For each job it is required to satisfy the associated performance and quality of service constraints. Hence, it is therefore important to classify these jobs into different classes aiming to utilize the system resources effectively and at the same time satisfies the users demands. Only a little number of researchers dealt with the multi-class of jobs load balancing problem in HPDCS such as [12, 19, 20].

In this paper, we propose a load balancing strategy for multiclass jobs that could be used in the HPDCS. The suggested strategy considers the heterogeneity of system's computing resources as well as the task heterogeneity. The performance of the proposed load balancing strategy is evaluated and compared with two other load balancing strategies by simulation. The obtained results demonstrate that the performance of the suggested load balancing strategy overtakes the other studied strategies in terms of average job response time, and average computing node utilization.

The remainder of this paper is structured as follows: Section II introduces related work. Section III explains the studied model and its assumptions. Section IV presents the suggested multi-class load balancing strategy. Section V gives the obtained results and their discussion. Lastly, Section VI concludes the paper.

2. Related Work

HPDCS consist of a dynamic set of heterogeneous resources communicating via one or more communication networks. Users of these systems submit their tasks at anytime and anywhere. One of the most important problems in the HPDCS is balancing the system workloads among computing nodes targeting to effectively use accessible resources and therefore enhance system performance [14,19,29]. A large number of researchers studied the load balancing problem in the traditional parallel and distributed systems. They developed a number of load balancing algorithms such as [4,9,10, 20,37,40]. Unfortunately, these algorithms cannot work directly in the recent HPDCS because they do not consider the dynamic characteristics and the heterogeneity of these systems in designing and analyzing the load balancing algorithms. Some of these load balancing algorithms have been modified to be able to deal with the HPDCS, and other completely new load balancing algorithms for such computing systems have been developed.

In [17], the authors proposed a simple and efficient load balancing algorithm that balances the system workloads between computing clusters that are far apart from each other. In [29,39], the authors presented a general review of job distribution and resource controlling strategies in grid computing environment. Also in [30, 31], the authors introduced a static policy for a heterogeneous system having servers and computing nodes. The servers distribute system’s workload over computing nodes using round robin policy. Their algorithm requests every server to knew workload information status for all computing nodes and the workload assigned to them through other servers. The authors in [32] introduced a load balancing strategy for a heterogeneous distributed computing system where the scheduler choices the computing resources according to the task characteristics, task requirements, and resource's information. The aim of their strategy is to reduce the Total Time to Release (TTR) for every task. TTR is composed of task processing time, task in queue waiting time, data input and output transfer time to and from resource node. The algorithms presented in [33] gives a job migration load balancing strategy that balances the system's load when any of the computing nodes becomes over-loaded. This algorithm does not consider the heterogeneity of system resources and/or network.

In [34], the authors introduced a grid computing managers that are based on ring topology. The Grid managers are in charge of the management of a dynamic set of computing nodes. They proposed a load balancing technique using real computing nodes workload information. Unfortunately, such technique can not be applied due to its enormous overhead cost in collecting the computing nodes workload information specially for large scale systems. The authors in [1,3,22,25,34-36] introduced many hierarchical load balancing algorithms for heterogeneous distributed computing systems. These algorithms are implemented at various levels to reduce the communication overhead and hence minimize the mean response time of the system's application.

In [22], load balancing strategies are proposed for a two-level grid computing architecture. The authors evaluated the performance of such strategies at both of global and local schedulers levels. The strategies presented in [11,22] did not have any task assignment rule and also communication overhead between clusters is not taken in consideration. Their load balancing strategy collects the computing nodes workload information periodically by a central master node called Grid Manager. This is a centralized policy and it suffers from the central point of failure problem as the failure of the Grid Manager leads to the whole system failure.

In [3] a two-level load balancing strategy is presented. It considers the distributed system computing resources heterogeneity. This strategy balances workload according to the processing power of the computing nodes. It overcomes the bottleneck of the strategies presented in [11,22,25] by removing Grid Manager that manages the global system's load information. In [35], the same authors of [34] presented Grid managers hierarchical structure which improves scalability of the grid computing environment. Their load balancing strategy has task allocation rule which automatically controls task's flow rate sent to a certain grid manager. In [15], the authors presented a two levels task scheduling strategy which balances the workload in cloud computing. It takes into account the new features of cloud computing, such as virtualization and flexibility etc. In [16,17], comparative studies between some of the existing cloud computing load balancing algorithms are conducted.

All the previously discussed strategies deal with the user tasks as one class which is against the reality as the tasks differ in many aspects by nature. Some tasks may be I/O-Bounded, CPU-Bounded, and others may be interactive. Hence, it is not fair to deal with all tasks as one class. Only countable number of researchers deals with users tasks as multi-classes. In [12] the authors presented a new indicators to qualify system heterogeneity and accessibility for multi-class user jobs. In [20,40], the authors studied the scheduling problem of different classes of users jobs in heterogeneous distributed computing environments. They proposed optimal static load balancing policies. Finally, the authors in [19] proposed two dynamic load balancing schemes for multi-user jobs in heterogeneous distributed systems and compared their performance using simulation. The objectives of the two polices are different. The first one minimizes the whole system average response time, while the other minimizes the average task response time individually. Their performance is compared with another two static load balancing strategies. As usual the dynamic strategies outperforms the static ones in terms of response time for low and moderate communication overhead, and they have the same performance as the static ones if the system communication overhead is high.

In this paper, we propose a multi-class load balancing strategy that balances different classes of non-cooperative user’s tasks on multiple heterogeneous distributed computing nodes. The main goal of this strategy is to minimize the per-class average response time. The scheduler consists of two modules. The first is the availability detecting model that is used to find all computing nodes in the system that satisfy all the requirements of every submitted job from all classes, and group these candidate computing nodes. The second is the load balancing model. It receives the set of candidate computing nodes from the availability detecting model and selects from them the one that gives the minimal response time to execute the user job. The proposed multi-class load balancing strategy considers the computing nodes heterogeneity. It allocates the system's load among all the available computing nodes by evenly distributing the service utilization aiming to minimize the response time.

The system average job response time, and the average computing node utilization are considered as the main performance metrics that need to be minimized and maximized respectively. The computing nodes utilize a local priority scheduling policy that gives higher priority to job classes with higher service rates at that computing node.

The proposed strategy provides the following unique characteristics of practical distributed computing environment:

- Heterogeneity: The heterogeneity of system's computing nodes is considered in the proposed strategy.
- Tasks are non-preemptable: The execution of the tasks on any computing node could not be suspended till its completion.
- Tasks are totally autonomous: No inter-process communication among tasks.
- Tasks are computationally expensive: Tasks takes more time performing computations in the CPU than I/O operations.

3. Proposed Framework

In this paper, we consider a heterogeneous distributed computing system model in which a set of n Computing Nodes (CNs) are connected via a high speed network as illustrated in **Fig. 1**. In this network, nodes are numbered 1, 2,...,n, and the system is utilized to execute m autonomous classes of non-cooperative processes admitted by users. Each computing node in the system is composed of a one exponential server having a service rate μ_i (i = 1, 2, ...,n), and its service policy is FCFS, or processor sharing where every job service rate equals μ_i / n if the ith server queue has n jobs.

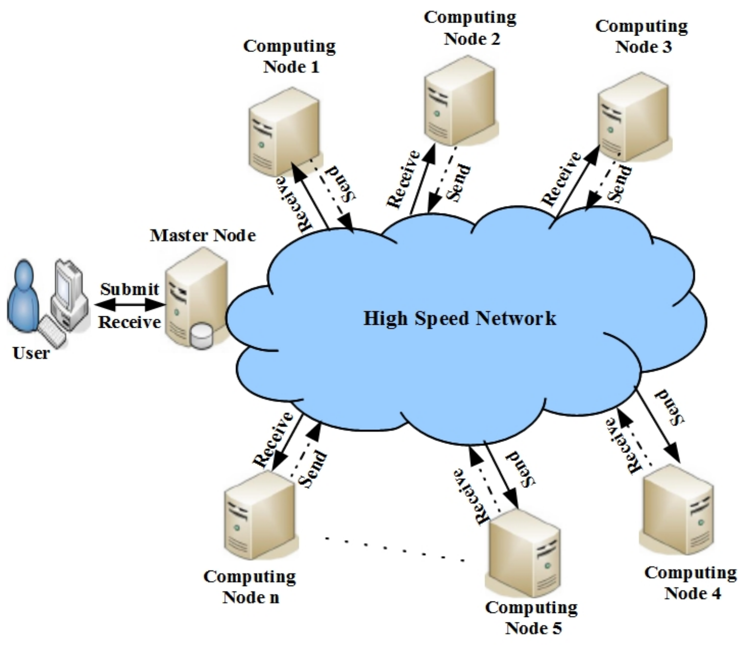

Fig. 1. System architecture

Computing nodes in the system differ in their speeds, memory, and disk space. A master node or a load manager is responsible for balancing system's workload and monitoring available system resources. **Fig. 2** shows the system queueing model. It consists of a task scheduling queue, task scheduler, and n local task queues. The scheduler queue is a temporarily buffer that is large enough to hold all incoming tasks. The FCFS strategy is utilized for scheduling waiting tasks in schedule queue, and local queues. It guarantees fairness, does not need task's execution time information in advance, does not need huge computing power, and its implementation is simple. The scheduler is composed of two modules namely: Availability Detecting Model (ADM), and Load Balancing Model (LBM). For every task in jth class, the ADM finds all CNs in the systems which satisfy the task requirements, and group these CNs in the set A_j. The ADM then passes the set A_j to the LBM to select the computing node from A_j that offers the minimal anticipated response time for executing the task as a candidate computing node. Users can submit tasks form any of the m classes to the system.

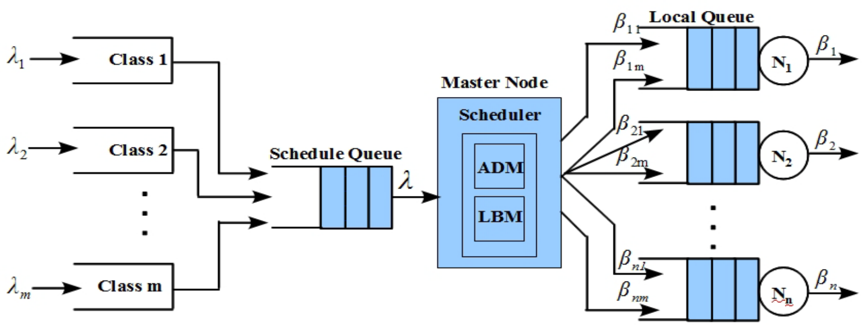

Fig. 2. System queuing model

We assume that system's admitted tasks are completely autonomous having no inter-process communication among them, and those tasks are computationally expensive ones. Also, we assume that jobs of the j th ($1 \leq j \leq m$) class reach to the system according to ergodic process, like Poisson process, having identically, independent distributed inter-arrive times with rate λ_k. Synchronized arrivals are not allowed.
For upcoming reference, the used notations through this research are summarized in the following table:

Table 1. Notation Definitions

Notation	Definition
n	System's computing nodes number $(1 \leq n \leq \infty)$
m	Task's classes number admitted to the system $(1 \leq m \leq \infty)$
λ_j	j th class task arrival rate to the system
λ	Total system arrival rate from all classes (See Eq. 1).
P_{ij}	The probability in which tasks from the j th class are forwarded to the i th computing node.
β_{ij}	Processing rate (load) at i th node from j th class tasks. (See Eq. 2)
β_i	Total processing rate (load) of computing node i. (See Eq. 3)
β	System's total task processing rate (load) for all classes. (See Eq. 4)
μ_{ij}	Allocated service rate at ith node for jth class task's.
μ_i	Total service rate of computing node i (See Eq. 5).
μ	Total system service rate (See Eq. 6).
ρ_{ij}	j th class service utilization at the ith node (See Eq. 7)
ρ_i	ith computing node service utilization (See Eq. 8)
ρ	System service utilization (See Eq. 9).
ρc_j	The system j th class service utilization (See Eq. 10)
TN_i	The i th computing node's mean response time over all classes (See Eq. 11).
TC_j	System expected mean task response time of jth class tasks (See Eq. 12).
T	The system's mean response time (See Eq. 13).

The system total task arrival rate from all classes is denoted by λ and λ_j is the j th class task arrival rate to the system. Hence

$$\lambda = \sum_{j=1}^{m} \lambda_j \tag{1}$$

Denote β_{ij} as the i th computing node processing rate of the j th class tasks, which is also referred to as i th computing node load from the j th class.
Let P_{ij} be the probability that i th computing node receives tasks from the j th class, where i=1,2,…,n, and j=1,2,…,m.

Hence, the ith computing node workload from the jth class is computed by:

$$\beta_{ij} = P_{ij}\lambda_j, \quad i=1,2,?n, \quad \text{and} \quad j=1,2,?m \quad . \tag{2}$$

Hence, the total workload of the ith computing node i from all classes can be expressed as

$$\beta_i = \sum_{j=1}^{m} \beta_{ij} = \sum_{j=1}^{m} P_{ij}\lambda_j, \quad i=1,2,?n \quad . \tag{3}$$

As a result, the system's total workload from all classes, β, can be computed as follows:

$$\beta = \sum_{i=1}^{n} \beta_i = \sum_{i=1}^{n}\sum_{j=1}^{m} \beta_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{m} P_{ij}\lambda_j \tag{4}$$

Denote μ_{ij} as the allocated service rate at ith node for jth class task. Hence the corresponding expected service time is computed by $\frac{1}{\mu_{ij}}$. As mentioned earlier, the service times of every computing node has exponential distribution which is independent from the arrival process. Hence the ith computing node service rate can be computed by:

$$\mu_i = \sum_{j=1}^{m} \mu_{ij}, \quad i=1,2,...,n. \tag{5}$$

Consequently, the total system service rate is computed by:

$$\mu = \sum_{i=1}^{n} \mu_i = \sum_{i=1}^{n}\sum_{j=1}^{m} \mu_{ij} \tag{6}$$

Denote ρ_{ij} as the jth class service utilization (traffic intensity) at the ith computing node. It is computed by:

$$\rho_{ij} = \frac{\beta_{ij}}{\mu_{ij}} = \frac{P_{ij}\lambda_j}{\mu_{ij}}, \; i=1,2,...,n, \text{and } j=1,2,...,m. \tag{7}$$

Hence, the service utilization of all jobs assigned to ith computing node is computed by:

$$\rho_i = \sum_{j=1}^{m} \rho_{ij} = \sum_{j=1}^{m} \frac{\beta_{ij}}{\mu_{ij}} = \frac{\beta_i}{\mu_i}, \; i=1,2,...,n. \tag{8}$$

Consequently, the total system service utilization can be computed by:

$$\rho = \sum_{i=1}^{n} \rho_i = \sum_{i=1}^{n}\sum_{j=1}^{m} \rho_{ij} = \sum_{i=1}^{n}\sum_{j=1}^{m} \frac{\beta_{ij}}{\mu_{ij}} = \sum_{i=1}^{n} \frac{\beta_i}{\mu_i} = \frac{\beta}{\mu} \tag{9}$$

Finally, the total system service utilization of the jth class ρc_j can be computed as follows:

$$\rho c_j = \sum_{i=1}^{n} \rho_{ij} = \sum_{i=1}^{n} \frac{\beta_{ij}}{\mu_{ij}} = \frac{\lambda_j}{\sum_{i=1}^{n} \mu_{ij}}, \ j = 1,2,..,m \tag{10}$$

The stability condition for the studied system model is $\lambda < \mu$, see [38] for more details.
In this research, each node is demonstrated as a single M/M/1 model. Thus the mean response time of the ith computing node could be calculated as follows:

$$TN_i = \frac{(1/\mu_i)}{(1-\rho_i)}, \ i = 1,2,\ldots,n. \tag{11}$$

Hence, the system expected mean response time for the jth class tasks can be obtained as follows:

$$TC_j = \sum_{i=1}^{n} P_{ij} TN_i, \ j = 1,2,\ldots,m. \tag{12}$$

Form Eq. 12, the average overall system mean response time overall classes can be computed by:

$$T = \sum_{j=1}^{m} \left(\frac{\lambda_j}{\lambda} TC_j \right) = \sum_{j=1}^{m} \left(\frac{\lambda_j}{\lambda} \sum_{i=1}^{n} P_{ij} TN_i \right) \tag{13}$$

From Eq. 13, one can notice that the overall system mean task response time could be minimized by minimizing average response time of jth class. Hence, the scheduling problem can be expressed as follows:

$$\text{Minimize} \quad TC_j = \sum_{i=1}^{n} P_{ij} TN_i, \ j = 1,2,\ldots,m.$$

subject to

$$\sum_{i=1}^{n} \beta_{ij} = \lambda_j, \quad j = 1,2,\ldots,m,$$

$$\beta_{ij} \geq 0, \qquad i = 1,2,\ldots,n, \text{ and } \ j = 1,2,\ldots,m.$$

4. Proposed Multi-Class Load Balancing Strategy

In this section, the strategy of scheduling different classes of tasks on multiple distributed heterogeneous CNs (servers) is presented. The proposed strategy considers the system's computational resources heterogeneity. It balances the system's load among CNs by evenly distributing the service utilization aiming to minimize the average response time. In other words, the system's workload is perfectly balanced between CNs by making all the CNs service utilization equal. This strategy minimizes the per-class mean response times. It has two distinct decisions:

- The tasks allocation to the CNs; and
- The tasks execution order at each computing node.

The first decision is considered as an overall load-balancing problem where tasks are balanced among multiple heterogeneous CNs to minimize the mean response-time of each class. The second is a local decision at each computing node and consists of solving an optimal sequencing problem: given a mix of tasks at a computing node, find the best service order of queued tasks to minimize the mean class response time. The proposed scheduling strategy puts the following restrictions on these two decisions:

- Allocation: tasks are allocated to CNs immediately upon arrival in a probabilistic manner; i.e., a task is assigned to a CN based on routing probabilities $\{P_{ij}\}_{1\le i\le n, 1\le j\le m}$. the values of the routing probabilities are determined using the scheduling strategy algorithm which minimizes the mean class task response time. This algorithm will be explained later.
- Sequencing: The sequencing strategy is the same at each CN and it is a simple static priority policy. It does not require knowledge of the future, and the execution of a task cannot be interrupted and subsequently resumed (i.e., non-preemptive).

The proposed scheduling strategy utilizes an existing optimal sequencing strategy [20] which minimizes the mean response time of all classes. It is designed according to proposition 2.1 in [20].

Proposition 1. Given an n-node heterogeneous system, and an m-class M/M/1 queue, class j having arrival rate λ_j and service rate μ_{ij} at node i. The sequencing strategy on node i which sets priority to class j over class k when $\mu_{ij} \ge \mu_{ik}$ minimizes the system overall mean response time.
This proposition can be proved directly from proposition 2.1 in [20].
According to proposition 1, the scheduling process gives high priority to classes having higher service utilization rates. Using this sequencing strategy, the task classes are reordered and categorized such that $\rho c_1 \ge \rho c_2 \ge \ldots \ge \rho c_m$

Where, $\dfrac{\lambda_j}{\sum_{i=1}^{n} \mu_{ij}}$, (j=1,2,…,m) is the system jth class service utilization (See Eq. 10).
This categorization process allocates a number of priority stages to classes of tasks. As a result of this sequencing strategy the jth class tasks are assigned priorities higher than the kth class tasks if j<k.
For the allocation strategy, the master node (load manager) allocates arriving tasks to CNs immediately upon arrival in a probabilistic manner; i.e., a task is assigned to a CN based on routing probability $\{P_{ij}\}_{1\le i\le n, 1\le j\le m}$. The proposed allocation algorithm determines the

probability that minimizes the response time of all classes. This algorithm selects the CN that minimizes the mean class task response time, and at the same time, it should not be overloaded for computing the task.
To satisfy this, the proposed algorithm uses the following service utilization measure L_i to detect the relative workload of each computing node i:

$$L_i = \frac{\rho_i}{\left(\frac{\rho}{n}\right)}, \ i = 1,2,\ldots,n. \tag{14}$$

Where ρ_i is the ith computing node service utilization given by Eq. (8), and (ρ/n) is the system's average computing node utilization given by Eq. (9). The proposed scheduling strategy aims to keep L_i very close to 1 which means that the system's CNs service utilization is evenly distributed. I_n the following lines, we list the details of the proposed allocation algorithm:
It is assumed that admitted tasks to the system are absolutely autonomous. This means that the amount of inter-process communication among tasks is zero. It is also assumed that tasks are computationally expensive ones. The FCFS strategy is utilized for in queues tasks. It guarantees definite fairness, does not require any task execution time information in advance, does not need huge computing power, and its implementation is easy [2,3].

4.1 Multi-Class Load Balancing Strategy

```
1. Reorder and label the task classes such that $\rho c_1 \geq \rho c_2 \geq \ldots \geq \rho c_m$ , where,
   $\rho c_j = \frac{\lambda_j}{\sum_{i=1}^{n} \mu_{ij}}$ , ( j=1,2,…,m) is the system jth class service utilization.
2. For each class j do
      a. Set $TC = \infty$; /* as initial value for the class response time */.
      b. Set $N_{min}$=1, and $L_{min}=\infty$;/* Assume that the lightest CN is node 1 and its
         relative workload is $\infty$ */
      c. Create a set of CNs $A_j$ that satisfy the jth class tasks software and hardware
         requirements $(A_j \subseteq n)$.
      d. For each computing node i in $A_j$ do
            i. Set $P_{ij}$=1;/* assume that the task will be allocated to that computing
               node, and calculate the expected response time for jth class $TC_j$ using
               Eq. 12 */
           ii. Calculate the relative workload $L_i$ for that node using Eq. 14.
          iii. if (($TC_j$<TC) && ($L_i$<$L_{min}$)) then
                  1. TC=$TC_j$;
                  2. $L_{min}$=$L_i$;
                  3. $N_{min}$=i;
           iv. End if
      e. End for
      f. $P_{N_{\min} j} = 1$; /* that is jth class task will be allocated to $N_{min}$ computing node */
      g. Allocate jth class tasks to computing node $N_{min}$;
```

3. End for

The previously outlined task allocation algorithm aims to minimize the mean response time for multiclass tasks running in a heterogeneous distributed computing systems. In step 1, the algorithm starts by reordering and labeling all the classes in a way that gives higher priorities to classes with higher traffic intensity. This is done by reordering the classes in a way that the following condition; $\frac{\lambda_1}{\sum_{i=1}^{n}\mu_{i1}} \geq \frac{\lambda_2}{\sum_{i=1}^{n}\mu_{i2}} \geq .... \geq \frac{\lambda_m}{\sum_{i=1}^{n}\mu_{im}}$ is satisfied.

For every class *j*, the algorithm sets initial values for the class response time TC, lightest computing node N_{min}, and lowest workload L_{min} in steps 2.a, and 2.b. Then in step 2.c, the algorithm creates a set A_j of CNs that satisfy the class task hardware and software requirements. Step 2.d is the core of the proposed allocation algorithm. It selects the computing node N_{min} in A_j that minimizes the jth class mean response time and has the lightest work load L_{min} by computing the relative workload using Eq. 14. This is done by estimating the jth class expected mean response times and the relative workloads for all CNs in the set A_j using equations 11 and 13 respectively. Step 2.d in the algorithm has three sub-steps. The sub-step 2.d.i assumes that the jth class tasks will be allocated to the ith computing node (i in A_j) by setting P_{ij}=1 and then it computes the jth class expected mean response times on that CN using Eq. 12. After that, in the sub-step 2.d.ii, the algorithm computes the relative workload L_i for the candidate computing node i in A_j using Eq. 14. Finally, in the sub-step 2.d.iii, the algorithm determines the computing node N_{min} in A_j that minimizes jth class tasks mean response time and having the lightest relative workload. Steps 2.f, and 2.g allocate jth class tasks to N_{min} node by setting $P_{N_{\min} j} = 1$ which minimizes the class mean response time.

4.2 Performance Metrics

The performance metrics used to evaluate the proposed strategy in this research are:

1. Average Task Response Time: The interval of time between the task arrival instant and the task leave instant, after finishing all needed communications and processing is known as the task response time. The average response time of multiple task classes is computed as in Eq. 13.
2. Average Computing Node Utilization: It is the average of the percentage of total node busy time out of total node available time for all computing nodes.

5. Simulation Results and Discussion

5.1 Experimental Environment

In this section, the performance of the proposed multi-class load balancing algorithm is evaluated using simulation. Even though the availability of various implementation tools for assessing scheduling policies such as network simulator NS2, OmNet++, Arena, Alea, OptorSim, and GridSim, in this study, all the simulation experiments are performed using GridSim v4.0 [25]. It has services for modeling and animating objects in distributed systems for instance users, applications, resources, and load distributers that are utilized in designing and assessing load balancing techniques. To effectively assess the performance of the suggested multi-class load balancing algorithm, an environment for the studied model is constructed having various resource features to be heterogeneous. In this environment, resources vary in OS, storage, CPU speed, and RAM. Using GridSim, Gridlet objects are used to model user tasks. It has all information required related to the task's and the details for managing execution. Also, from the Grid Information Service (GIS) entity, all the required information concerning the existing system resources could be acquired as it keeps track of all resources available in the simulation environment. A Computer (Core I3 Processor, 3.1 GHz, 4GB RAM) with Windows 7 OS is used to carry out all the simulations experiments.

5.2 Simulation Results and Analysis

Tasks form different classes arrive to the scheduler sequentially according to exponentially distributed inter-arrival times that are independent and identical. Instantaneous arrivals are not allowed. Service times also generated using exponential distribution. Task parameters (such as size, requirements and needed service) are randomly generated. For reliability, every presented result is the mean value attained from 10 simulation rounds using various seeds in generating random numbers. In the simulator, collecting statistics starts after ten minutes warm up interval. All time units are in seconds. The performance of the proposed algorithm is evaluated using a wide range of system parameters by varying the system arrival rate λ, system service rate μ and number of computing nodes n.

The performance of a heterogeneous distributed computing model under the suggested multi-class load balancing strategy is evaluated and compared with that of two other strategies namely; Random Distribution Load Balancing Strategy (RDLBS) and Uniform Distribution Load Balancing Strategy (UDLBS). In the RDLBS a computing node for a task execution is randomly determined without putting any performance indicators to that computing node or to the system in mind. This strategy is explained in [22].

In the UDLBS the tasks flow rate from the scheduler to a computing node is set to the value $\frac{1}{n}$, where n is the computing nodes number in the system. Even though, in the suggested Multi-Class Load Balancing Strategy (PMCLBS) all the arriving user tasks from different classes to the scheduler are distributed on the computing nodes that minimize their class response time. Also, the computing nodes utilize a scheduling policy which gives higher priority to classes with higher service rates and this policy minimizes the response time as it is proved in [20].

In the first experiment, we study the effect of varying the system's traffic intensity from 0.1 to 0.9 on the performance of a heterogeneous distributed computing system having 16 computing nodes. **Fig. 3**, shows the system average response time of the three evaluated scheduling strategies. From that figure, one can easily notice that the PMCLBS outperforms the RDLBS and the UDLBS in terms of average response time. The improvement in response time increases as the system traffic intensity increases. This performance was anticipated because the PMCLBS allocates tasks to a computing node which minimizes the class response time, and the computing node in turn utilizes a scheduling policy which gives higher priority to classes with higher service rates. In contrast, the RDLBS randomly selects a computing node for executing task and the computing node in turn services tasks using pure FCFS scheduling policy. Also, the UDLBS evenly distributes the system workload on the computing nodes without putting any (computing node, or system) performance metrics in mind, and the computing node in turn services tasks using pure FCFS scheduling policy.

Fig. 4, shows the average computing node utilization using the three evaluated scheduling strategies. From that figure, it is clear that as the system traffic intensity increases, the computing nodes utilization increases using the three evaluated scheduling strategies. The utilization of computing nodes is almost the same for low system workloads. The PMCLBS utilizes the computing nodes more efficiently that the RDLBS, and the UDLBS for higher system workloads.

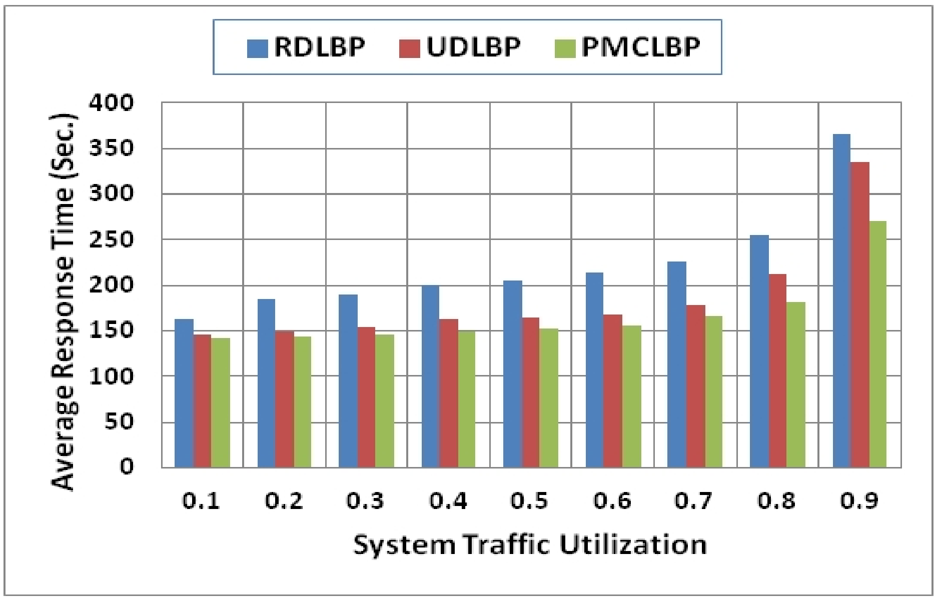

Fig. 3. Task average response time

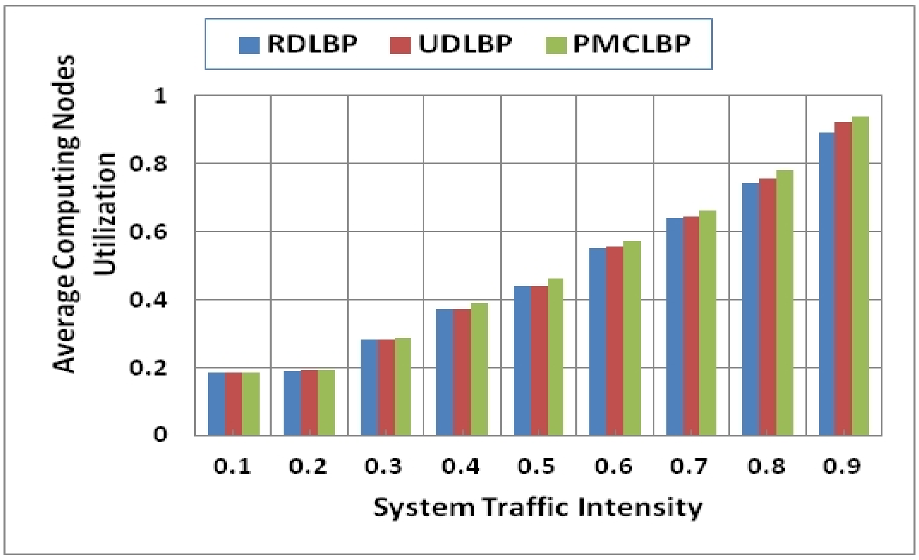

Fig. 4. Average computing nodes utilization

In the second experiment, the effect of varying number of computing nodes n form 2 to 128 on the system performance using the same performance metrics is studied. **Fig. 5** shows the effect of changing number of computing nodes on the average task response time using the three evaluated scheduling strategies. It is clear from that figure that PMCLBS outperforms the RDLBS and the UDLBS. The improvement is of a certain magnitude for moderate number of computing nodes. The performance of the three studied load balancing strategies converges when the number of computing nodes is low or high. This is logical because when the number of computing nodes is low, the computing nodes become highly loaded and almost all tasks suffer from waiting in the queues. Also, when the number of computing nodes increases, their utilization decreases because many of them will be available. This removes the effect of balancing the workload using the three evaluated load balancing strategies, and this explains why the response time obtained by the three strategies is almost the same for low and high number of computing nodes. This result is ensured in **Fig. 6** which presents the effect of varying the number of computing nodes on their average utilization using the three evaluated scheduling strategies. It is noticed from that figure that the computing nodes utilization decreases as their number increases. Again, one can notice that the PMCLBS outperforms the other two strategies in terms of average computing node utilization when the number of computing nodes is moderate. Also, when the number of computing nodes is low or high, the performance of three studied scheduling strategies is almost the same for the same reason explained earlier.

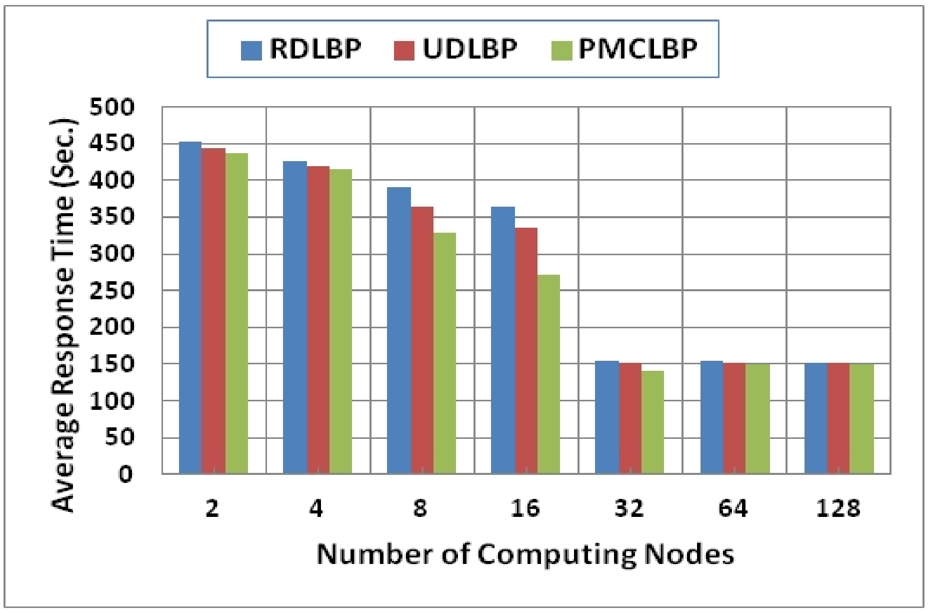

Fig. 5. Average response time versus number of computing nodes

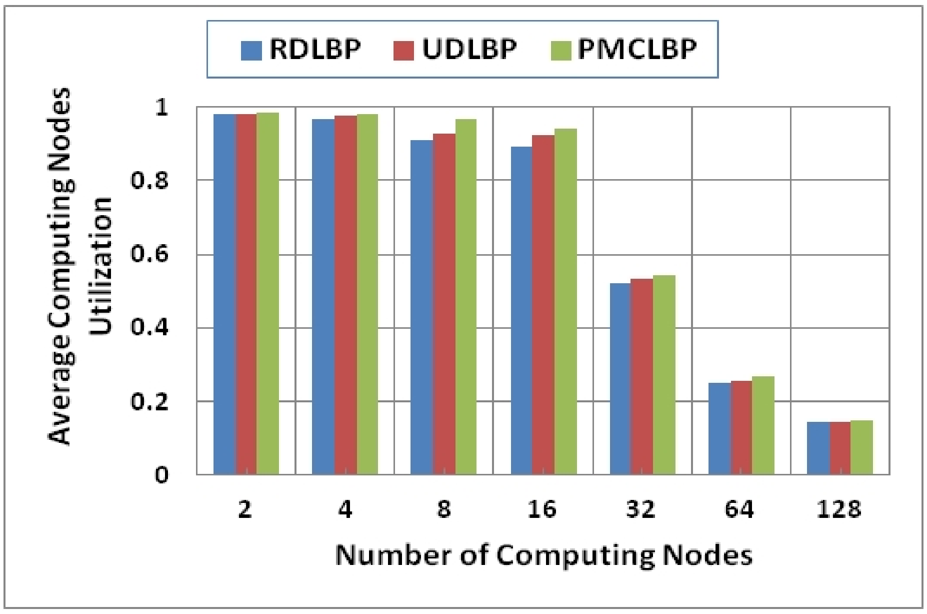

Fig. 6. Computing nodes utilization versus number of computing

5. Conclusion

In this research paper, the problem of scheduling various classes of user jobs on heterogeneous distributed computing systems is studied. A multi-class load balancing strategy aiming to minimize the average per-class response time is proposed. The performance of the suggested multi-class load balancing strategy is compared with that of the random distribution, and uniform distribution load balancing strategies. The simulation results show that, the propose scheduling strategy overtakes the random and uniform distribution load balancing strategies in terms of average system response time, and average computing nodes utilizations. This improvement is clear for moderate system workload. When the system workload is heavy or light the performance of the three studied scheduling strategies converges. In the future, an extension to the proposed multi-class load balancing strategy to be able to deal with more complicated hierarchical models that reflect the real models will be studied

Acknowledgments

The authors would like to extend their sincere appreciation to the Deanship of Scientific Research at King Saud University for its funding of this research through Research Group Project RGP-VPP-229.

References

[1] 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.

[2] S. F. El-Zoghdy, “A Hierarchical Load Balancing Policy For Grid Computing Environments,” *International Journal of Computer Network and Information Security*, Vol. 4, pp. 9-20, 2012. Article (CrossRef Link).

[3] S. F. El-Zoghdy, "A capacity-based load balancing and job migration algorithm for heterogeneous Computational grids," *International Journal of Computer Networks & Communications (IJCNC)* Vol.4, No.1, pp. 113-125, 2012. Article (CrossRef Link)
[4] S. F. El-Zoghdy, H. Kameda, and J. Li, "A comparative study of static and dynamic individually optimal load balancing policies," in *Proc. of the IASTED Inter. Conf. on Networks, Parallel and Distributed Processing and Applications*, pp. 200-205. 2002. Article (CrossRef Link)
[5] I. Foster and C. Kesselman (editors), " The Grid2: Blueprint for a New Computing Infrastructure," *Morgan Kaufmann Puplishers, 2nd edition*, USA, 2004.
[6] 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. Article (CrossRef Link).
[7] Paritosh Kumar, "Load Balancing and Job Migration in Grid Environment," *MS. Thesis*, THAPAR UNIVERSITY, 2009.
[8] K. Li, "Optimal load distribution in nondedicated heterogeneous cluster and grid computing environments," *Journal of Systems Architecture*, vol. 54, pp. 111–123, 2008. Article (CrossRef Link).
[9] H. Kameda, J. Li, C. Kim, and Y. Zhang, "Optimal Load Balancing in Distributed Computer Systems," *Springer*, London, 1997. Article (CrossRef Link).
[10] S. K. Goyal, "Adaptive and dynamic load balancing methodologies for distributed environment: a review," *International Journal of Engineering Science and Technology (IJEST)*, Vol. 3 No. 3, pp. 1835-1840, 2011.
[11] B. Yagoubi and Y. Slimani, "Task Load Balancing Strategy for Grid Computing," *Journal of Computer Science*, vol. 3, no. 3: pp. 186-194, 2007. Article (CrossRef Link).
[12] Xiao Qin, and Tao Xie," An Availability-Aware Task Scheduling Strategy for Heterogeneous Systems," *IEEE Transactions on Computers*, vol. 57, no. 2, pp. 188-199, 2008. Article (CrossRef Link).
[13] Eddy Caron, Vincent Garonne, and Andrei Tsaregorodtsev, "Definition, Modeling and simulation of a grid computing scheduling system for high throughput computing," *Future generation of computer systems*, Vol. 23, pp.968-976, 2007. Article (CrossRef Link).
[14] Hameed Hussain, Saif Ur Rehman Malik, and others, "A survey on resource allocation in high performance distributed computing systems," *parallel computing*, Vol. 39, pp. 709-736, 2013.
[15] Hameed Hussain, Saif Ur Rehman Malik, and others, "A survey on resource allocation in high performance distributed computing systems," *parallel computing*, Vol. 39, pp. 709-736, 2013.
[16] Siu-Cheung Chau, and Ada Wai-Chee Fu, "Load Balancing between Computing Clusters," *Lecture Notes in Computer Science*, Volume 3032, pp 75-82, 2004. Article (CrossRef Link).
[17] Ms. Nitika, Ms. Shaveta, Mr. Gaurav Raj, "Comparative Analysis of Load Balancing Algorithms in Cloud Computing," *International Journal of Advanced Research in Computer Engineering & Technology*, Volume 1, Issue 3, May 2012.
[18] Amandeep Kaur Sidhu, and Supriya Kinger, "Analysis of Load Balancing Techniques in Cloud Computing," *International Journal of Computers & Technology*,, Volume 4 No. 2, March-April, 2013.
[19] Y. Fang, F. Wang, and J. Ge, "A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing," *Web Information Systems and Mining, Lecture Notes in Computer Science*, Vol. 6318, 2010, pages 271-277. Article (CrossRef Link).
[20] Satish Penmatsa, and Anthony T. Chronopoulos, "Dynamic Multi-User Load Balancing in Distributed Systems," in *Proc. of IEEE International Parallel and Distributed Processing Symposium*, IPDPS 2007. 26-30 March 2007 Article (CrossRef Link).
[21] J. Sethuraman, and M. S. Squillante, "Optimal Stochastic Scheduling in Multiclass Parallel Queues," in *Proc. of ACM Sigmetric Conf.*, May 1999. Article (CrossRef Link).
[22] R. Buyya, "A grid simulation toolkit for resource modelling and application scheduling for parallel and distributed computing," Article (CrossRef Link)

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

[24] 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. Article (CrossRef Link)

[25] Malarvizhi Nandagopal and Rhymend V. Uthariaraj, "Hierarchical Status Information Exchange Scheduling and Load Balancing For Computational Grid Environments," *IJCSNS International Journal of Computer Science and Network Security*, VOL.10 No.2, pp. 177-185, 2010.

[26] 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.

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

[28] 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. Article (CrossRef Link).

[29] R. Sharma, V. K. Soni, M. K. Mishra, and P. Bhuyan, "A survey of job scheduling and resource management in grid computing," *World Academy of Science, Engineering and Technology*, 64, pp.461-466, 2010.

[30] Grosu, D., and Chronopoulos, A.T.: "Noncooperative load balancing in distributed systems," *J. Parallel Distrib. Comput. 65(9)*, pp. 1022–1034, 2005. Article (CrossRef Link).

[31] Penmatsa, S., and Chronopoulos, A.T.: "Job allocation schemes in computational Grids based on cost optimization,' in *Proc. of 19th IEEE Inter. Parallel and Distributed Processing Symposium*, Denver, 2005. Article (CrossRef Link).

[32] N.Malarvizhi, and V.Rhymend Uthariaraj, "A New Mechanism for Job Scheduling in Computational Grid Network Environments," in *Proc. of 5th Inter. Conference on Active Media Technology*, vol. 5820 of Lecture Notes in Computer Science, Springer, pp. 490-500, 2009. Article (CrossRef Link).

[33] H. Johansson and J. Steensland, "A performance characterization of load balancing algorithms for parallel SAMR applications," Uppsala University, Department of Information Technology, Tech. Rep. 2006- 047, 2006.

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

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

[36] N. Malarvizhi, and V.Rhymend Uthariaraj, "Hierarchical Load Balancing Scheme for Computational Intensive Jobs in Grid Computing Environment," in *Proc. of Int. Conf on Advanced Computing*, India, Dec 2009, pp. 97-104. Article (CrossRef Link).

[37] C. K. Pushpendra, and S. Bibhudatta, "Dynamic load distribution algorithm performance in heterogeneous distributed system for I/O- intensive task," *TENCON 2008, IEEE Region 10 Conference,19-21*, pp.1 – 5, Nov. 2008. Article (CrossRef Link)

[38] Raj Jain, "The Art of Computer System Performance Analysis," John Wiley & Sons, Inc, 1991.

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

[40] Jie Li, and Hisao Kameda, "Load Balancing Problems for Multiclass Jobs in Distributed/Parallel Computer Systems," *IEEE Trans. On Computers*, Vol. 47, NO. 3,1998. Article (CrossRef Link)

[41] Zhao Tong, Zheng Xiao, Kenli Li, and Keqin Li," Proactive Scheduling In Distributed Computing – A Reinforcement Learning Approach," *J. Parallel Distrib. Comput.*, 2014 (In press). Article (CrossRef Link).

S. F. Elzoghdy received the B.Sc. (Hons.), M.Sc., degrees in Computer Science from Faculty of Science, Menoufia University, Shebi El-kom, Egypt, in 1993, 1997, respectively. He awarded the PhD degree from Operating System & Distributed/Parallel Processing (OSDP) laboratory, Institute of Information Sciences and Electronics (IISE), University of Tsukuba, Tsukuba Science City, Japan in 2004. He is currently an Associated Prof. at the Mathematics and Computer Science Dept., Faculty of Science, Menoufia University. His current research interests include load balancing in distributed and parallel computer systems, modeling and simulation, grid computing load balancing, security and cryptography, design and analysis of parallel algorithms.

Ahmed Ghoneim received his M.Sc. degree in software modeling from University of Menoufia, Egypt, and the Ph.D. degree from the University of Magdeburg (Germany) in the area of software engineering, in 1999 and 2007 respectively. He is currently an assistant professor at the department of software engineering, king Saud University. His research activities address software evolution; service oriented engineering, software development methodologies, Quality of Services, Net-Centric Computing, and Human Computer Interaction (HCI).