An Efficient Algorithm for Resource Allocation in Parallel and Distributed Computing Systems

S. F. El-Zoghdy
Computer Science Dep.,
College of Computers &
Information Technology,
Taif University, Taif, KSA

M. Nofal
Computer Engineering Dep.,
College of Computers &
Information Technology,
Taif University, Taif, KSA

M. A. Shohla
Computer Engineering Dep.,
College of Computers &
Information Technology,
Taif University, Taif, KSA

A. El-sawy
Computer Science Dep.,
College of Computers &
Information Technology,
Taif University, Taif, KSA

Abstract*— Resource allocation in heterogeneous parallel and distributed computing systems is the process of allocating user tasks to processing elements for execution such that some performance objective is optimized. In this paper, a new resource allocation algorithm for the computing grid environment is proposed. It takes into account the heterogeneity of the computational resources. It resolves the single point of failure problem which many of the current algorithms suffer from. In this algorithm, any site manager receives two kinds of tasks namely, remote tasks arriving from its associated local grid manager, and local tasks submitted directly to the site manager by local users in its domain. It allocates the grid workload based on the resources occupation ratio and the communication cost. The grid overall mean task response time is considered as the main performance metric that need to be minimized. The simulation results show that the proposed resource allocation algorithm improves the grid overall mean task response time. *(Abstract)

Keywords-grid computing; resource management; load balancing; performance evaluation; queuing theory; simulation models (key words)

I. Introduction

As a result of advances in wide-area network technologies and the low-cost of computing resources, currently, a wide variety of parallel and distributed computing systems are available to the user community. These varieties range from the traditional multiprocessor vector systems to clusters or networks of workstations and even the geographically dispersed meta-systems connected by high-speed Internet connections (Computing Grid). Computing Grid is hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities. It enables coordinated resource sharing within dynamic organizations consisting of individuals, institutions, and resources, for solving computationally intensive applications. Such applications include, but not limited to meteorological simulations, data intensive applications, research of DNA sequences, and nanomaterials. It supports the sharing and coordinated use of resources, independently of their physical type and location, in dynamic virtual organizations that share the same goal. Thus computing grid is designed so that users won't have to worry about where computations are being performed [1-4].

Basically, grid resources are geographically distributed computers or clusters (sites), which are logically aggregated to serve as a unified computing resource. The primary motivation of grid computing system is to provide users and applications with pervasive and seamless access to vast high performance computing resources by creating an illusion of a single system image [1, 3, 5-7]. Grid Computing is becoming a generic platform for high performance and distributed computing due to the variety of services it offers such as computation services, application services, data services, information services, and knowledge services. These services are provided by the servers or processing elements in the grid computing system. The servers and the processing elements are typically heterogeneous in the sense that they have different processor speeds, memory capacities, and I/O bandwidths [5,8].

The recent development of grid computing technologies has provided us a means of using and sharing heterogeneous resources over local/wide area networks, and geographically dispersed locations. However, the Grid dynamic framework nature where resources are subjected to changes due to system performance degradation, node failure, allocation of new nodes in the infrastructure, etc. Hence, a grid resource management system (RMS) should be capable of adapting to these changes and take appropriate decisions to improve performance of users computing applications. A resource consumer is defined as an agent that controls the consumer. A RMS is defined as a service that is provided by a distributed computing system that manages a pool of named resources that is available for computing such that a system- or job-centric performance metric is optimized.

At the same time, the decisions for resource sharing should be made while maintaining the autonomy of their environments and geographical locations. Thus, the RMS should provide a highly scalable and configurable approach for sharing and securely accessing the resources [9].

To increase the system throughput, it is desired to allocate the tasks of a distributed (parallel) application program to the PEs to some objectives, ranging from the minimization of task execution time and communication cost [10–13], to the maximization of system reliability and safety [14-16]. Moreover, the system components (PEs and communication links) may be capacitated with limited amount of resources which constrains the demand of the allocated modules.

Resource allocation in heterogeneous parallel and distributed computing systems is the process of assigning (scheduling) tasks to processing elements (computers or processors) for execution such that some performance objective is optimized. For example, a common objective in resource allocation is to minimize the total response time required to complete a set of tasks [11, 12, 16, 17].

Basically, a Grid scheduler (GS) receives applications from Grid users, selects feasible resources for these applications according to acquired information from the Grid Information Service Module (GISM), and finally generates application-to-resource mappings, based on certain objective functions and predicted resource performance [18]. Unlike what happens in traditional parallel and distributed systems, GS usually cannot control Grid resources directly, but work like brokers. They are not necessarily located in the same domain with the resources which are visible to them.

In this paper, we propose a new resource allocation algorithm that would allow users to carry out their tasks by transparently accessing autonomous, distributed, and heterogeneous resources and improves the Grid computing performance in terms of mean task response time. The proposed algorithm takes into account the heterogeneity of the grid computational resources. It distributes the workload based on the resources occupation ratio and the communication cost. As in [19], we focus on the steady-state mode, where the number of tasks submitted to the grid is sufficiently large and the arrival rate of tasks does not exceed the grid overall processing capacity. The class of problems addressed by the proposed policy is the computation-intensive and totally independent tasks with no communication between them. A simulation model is built to evaluate the performance of the proposed policy. Through simulation, the performance of the proposed resource allocation algorithm is evaluated and compared with that of similar algorithms.

The rest of this paper is organized as follows: Section II presents related work. Section III describes the Grid computing model and assumptions. Section IV introduces the proposed resource allocation algorithm. Section V presents the simulation environment and results. Finally, Section VI summarizes this paper.

Related works and motivations

Resource allocation problem has been studied intensively in the traditional distributed systems literature for more than two decades. Various policies and algorithms have been proposed, analyzed, and implemented in a number of studies [20-22]. It is more difficult to achieve resource allocation in Grid computing systems than in traditional distributed computing ones because of the heterogeneity and the complex dynamic nature of the Grid systems [18--23].

Many papers have been published recently to address the problem of resource allocation in Grid computing environments. Some of the proposed algorithms for the Grid computing environments are modifications or extensions to the traditional distributed systems resource allocation algorithms. In [24], a decentralized model for heterogeneous grid has been proposed as a collection of clusters. In [17], the authors presented a tree-based model to represent any Grid architecture into a tree structure. The model takes into account the heterogeneity of resources and it is completely independent from any physical Grid architecture. However, they did not provide any task allocation procedure. Their resource management policy is based on a periodic collection of resource information by a central entity, which might be communication consuming and also a bottleneck for the system. In [18], the authors proposed a ring topology for the Grid managers which are responsible for managing a dynamic pool of processing elements (computers or processors).The resource allocation algorithm was based on the real computers workload. In [25], the authors proposed a hierarchical structure for grid managers rather than ring topology to improve scalability of the grid computing system. They also proposed a task allocation policy which automatically regulates the job flow rate directed to a given grid manager. In [26], Aram proposes a resource allocation policy using reinforcement learning by creating multiple agents. In [27], the author presents dynamic resource allocation mechanisms by using service level agreement, best fit algorithm and process migration. In [28], Tibor introduces a resource allocation protocol for providing quality of service by using probability tree modeled as an AND/OR tree and the execution of a process is carried out through a search of a solution tree. In [29], Manpreet presents a resource oriented ant algorithm using ant colony as its key allocation strategy. In [30], Rouhollah and Hadi proposed an Analytic hierarchy process (ARA) by using Multi-Criteria Decision Making (MCDM), static and dynamic methods. In [31], Adil et al. proposed a bidding-based grid resource selection by applying a single reservation mechanism. In [32], Dawei, introduces an optimizing grid resource allocation by combining fuzzy clustering with application preference. He applied a novel heuristic, min-min algorithm and ACO (Ant Colony) algorithm.

In this paper, we developed a distributed task resource allocation algorithm that can cater for the following unique characteristics of practical Grid Computing environment:

- Large-scale: As a grid can encompass a large number of high performance computing resources that are located across different domains and continents, it is difficult for centralized model to address communication overhead and administration of remote workstations.
- Heterogeneous grid resources: The Grid resources are heterogeneous in nature, they may have different hardware architectures, operating systems, computing power, resource capacity, and network bandwidth between them.
- Effects from considerable transfer delay: The communication overhead involved in capturing load information of local grid managers before making a dispatching decision can be a major issue negating the advantages of task migration. We should not ignore the considerable dynamic transfer delay in disseminating load updates on the Internet.
- Tasks are non-preemptable: Their execution on a grid resource can't be suspended until completion.
- Tasks are independent: There is no communication between tasks.

- Tasks are computation intensive (CPU-bounded): Tasks spend more time doing computations.

II. Computing Grid Model

We consider a computing grid model which is based on a hierarchical geographical decomposition structure. It consists of a set of clusters or sites present in different administrative domains. For every local domain, there is a Local Grid Manager (LGM) which controls and manages a local set of sites (clusters). Every site owns a set of processing elements (PEs) and a Site Manager (SM) which controls and manages the PEs in that site. Resources within the site are interconnected together by a Local Area Network (LAN). The LGMs communicate with the sites in their local domains via the corresponding SMs using a High-Speed network. LGMs all over the world are connected to the global network or WAN by switches.

Grid users can submit their tasks for remote processing (remote tasks) through the available websites browsers using the Grid Computing Service (GCS) to the LGMs. This makes the job submission process easy and accessible to any number of clients. The Global Scheduler (GS) at the LGMs distributes the arriving tasks to the SMs according to a task allocation policy which is based on the available information about the SMs. Also, any local site or cluster user can submit his computing tasks (local tasks) directly to the SM in his domain. Hence, any SM will have two kinds of arriving tasks namely, remote tasks arriving from its associated LGM and local tasks submitted directly to the SM by the local users. We assume that local tasks must be executed at the site in which they have been submitted (i.e., they are not transferred to any other site). The Local Scheduler at the SM in turn distributes the arriving tasks on the PEs in its pool according to a task allocation policy which is based on the PE's load information. When the execution of the tasks is finished, the GCS notify the users by the results of their tasks.

A top-down three level view of the considered computing grid model is shown in Fig. 1. It can be explained as follows:

- **Level 0:** Local Grid Manager (LGM)

Every node in this level, called Local Grid Manager (LGM), is associated with a set of SMs. It realizes the following functions:

1) It manages a pool of Site Managers (SMs) in its geographical area (domain).

2) It collects information about its corresponding SMs.

3) New SMs can join the GCS by sending a join request to register themselves at the nearest parent LGM.

4) LGMs are also involved in the task allocation and load balancing process not only in their local domains but also in the whole grid.

5) It is responsible for balancing the accepted workload between its SMs by using the GS.

6) It sends the task allocation decisions to the nodes in the level 1 (SMs).

- **Level 1:** Site Manager (SM)

Every node in this level, called Site Manager (SM), is associated with a grid site (cluster). It is responsible for:

1) Managing a pool of processing elements (computers or processors) which is dynamically configured (i.e., processing elements may join or leave the pool at any time).

2) Registering a new joining computing element to the site.

3) Collecting information such as CPU speed, Memory size, available software and other hardware specifications about active processing elements in its pool and forwarding it to its associated LGM.

4) Allocating the incoming tasks to any processing element in its pool according to a specified task allocation algorithm.

- **Level 2:** Processing Elements (PE)

At this level, we find the worker nodes (processing elements) of the grid linked to their SMs. Any private or public PC or workstation can join the grid system by registering within the nearest parent SM and offer its computing resources to be used by the grid users. When a computing element joins the grid, it starts the GCS system which will report to the SM some information about its resources such as CPU speed, memory size, available software and other hardware specifications.

Every PE is responsible for:

1) Maintaining its workload information.

2) Sending instantaneously its workload information to its SM upon any change.

3) Executing its load share decided by the associated SM based on a specified task allocation policy

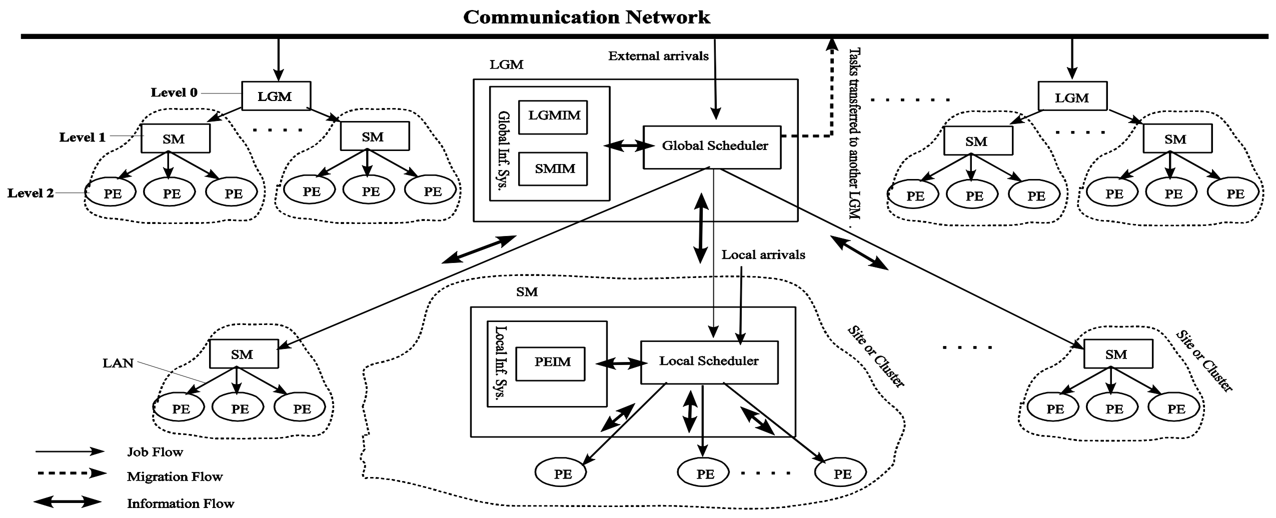

Fig. 1. Computing Grid Model Architecture

As it could be seen from this decomposition, adding or removing SMs or PEs becomes very easy, flexible and serves both the openness and the scalability of proposed grid computing model. Also, the proposed model is a completely distributed model. It overcomes the bottleneck of the hierarchal models presented in [1, 33] by removing the Grid Manager or Global node which centralizes the global load information of the entire grid. The Grid manager node can be a bottleneck and therefore a point of failure in their models. The proposed model aims to reduce the overall mean response time of tasks and to minimize the communication costs.

Any LGM acts as a web server for the grid model. Clients (users) submit their computing tasks to the associated LGM using the web browser. Upon a remote task arrival, according to the available load information, the LGM accepts the incoming task for proceeding at any of its sites or immediately forwards it to the fastest available LGM. The accepted rate of tasks will be passed to the appropriate SM based on the proposed task allocation algorithm. The SM in turn distributes these computing tasks according to the available PEs load information to the fastest available processing element for execution.

A. System parameters

For each resource participating in the grid the following parameters are defined which will be used later in the task allocation process.

1) ***Task parameters:*** *Every Task is represented by a task Id, number of task instructions NTI, and a task size in bytes TS.*

2) ***PEs parameters:*** *CPU speed, available memory, workload index which can be calculated using the total number of jobs queued on a given PE and its speed.*

3) ***Processing Element Capacity (PEC):*** *Number of tasks per second a PE can process. It can be calculated using the CPU speed and an average number of instructions per task.*

4) ***Total Site Manager Processing Capacity (TSMPC):*** *Number of tasks per second the site can process. It can be calculated as the sum of the PECs of all the processing elements of that site.*

5) ***Total Local Grid Manager Processing Capacity (LGMPC):*** *Number of tasks that can be executed under the responsibility of the LGM per second. The LGMPC can be calculated by summing all the TSMPCs for all the sites managed by the LGM.*

6) ***Total Grid Processing Capacity (TGPC):*** *Number of tasks executed by the whole grid per second. The TGPC can be calculated by summing all the LGMPCs for all the LGMs in the grid.*

7) ***Network Parameter:*** *Bandwidth size*

8) ***Performance Parameters:*** *The overall mean task response time is used as the performance parameter.*

III. PROPOSED TASK RESOURCE ALLOCATION ALGORITHM

A two-level task resource allocation algorithm for the multi-cluster grid computing environment, where clusters are located in different local area networks, is proposed. This algorithm takes into account the heterogeneity of the computational resources. It distributes the system workload based on the fastest available processing elements load balancing policy. We assume that the tasks submitted to the grid system are totally independent tasks with no inter-process communication between them, and that they are computation intensive tasks. The FCFS scheduling policy is applied for tasks waiting in queues, both at Global scheduler and Local scheduler. FCFS ensures certain kind of fairness, does not require advance information about the task execution time, do not require much computational effort, and is easy to implement. Since the SMs and their PEs resources in a site are connected using a LAN (very fast), only the communication cost between the LGMs and the SMs is considered.

The proposed task allocation algorithm is explained at each level of the grid architecture as follows:

A. Local Grid Manager Level

A LGM is responsible of managing a group of SMs as well as exchanging its load information with the other LGMs. It has Global Information System (GIS) which consists of two information modules: Local Grid Managers Information Module (LGMIM) and the Sites Managers Information Module (SMIM). The LGMIM contains all the needed information about the other LGMs such as load information and communication bandwidth size. The LGMIM is updated periodically by the LGMs. Similarly, the SMIM has all the information about the local SMs managed by that LGM such as load information, memory size, communication bandwidth, and available software and hardware specifications. Also, the SMIM is periodically updated by the SMs managed by that LGM. Since the LGMs communicate using the global network or the WAN (slow internet links) while the LGM communicates with its SMs using a High Speed network (fast communication links), the periodical interval for updating LGMIM t_G is set to be greater than the periodical interval for updating the SMIM (t_S i.e., $t_G > t_S$) to minimize the communication overhead. The GS uses the information available in these two modules in taking the task allocation decisions.

When an external (remote) task arrives at ith LGM, its GS does the following steps:

Step 1: Workload Estimation

1) To minimize the communication overhead, based on the information available at its SMIM which is more frequently updated than the LGMIM (since $T_G > T_S$), the GS accepts the task for local processing at the current LGM$_i$ ***if*** *that LGM is in the steady state (i.e., $\rho_i < 1$) and goto step 2*

else

begin {else}

a) Check the task size S in MB.

b) Based on the information available at the LGMIM, for every LGM$_K$, $K \neq i$ compute the following:

$$C_K = R_K + \frac{S}{LinkSpeed(LGM_i, LGM_k)},$$

K=(1,2,…,i-1,i+1,…,L)

where:

- $R_K = \frac{N_K}{\mu_k}$ is the occupation ratio at the LGM$_K$; where N_K is the total number of tasks at the LGM$_K$, and μ_K is the total processing capacity of the LGM$_K$.
- $LinkSpeed(LGM_i, LGM_k)$ is the speed (in Mbps) of communication link between the current LGM$_i$, and the other LGM$_K$, $K \neq i$.
- L is the number of LGMs in the whole grid.

a. Detecting the fastest available LGM to send the task to it

1) Find the LGM$_K$, $K=1,2,\ldots,i-1,i+1,\ldots,L$ having the lowest value of C_K.

2) Forward the task immediately to the LGM$_K$, update the LGMIM at the GIS and goto step 1 for servicing a new task.

end {else}

Note: We assume that a transferred task from LGM$_i$ to LGM$_K$ for remote processing receives its service at the LGM$_K$ and is not transferred to other LGMs (i.e., each task is forwarded at most once to minimize the communication cost).

Step 2: Distributing the workload accepted for processing at the LGM$_i$ on its SMs.

Based on the information available on the SMIM, for every SM number j managed by the LGM$_i$, compute the following:

$$C_{ij} = R_{ij} + \frac{S}{LinkSpeed(LGM_i, SM_j)}, \; j=(1,2,\ldots,m)$$

where:

- S is the task size in MB.
- $R_{ij=} \frac{N_{ij}}{\mu_{ij}}$ is the occupation ratio at the jth SM managed by the LGM$_i$; where N_{ij} is the total number of tasks at the jth SM managed by the LGM$_i$, and μ_{ij} is the total processing capacity of jth SM managed by the LGM$_i$.
- $LinkSpeed(LGM_i, SM_j)$ is the speed (in Mbps) of communication link between the jth SM and the LGM$_i$.
- m is the number of SMs managed by the LGM$_i$.

1) Find the SM$_j$ having the lowest value of C_{ij} (fastest available SM), j=1,2,.., m.

2) Schedule the task for processing at SM$_j$.

3) Finally update the SMIM at GIS and goto step 1 for servicing a new task.

B. Site Manager Level

As it is explained earlier, the SM or master node is responsible of monitoring a dynamic pool of heterogeneous processing elements (PEs) that are connected via a LAN and taking the task allocation decisions to distribute the workload on the PEs in its pool. It has Local Information System which handles all the information about all the PEs managed by that SM such as load information, memory size, and available software and hardware specifications. This information is stored in what is called Processing Elements Information Module (PEIM). Since the SM and the PEs within its site are interconnected via a LAN which is regularly very fast, the PEIM is instantaneously updated by the PEs when any change occurs in their state and the communication cost within a site is ignored.

To be close to reality, any local site or cluster user can submit its computing tasks (local tasks) directly to the SM. Hence, any SM will have two different kinds of arriving tasks namely, remote tasks arriving from the associated LGM and local tasks submitted directly to the SM by the local users. To limit the communication cost, we assume that local tasks will be executed at the site in which they have been submitted as long as the site is in the steady state otherwise, the LS forwards the exceeded rate to the associated LGM. The SM periodically updates the GIS at the LGM with its load and resources information. The SM periodically updates the GIS at the LGM with its load and resources information. The LS at the SM will use a task allocation policy similar to that used by the GS at LGM. This means that the site workload will be distributed among its group of PES based on the fastest available PE policy. Using this policy, the utilization of PEs will be maximized, and hence their throughput will be improved which leads to improve whole system performance.

The LS schedules the arriving tasks, either remote or local, based on the FCFS policy. For any arriving task, the LS does the following:

Step 1: Workload Estimation

(i) Based on the information available at the PEIM, the LS, for every PE_K, k=1,2,…,n, computes the occupation ratio:

$R_{ijk} = \dfrac{N_{ijk}}{\mu_{ijk}}$, j=1,2,…,m and k=1,2,…,n for m SMs.

where:

- N_{ijk} is the total number of tasks in the queue of the kth PE at jth SM managed by ith LGM (LGM_i).
- μ_{ijk} is the processing capacity of kth PE at jth SM managed by ith LGM (LGM_i).

Step 2: Decision Making (Finding the fastest PE available to process the task in it)

1) Find the PE_K, K=1,2,...,n having the lowest value of R_{ijk}

2) Schedule the task for processing at that PE_k and goto step 1 to schedule a new task.

C. Performance Metrics

We refer to the length of time between the instant from the task arrival time to the grid and the instant when it leaves the grid, after all processing and communication are over as the task response time. Let r_j be the response time of $task_j$, hence the overall mean response time RT is given by: $RT = \dfrac{1}{N}\sum_{j=1}^{N} r_j$, where *N* is the total number of processed tasks.

IV. Simulation Results and Discussion

A. Simulation Tool and Environment

Even though there are many available tools for simulating scheduling algorithms in Grid computing environments such as Bricks, OptorSim, SimGrid, GangSim, Arena, Alea, and GridSim, see [34] for more details, the simulation was carried out using the GridSim v4.0 simulator [35]. It provides facilities for modeling and simulating entities in grid computing environments such as heterogeneous resources, system users, applications, and resource load balancers which are used in designing and evaluating load balancing algorithms. In order to evaluate the performance of the proposed task allocation algorithm, a heterogeneous grid environment was built using different resource specifications. The resources differ in their operating systems, RAM, and CPU speed. In GridSim, tasks are modeled as Gridlet objects which contain all the information related to the task and the execution management details. All the needed information about the available grid resources can be obtained from the Grid Information Service (GIS) entity that keeps track of all resources available in the grid environment.

B. Simulation Tool and Environment

Even though there are many available tools for simulating scheduling algorithms in Grid computing environments such as Bricks, OptorSim, SimGrid, GangSim, Arena, Alea, and GridSim, see [34] for more details, the simulation was carried out using the GridSim v4.0 simulator [35]. It provides facilities for modeling and simulating entities in grid computing environments such as heterogeneous resources, system users, applications, and resource load balancers which are used in designing and evaluating the task allocation algorithms. In order to evaluate the performance of the proposed algorithm, a heterogeneous grid environment was built using different resource specifications. The resources differ in their operating systems, RAM, and CPU speed. In GridSim, tasks are modeled as Gridlet objects which contain all the information related to the task and the execution management details. All the needed information about the available grid resources can be obtained from the Grid Information Service (GIS) entity that keeps track of all resources available in the grid environment.

All simulations experiments have been performed on a PC (Dual Core Processor, 3.2 GHz, 2GB RAM) running on Windows xp OS. The bandwidth speed between LGMs (low capacity link) was set to 10Mbps, and the bandwidth speed between LGMs and SMs (high capacity link) varies from 50Mbps to 100Mbps. All time units are in seconds.

C. Performance evaluation and Analysis

Both of the external (remote) tasks and local tasks arrive sequentially to the LGMs and the SMs respectively with inter-arrival times which are independent, identically, and exponentially distributed. Simultaneous arrivals are excluded. The service times of LGMs are independent and exponentially distributed. Task parameters (size and service demand) are generated randomly. Each result presented is the average value obtained from 5 simulation runs with different random numbers seeds.

Experiments 1:

On a heterogeneous grid model consisting of 3 LGMs having 4, 2, 1, 5 SMs respectively. The total grid processing capacity is set to 1000 task/second (t/s). For this model to be stable, total task arrival rate (remote arrivals plus local arrivals) must be less than 1000 t/s.

During experiments explanation, task allocation and load balancing are used interchangeably. In this experiment, we focused on the results related to objective parameter (i.e., overall mean task response time) according to various numbers of tasks. During the experiment, 20 % from the total tasks arrived to the SMs are local tasks. In Fig. 3, we compare between the grid overall mean task response time obtained under the proposed load balancing (task allocation) policies (PLBPs) and that obtained without using any load balancing policies at all (No. LB). From that figure, we can see that as the number of tasks increases the overall mean task response time increases. The increase of grid overall mean task response time is less in PLBPs as compared to the increase in the grid overall mean task response time without using any load balancing policies.

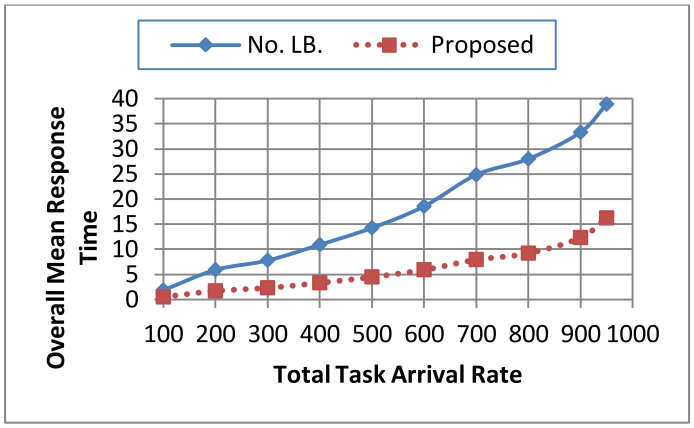

Fig. 3. Grid Overall Mean Task Response Time Of Plbps Vs. No. LB

To evaluate how much improvement is obtained in the grid overall mean task response time as a result of applying the PLBPs, we computed the improvement ratio $(T_N - T_P)/T_N$, where T_N is the grid overall mean task response time without using any balancing polices, and T_P is the grid overall mean task response time under PLBPs, see Fig. 4. From that figure, one can see that the improvement ratio gradually decreases as the grid workload increases, and it decreases rapidly as the grid workload approaches the saturation point (i.e., traffic intensity $(\lambda/\mu)\approx 1$). The maximum improvement ratio is about 73% and is obtained when the grid workload is low. This result was anticipated since the PLBPs distribute the grid workload based on the resources occupation ratio and the communication cost which leads to maximizing grid resources utilization and as a result the grid overall mean task response time is minimized. In contrast, the distribution of the grid workload on the resources without using any loads balancing policies (No. LB.) leads to unbalanced workload distribution on the resources, which leads to poor resources utilization and hence, the grid performance is affected.

Experiments 2:

In this experiment, the performance of the PLBPs is compared with that of Random_GS and Random_LS policies described in [33], and Min_load and Min_cost policies described in [36]. Our model is limited to approach their models by reducing the number of LGMs to 1 and setting the Local Tasks Arrival Rate (LTAR) to 0 (i.e., no local arrivals is allowed). In this case the LGM represent the Grid Manager (GM) or Global Scheduler (GS) in their models. During the experiment, we set the number of SMs to 4 with total processing capacity of 550 t/s.

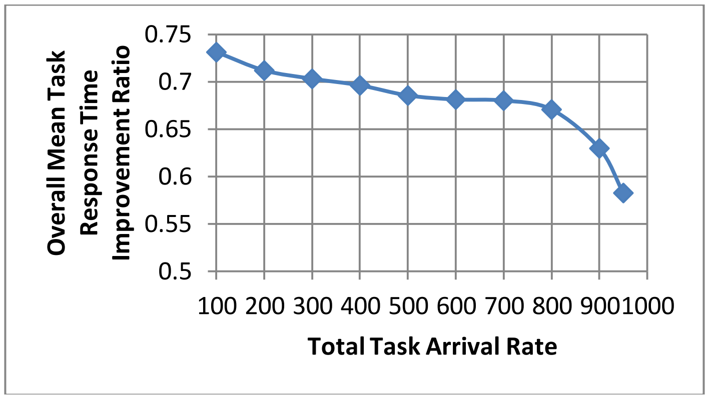

Fig. 4. Grid overall mean task response time improvement ratio

For this model to be stable, external arrival rate must be less than 550 t/s. Each simulation ends after 550,000 tasks are completed. Fig. 5 shows the overall mean task response time obtained under the Random_GS and Random_LS, Min_Load and Min_Cost, and the proposed load balancing policies. From that figure, we can see that the grid overall mean task response time obtained by all policies increases as the total arrival rate increases. Also from that figure, we can see that the PLBPs outperforms the Random_GS and Random_LS, and Min_Load and Min_Cost policies in terms of grid overall mean task response time.

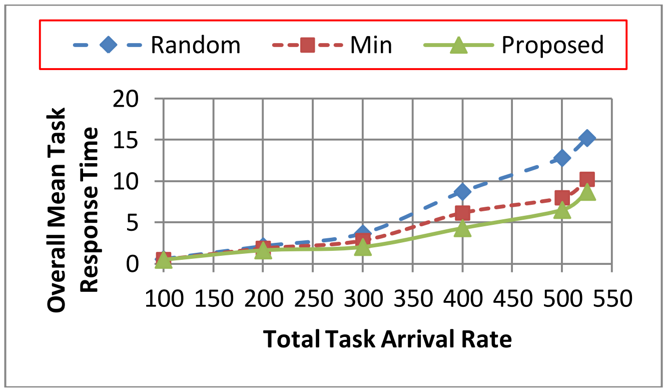

Fig. 5. Grid overall mean task response time of Random_GS and Random_LS, Min_Load and Min_Cost, and the proposed load balancing policies.

To evaluate how much improvement is obtained in the grid overall mean task response time as a result of applying the PLBPs over the other policies, we computed the improvement ratios $(T_R - T_P)/T_R$, and $(T_M - T_P)/T_M$ where T_R, T_M, and T_P are the grid overall mean task response time obtained using the Random_GS and Random_LS, Min_Load and Min_Cost, and the PLBPs, see Fig. 6. From that figure, one can see that the PLBPs outperforms the Random_GS and Random_LS, and Min_Load and Min_Cost policies in terms of grid overall mean task response time and the maximum improvement is bout 50% and 30% respectively. The improvement ratio gradually increases as the grid workload increases until the workload becomes moderate where the maximum improvement ratio is obtained and after that the improvement ratio decreases gradually as the grid workload increases approaching the saturation point (i.e., traffic intensity (λ/μ)≈1).

This result was anticipated since the PLBPs distribute the grid workload based on the resources occupation ratio which leads to maximizing the resources utilization and as a result, the grid overall mean response time is minimized. In contrast, the Random_GS and Random_LS load distribution policies distribute the workload on the resources randomly without putting any performance metric in mind which may lead to unbalanced workload distribution.

This situation leads to poor resources utilization and hence, the grid performance is degraded. Also, Min_Load and Min_Cost load balancing policies suffer from higher communication cost compared to the PLBPs. Notice that in the PLBPs, once a task is accepted by a LGM, it will be processed by any of its sites and it will not be further transferred to any other LGM. In contrast to the Min_Load and Min_Cost load balancing policies where a task may circulate between the grid resources leading to higher communication overhead. To be fair, we must say that according to the obtained simulation results, the performance of the Min_Load and Min_Cost load balancing policies is much better than that of the Random_GS and Random_LS distribution policies.

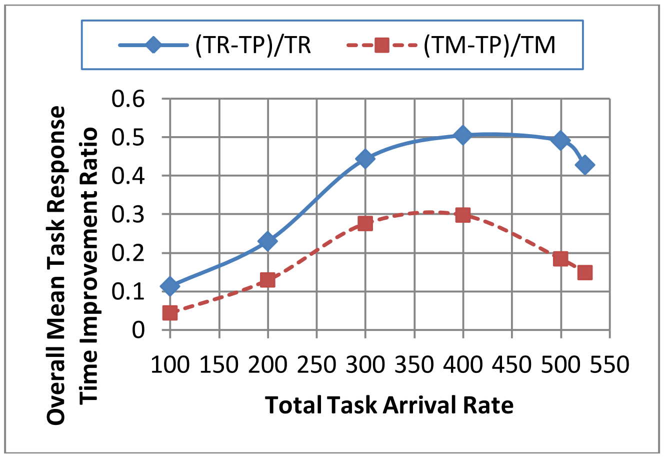

Fig. 6. Improvement ratio obtained by the proposed load balancing policies over Random_GS and Random_LS, and Min_Load and Min_Cost policies.

Experiments 3:

This experiment is done to study the effect of the local arrival rate on the performance of the PLBPS. During the experiment, the same grid parameters setting of the second experiment is used, and we set the ratio of the LTAR=0% , LTAR=10% and 25% form the TTAR to the grid. As it can be seen form Fig. 7, the overall mean task response time decreases as the LTAR ratio from the TTAR increases. This result is obvious since the LTAR arrives directly to the SMs and don't suffer from any transmission delay at all.

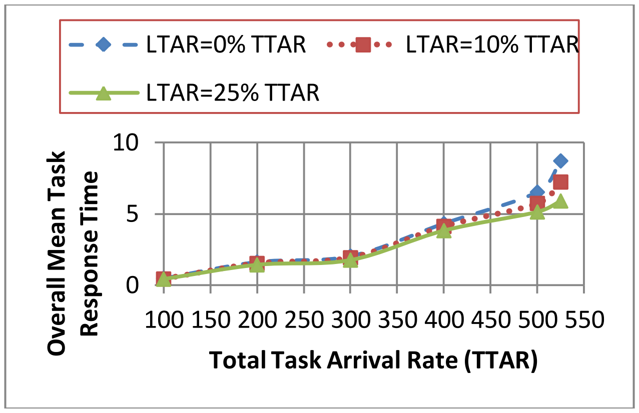

Fig. 7. Grid overall mean task response obtained for different ratios of LTAR from TTAR.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed a decentralized two-level task allocation algorithm for allocating the workload in a multi-cluster grid environment where clusters are located at administrative domains. The proposed algorithm takes into account the heterogeneity of the grid computational resources, and it resolves the single point of failure problem which many of the current policies suffer from. The task allocation decisions in this policy are taken at the local grid manager and at the site manager levels. The proposed policy allows to any site manager to receive two kinds of tasks namely, remote tasks arriving from its associated local grid manager, and local tasks submitted directly to the site manager by the local users in its domain, which makes this policy closer to reality and distinguishes it from any other similar policy. It allocates the workload based on the resources occupation ratio and the communication cost which leads to minimize the grid overall mean task response time. To evaluate the performance of the proposed task allocation policy a simulation model is built. In this model, the grid overall mean task response time is considered as the main performance metric that need to be minimized. The simulation results show that the proposed algorithm improves the grid performance in terms of overall mean task response time.

REFERENCES

[1] B. Yagoubi and Y. Slimani, "Task load balancing strategy for grid computing", J. of Computer Science, vol. 3, no. 3: pp. 186-194, 2007.

[2] I. Foster and C. Kesselman, The Grid2: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Puplishers, 2nd edition, USA, 2004.

[3] K. Lu, R. Subrata, and A. Y. Zomaya, "On the performance-driven load distribution for heterogeneous computational grids", J. of Computer and System Science, vol. 73, no. 8, pp. 1191-1206, 2007.

[4] P. Kumar, Load Balancing and Job Migration in Grid Environment, MS. Thesis, Thapar University, 2009.

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

[6] S. Parsa and R. Entezari-Maleki ," RASA: A new task scheduling algorithm in grid environment", World Applied Sciences J. (Special Issue of Computer & IT), pp. 152-160, 2009

[7] Y. Li, Y. Yang, M. Ma, and L. Zhou, "A hybrid load balancing strategy of sequential jobs for grid computing environments", J. Future Generation Computer Systems, vol. 25, pp. 819-828, 2009.

[8] S. F. El-Zoghdy, "A capacity-based load balancing and job migration algorithm for heterogeneous Computational grids", Int. J. of Computer Networks & Communications (IJCNC) vol.4, no.1, pp. 113-125, 2012.

[9] J. Pathak, J. Treadwell, R. Kumar, P. Vitale, F. Fraticelli, and P. Alto, "A framework for dynamic resource management on the grid", HPL-2005-153, August, 2005.

[10] P.Yin, S. Yu, P. Wang, and Y. Wang, "Multi-objective task allocation in distributed computing systems by hybrid particle swarm optimization", J. Applied Mathematics and Computation, vol. 184 , pp.407–420, 2007.

[11] C.H. Lee, K.G. Shin, "Optimal task assignment in homogeneous networks", IEEE Trans. on Parallel and Distributed Systems, vol. 8, pp.119–129, 1997.

[12] A. Tom Pa, S. R. Murthy, "Optimal task allocation in distributed systems by graph matching and state space search", J. of Systems and Software, vol. 46, pp. 59–75, 1999.

[13] A. Ernst, H. Hiang, M. Krishnamoorthy, "*Mathematical programming approaches for solving task allocation problems*", Proc. of the 16th National Conf. of Australian Society of Operations Research, 2001.

[14] S. Kartik, S. R. Murthy, "Task allocation algorithms for maximizing reliability of distributed computing systems", IEEE Transactions on Computers, vol. 46 pp.719–724, 1997.

[15] S. Srinivasan, N.K. Jha, "Safety and reliability driven task allocation in distributed systems", IEEE Trans. on Parallel and Distributed Systems, vol. 10, pp. 238–251,1999.

[16] C. Hsieh, "Optimal task allocation and hardware redundancy policies in distributed computing systems", European J. of Operational Research, vol. 147, pp. 430–447, 2003.

[17] http://www.engr.uconn.edu/~lester/papers/Wseas04.pdf, A. M. Mohamed, R. Ammar and L. Lipsky, "Efficient resource allocation for parallel and distributed systems"

[18] F. Dong and S. G. Akl, "Scheduling algorithms for grid computing: state of the art and open problems", *Tech. Report No*. 2006-504, School of Computing, Queen's University Kingston, Ontario, 2006.

[19] 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.

[20] J. Regehr, J. Stankovic, and M. Humphrey, "The case for hierarchical schedulers with performance guarantees", *Tech. Report No* CS-2000-07, University of Virginia, 2000.

[21] S. Zhou, X. Zheng, J. Wang, and P. Delisle, "Utopia: A load sharing facility for large, heterogeneous distributed computing systems", J. J. Softw. Pract. Exper., vol. 23, no. 12, pp. 1305–1336, 1993.

[22] G. Banga, P. Druschel, J. Mogul,"Resource containers: A new facility for resource management in server systems", Proc. of the 3rd USENIX Symposium on Operating Systems Design and Implementation (OSDI 99), February 1999.

[23] K. Krauter, R. Buyya and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing", J. Softw. Pract. Exper., vol. 32, pp.135–164, 2002.

[24] P. Pazel, T. Eilam, L. Fong, M. Kalantar, K. Appleby, and G. Goldszmidt."Neptune: A dynamic resource allocation and planning system for a cluster computing utility", 2nd Int. Symp. on Cluster Computing and the Grid (CCGRID'02), Berlin, Germany, May 2002.

[25] S. Corsava and V. Getov, " Intelligent architecture for automatic resource allocation in computer clusters", Int. Parallel and Distributed Processing Symposium, Nice, France, Apr 2003.

[26] G. Aram, C. Karl and L. Kristina," Resource allocation in the grid using reinforcement learning", IEEE Comput. Soc., vol. 3, pp.1314-1315, 2004.

[27] I. Leila, B. Mills and A. Hennebelle, "A formal model of dynamic resource allocation in grid computing environment", Proc. of the 9th ACIS Int. Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD '08), IEEE Computer Society Washington, DC, USA, 2008.

[28] S. Manpreet, "GRAAA: Grid resource allocation based on ant algorithm", .J. Adv. Inf. Technol., vol. 1, no. 3, pp. 133-135, 2010.

[29] G. Rouhollah, and S.S. Hadi,. "A grid resource allocation method based on analytic hierarchy process", 5th Int. Sym. on Telecommunications (IST'2010), 2010.

[30] Y. Adil, , A. A. Hanan, and A. A. Atahar, "A bidding-based grid resource selection algorithm using single reservation mechanism", Int. J. Comp. Appl., vol. 16, no. 4, pp. 39-43, 2011..

[31] S. Dawei, C. Guiran, J. Lizhong and W. Xingwei, "optimizing grid resource allocation by combining fuzzy clustering with application preference", Int Conf. on Advanced Computer Control (ICACC), pp: 22-27, 2010.

[32] G. Tibor, "A resource allocation protocol for providing quality of service in grid computing, using a policy-based approach", AICT-ICIW '06 Proc. of the Advanced Int'l Conf. on Telecommunications and Int'l Conf. on Internet and Web Applications and Services, IEEE Computer Society Washington, DC, USA, 2006.

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

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

[35] R. Buyya, "A grid simulation toolkit for resource modelling and application scheduling for parallel and distributed computing", www.buyya.com/gridsim/

[36] J. Balasangameshwara, and 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.