VOL. 10, NO. 15, AUGUST 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences

www.arpnjournals.com

RESOURCE ALLOCATION MECHANISMS IN COMPUTATIONAL GRID: A SURVEY

Omar Dakkak, Suki Arif and Shahrudin Awang Nor
Inter Net Works Research Laboratory, School of Computing, Universiti Utara Malaysia, UUM Sintok, Kedah, Malaysia
E-Mail: oaldakkak@gmail.com

ABSTRACT

Nowadays, the electronic resources are available almost in every institution or facility. These electronic resources could be CPU, memory, electrical devices and so on. Most of these resources are wasted or not completely utilized. Hence, the role of Computational Grid comes. Grid Computing focuses on computing resources (such as CPU), in order to achieve a huge task in a short time. Due to the high heterogeneity in Grid environment, proposing an optimal resource allocation mechanism that can work in all scenarios is a dilemma. This paper presents a critical review about some of the most widely known and recently proposed mechanisms in Grid Computing. Thus, it will give the researchers an idea about the features of the most recent and used resource allocation mechanisms in Grid.

Keywords: grid computing, resource allocation, CPU, load balance.

INTRODUCTION

Grid Computing is a set of recourses; these resources are virtually combining their separated computational power in one computational power to execute a huge task (Czajkowski, Fitzgerald, Foster, and Kesselman, 2001; Foster, Kesselman, and Tuecke, 2001). Usually, in Computational Grid environment, the main resource is the Central Processing Unit (CPU). Therefore, Grid Computing is mostly used in the research field that demands high computational power ability to perform massive and complicated calculations. Grid gives the flexibility to the users regarding the locations of the resources. It enables the user to utilize the resources even though if the resources are distributed in different zones. Thus, Computational Grid can reduce the cost for the research sponsor, as well as the execution time for the researcher.

Grid has its own architecture which consists of five layers as depicted in Figure-1. These layers are (from down to top): fabric layer, connectivity layer, resource layer, collective layer and application layer (Foster *et al*., 2001). Fabric layer is providing the communication interface between the physical network and the rest of Grid layers. The core communication and authentication protocols that are needed for Grid network are defined and determined by the connectivity layer. Date exchanging between fabric layer resources is enabled by communication protocols. Resource layer is responsible for finding the available and idle resources that are connected to the Grid network, in order to utilize them by the user. Allocating these resources in a suitable and proper way is the main task of collective layer which is a key layer in this architecture. In addition to the scheduling, collective layer provides monitoring and data replication, whereas application layer is responsible for linking the user with the Grid network (Foster *et al*., 2001). Figure-1 shows the Grid system architecture.

Five main components are involved in Grid operation, in order to accomplish the whole process. These components are: Grid application, Grid scheduler, Grid Information Service (GIS) (Dong and Akl, 2006), job Launching and Monitoring (LM) and Local Resource Manager (LRM These components are communicating with each other via the internet).When the user (who runs the Grid application),

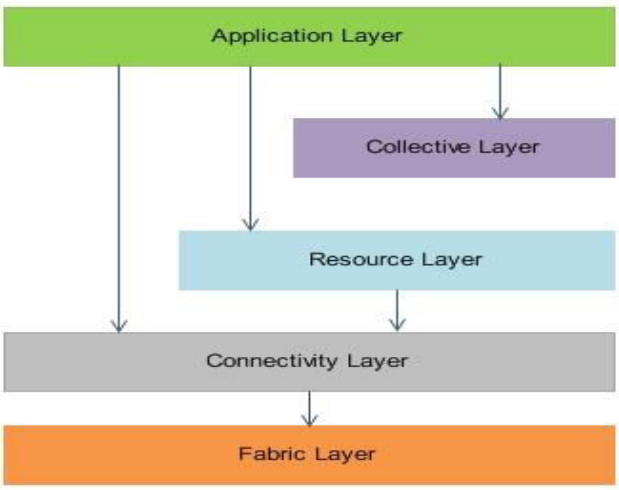

Figure-1. Grid system structural design.

sends a request to the Grid scheduler in order to execute a huge task, the Grid scheduler inquiries about resources condition from GIS, which provides the right mapping for the user request. Grid scheduler can exist in many scenarios, such as: centralized, decentralized and peer to peer (Hamscher, Schwiegelshohn, Streit, and Yahyapour, 2000). The role of GIS is providing information about the available resources status like: available bandwidth, available memory, available CPU cycle and so on. Then,

Analogical Benchmark (AB) (Khokhar, Prasanna, Shaaban, and Wang, 1993; Siegel, Dietz, and Antonio, 1996) tests the resource whether it is able to execute the task or not. This information is forwarded to the Grid scheduler, which in its turn, sends this information to LM (Czajkowski *et al*., 1998). Job launching and monitoring is responsible for arranging input, output data and monitoring the performance of the resources. Local Resource Manager (LRM) is observing the resources condition in case that any update on the resources status occurred. Then, LRM informs the GIS, therefore GIS updates its information about the available resources. Figure-2 illustrates the whole process.

The rest of the paper is organized as follows. We briefly describe the concept of resource allocation in Section 2. In Section 3, we review some of the most recent algorithms in Computational Grid. Finally, we conclude this paper in Section 4.

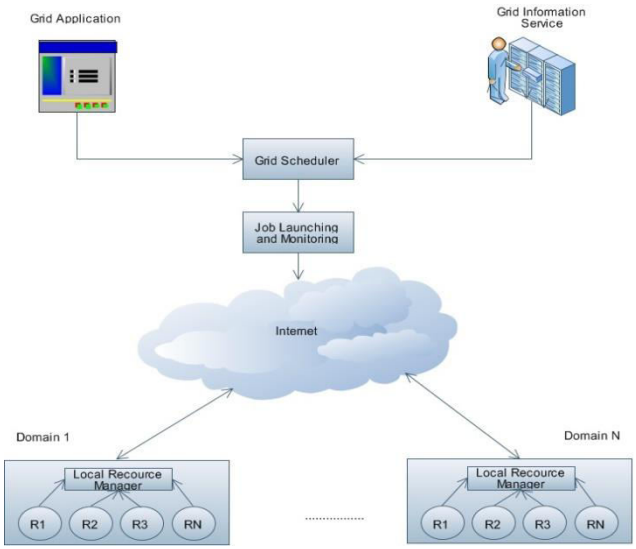

Figure-2. Grid scheduling process (Kannasoot, 2011).

THE CONCEPT OF RESOURCE ALLOCATION

Resource allocation is the core of Grid process (Ismail, 2007). Resource allocation (scheduling) refers to the selection of the best resources for the suitable job. Since the Grid environment is highly dynamic, the resources and the tasks could alter frequently. Therefore, to serve one user, that could mean the mechanism will allocate and co-allocate the mapping between the resources and the tasks considerably.

Wu *et al*. in (Wu*, Ye, and Zhang, 2005) defined the concept of resource allocation through mathematical method. According to (Wu* *et al*., 2005), if we have two groups, the first group is a set of tasks $t = [t_1, t_2, t_3, \dots t_n]$ and the second group is a set of resources $R = [r_1, r_2, r_3, \dots \dots r_n]$, when, the allocation is conducted for each resource to each task, two values will be generated. The first value reflects the utilization of the resource U, it refers to the benefit that the user gets from utilizing this resource to execute a certain task. The second value reflects the cost C, which refers to the penalty of using the resource such as: latency, communication overhead and so on. The relation between the optimal resource allocation O, the utilization of the resources U and the cost of utilizing these resource C can be given in the following equation:

$$0 = U - C \quad (1)$$

Based on this equation, it is clear that the relation between 0 and is proportional, whereas the relation between 0 and C is inverse relationship. Therefore, in order to achieve optimal (or near optimal) resource allocation$O$, the greater the U and the minimum the$C$, the closer optimal will be. The value of the allocated part of the resource could be continuous value or discrete value and non-negative in both cases. For instance, CPU processing time is continuous value, whereas discrete and indivisible value can be vehicle.

RESOURCE ALLOCATION MECHANISMS IN COMPUTATIONAL GRID

Resource allocation mechanisms are playing a very important role in the scheduling process in Grid system. The efficiency of the mechanism will determine the quality of service that users receive. There is a plethora of the existing resource allocation mechanisms in Grid system (Qureshi *et al*., 2014). This plethora in these mechanisms can be justified for two main reasons. The first reason is because of the high heterogeneity of the Grid system and that means, Grid includes many types of different topologies and different kinds of resources which are connected to the Grid. The second reason is due to the huge diversity of the Grid applications. The requirements of Grid application vary from application to another. Some applications demand high throughput, whereas other applications demands less execution time. In addition to the above, many mechanisms offer several options in terms of resources charge, executing job by minimum resource usage or maximum resource usage.

Resource allocation mechanisms could be static, dynamic or both (adaptive). Static mechanism performs the scheduling based on the compile time for the application. Whereas dynamic mechanism performs the scheduling based on the run time for the application. The static approach is more stable and easier to predict, while dynamic approach is more suitable in such environment like the Grid which is dramatically dynamic.

Some of resource allocation mechanisms perform load balancing, while other mechanisms do not. In addition, some mechanisms perform partial load balancing. In Computational Grid, load balancing means distributing the workload among the processes in order to guarantee that all processes (resources) have equal or near to equal workload. Load balancing is very important to

assure that the system is stable and scalable. As well as, when the workload is distributed well among the resources, the performance quality will be much better.

In this section, we will review some of the most recent and used resource allocation mechanisms. Any resource allocation mechanism can be centralized, distributed or hybrid.

Load forecast-based allocation

Load Forecast-Based Allocation was proposed in (Cheng and Li, 2006; Zhi-jie and Cun-rui, 2012). This mechanism aims to serve the user request as fast as possible with a limited budget for resources. After the resources get registered in Grid Information Service (GIS), the user sends a query message in order to discover the resources. The GIS responses to the user by sending the list of the available resources, then the user replies by sending back its bidding to GIS. This procedure is repeated until the user requirements are satisfied. Finally the user gets the resource prices. The CPUs in this mechanism are analyzed in order to combine them in one powerful CPU to execute the user's job. When this algorithm starts, the CPUs have no load at the beginning, later they will have load and this load will be estimated based on the status of the CPU. This mechanism has showed great result; due to all the users know the situation about each others. Thus, they can expect the load on the CPUs. Figure-3 illustrates the working steps for this algorithm.

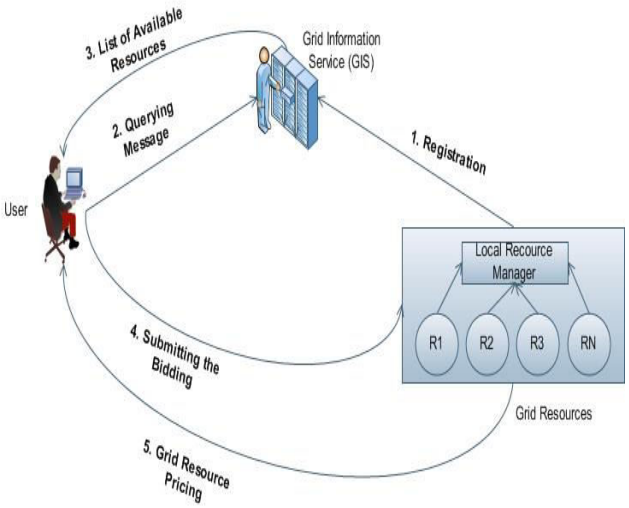

Figure-3. Load forecast-based allocation working steps.

Greedy double auction

In (Ding, Luo, and Gao, 2010), Ding *et al.* proposed Greedy Double Auction Mechanism (GDAM). The rising of the Grid economy, has caused to propose economic-based model (mechanisms) to allocate and mange the resources in Grid. The economic mechanisms have many features; the auction is one of the most significant features of these types of mechanisms. Auction concept in Grid is trading the Grid resources (such as; computing and storage entities)that are available in the Grid market (Buyya, Abramson, and Giddy, 2001; Buyya, Abramson, and Venugopal, 2005).

There are two types of auctions, the first type is Single-unit Double Auction (SDA) and the second type is Multi-unit Double Auction (MDA). MDA can deal with more than one unit bartered in each auction, thus Ding et al. has proposed GDAM algorithm based on MDA. GDAM is seeking for making trades between the user and the resource providers as much as possible with the assurance of both parties' demands. The results showed that GDAM is outperformed MDA in terms of total number of the trades (more auctioneers means more economic efficiency) and the user satisfying ratio. GDAM works fine with Grid environment, especially if the number of participating users is high.

Association based mechanism

In (Manavalasundaram and Duraiswamy, 2012), Association Based Resource Allocation Mechanism was proposed. This mechanism concerns about computational resources as well as economic rate. This mechanism is based on the concept of the clusters. It coordinates the resources in distributed clusters to form the Grid association and allows the users to share them. In this mechanism, the resource consumers are the end users, resource providers are clusters and Grid Association Agent (GAA) is representing Resource Management System (RMS). If task arrives to the Local Resource Scheduler Manager (LRM) and the local resources can't satisfy the user's task (in terms of economic cost and swift serving), then the task will be transferred to another cluster. Association Based Resource Allocation Mechanism achieves full autonomy for every user.

Load balancing on arrival

Load balancing on Arrival (LBA) (Saravanakumar and Prathima, 2010) aims to reduce the average response time by distributing the workload among the processors in order to make the load balanced in the long term. LBA is an adaptive mechanism. The main idea for this mechanism is using the workload information, to be able to determine whether the node should be a sender (overloaded) or a receiver (idle/able to run extra task (s)). This mechanism moves the job to a "buddy processor" if the current processor is overloaded. Buddy processor refers to a set of processors which are connected to each other. LBA predicts job arrival rate and CPU processing rate and thus, it can estimate the load for each processor. LBA concerns about the transferring cost (overhead) and the heterogeneity for both the resources and the network, before transferring the job to another processor (if needed). LBA supposes that each processor has an infinite buffer memory to keep the awaited jobs in the queue maintained. Also, this mechanism considers that each

processor has knowledge about its buddy processors. The communication cost between the buddy processors is also assumed to be determined. Figure-4 shows the system model for this mechanism.

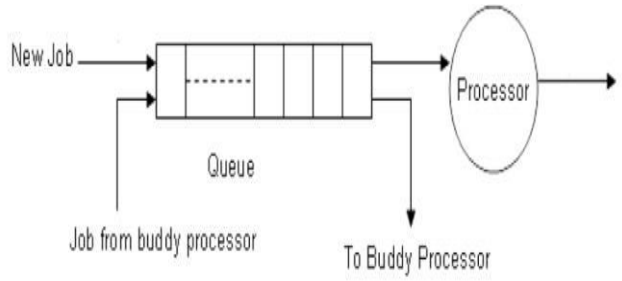

Figure-4. LBA system model (Anand, Ghose, and Mani, 1999).

LBA was evaluated with Estimated Load Information Scheduling Algorithm (ELISA) and Perfect Information Algorithm (PIA) (Anand *et al*., 1999). The evaluation results have shown that LBA outperforms both mechanisms in terms of uneven load distribution ratio and migration limit number. Even though this mechanism is designed for small Grid systems, it shows that it can balance the workload for the huge jobs efficiently.

Adaptive grid scheduling

This algorithm is decentralized and hybrid. Adaptive Grid Scheduling (AGS) (Saleh, Sarhan, and Hamed, 2012) is targeting to schedule the computational resources in Grid environment. AGS uses static and dynamic approach. AGS has two main phases. The first phase is discovering and monitoring the resources; the second phase is allocating and co-allocating the resources. Through these two phases, AGS can detect the possibility if any failure occurred in the resources, and then restore the system to the default state. To achieve this goal, AGS uses Direct Acyclic Graph (DAG). DAG enables AGS to move from the first phase to the next one. After the resources are discovered and monitored, the first phase ends and the second phase starts. AGS starts allocating the resources to the tasks by using static approach. Then, a co-allocation starts (if needed) to optimize the makespan by employing DAG features, which is In-advance Task Transmission (ITT). ITT schedules the tasks to the resources in advance before waiting the busy resources to finish from executing the current task(s). ITT conducts the scheduling with dynamic approach. Figure-5 shows the working steps for this mechanism.

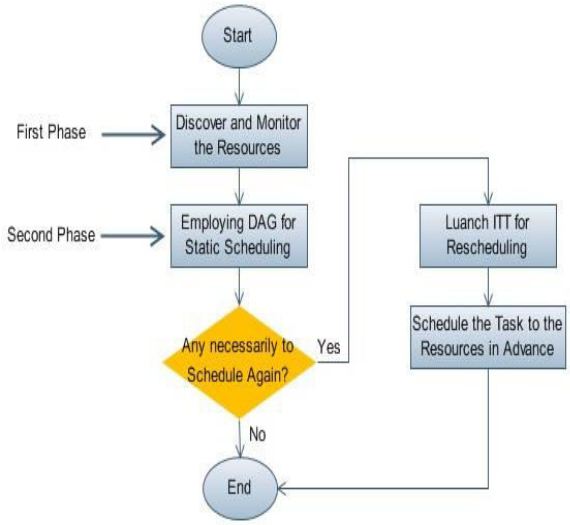

Figure-5. Adaptive grid steps.

Game theoretical energy efficient

In (Kołodziej *et al*., 2014), Zomaya *et al.* proposed Game Theoretical Energy Efficient to allocate the jobs for applications that have constrains regarding the task execution time. This mechanism aims to reduce the total power consumption and the makespan for the applications. In this mechanism, resources are called "Players". This mechanism employs Dynamics Voltage Scaling (DVS). DVS adjusts the frequency and the voltage for CPUs dynamically and smoothly. To limit the power consumption, this mechanism makes the CPUs working with low frequency. This mechanism tries to use the minimum resources to execute the small tasks in order to save the power. Game Theoretical Energy Efficient was evaluated with Min-Min heuristic algorithm. The evaluation showed that Game Theoretical Energy Efficient results in better performance than Min-Min in terms of energy consumption.

Hierarchy load balanced algorithm

Hierarchy Load Balanced Algorithm (HLBA) in (Manimala and Suresh, 2013) was designed for Computational Grid in hierarchical topology. This Algorithm mainly concerns about CPU resources. This algorithm is scheduling the jobs and performing the load balancing as well. In this algorithm, the computing powers are; CPU speed, idle CPU percentage and the average load for each cluster. Three main parameters, this algorithm is concerned about; idle CPU percentage, memory utilization and bandwidth utilization. The scheduler in HLBA gets the requested job from the user. After sending the request to the information system, the information system will get valuable information such as; Average Cluster Processing (ACP). The scheduler chooses the fastest ACP clusters, and compares between these ACPs and the average load value of the system. Based on certain measurements, the

job will select one ACP or seek for the second next high ACP.

The authors enhanced their previous algorithm by proposing Improved Hierarchy Load Balanced Algorithm (IHLBA) (Manimala and Suresh, 2013). IHLBA does the same first three steps. Then it sorts the clusters based on their average load. The aim of IHLBA is to minimize the makespan. HELBA and IHELBA are using local update and global update. Local update is useful to give update about the new value of the parameter in the cluster. Whereas global update occurs when the job execution is finished, therefore the task should release the resource. Thus, the whole parameters have to be calculated again.

Balanced overlay network

Balanced Overlay Network (BON) (Bridgewater, Boykin, and Roychowdhury, 2007) relies on the overlay approach. This approach is depending on local communication only to perform job allocation decisions through random walk. BON is scalable and almost optimal as long as the network in not overloaded. Once the network reaches to the clipping point, the links that connect the nodes, start cutting themselves. In this case, the balance remains acceptable due to each node still connected with another two nodes (power of the two choices).

Correlation coefficient method was used in Balanced Overlay Network (BON) to evaluate the load distribution performance in the system. The results showed that BON achieves near to optimal distribution, whereas when the power of the node increases, the load level increases as well.

Table-1 below shows a comparison between the previous reviewed mechanisms in terms of type, aim of the mechanism and load balance.

CONCLUSIONS

In this paper, we highlighted some mechanisms, which are utilized in Grid Computing in order to schedule the resources. First, we gave an idea about the Grid System and how it works. Then, we illustrated the concept of resource allocation based on certain points of view in literature. We reviewed some of the most widely known and recently used mechanisms. In addition, we provided a comparison between them based on the type, aim and load balance.

Furthermore, from the literature, numerous resource allocation mechanisms exist. Despite the plethora in these mechanisms, most of them are still not able to allocation in Grid Computing is still an interesting area for researchers to conduct more studies in this domain.

Table-1. Resource allocation mechanisms comparison.

Mechanism	Type	Aim	Load balance
Load Forecast-Based Allocation	Static	Fast Request with Limited Budget	No
Greedy Double Auction	Static	Resources and Price	No
Association Based Mechanism	Static	Computational Cost	No
Load Balancing on Arrival	Adaptive	Minimizing-Average Response Time	Yes
Adaptive Grid Scheduling	Adaptive	Reducing Makespan and Transmission Communication Time	No
Game Theoretical Energy Efficient	Dynamic	Minimizing Power Consumption	No
Hierarchy Load Balanced Algorithm	Adaptive	Balancing Workload	Yes
Improved Hierarchy Load Balanced Algorithm	Adaptive	Balancing Workload and Minimizing Makespan	Yes
Balanced Overlay Network	Dynamic	Select the Best Resources to Run the Task & Balance the Workload	Yes

ACKNOWLEDGMENTS

The authors wish to thank the Ministry of Education, Malaysia for funding this study under the Long Term Research Grant Scheme (LRGS/bu/2012/UUM/Teknologi Komunikasi dan Infomasi).

REFERENCES

Anand, L., Ghose, D., and Mani, V. 1999. ELISA: an estimated load information scheduling algorithm for distributed computing systems. Computers and Mathematics with Applications. 37(8), 57-85.

Bridgewater, J. S., Boykin, P. O., and Roychowdhury, V. P. 2007. Balanced overlay networks (BON): An overlay technology for decentralized load balancing. Parallel and Distributed Systems, IEEE Transactions on. 18(8), 1122-1133.

Buyya, R., Abramson, D., and Giddy, J. 2001. A case for economy grid architecture for service oriented grid computing. Paper presented at the Parallel and Distributed Processing Symposium, International.

Buyya, R., Abramson, D., and Venugopal, S. 2005. The grid economy. Proceedings of the IEEE. 93(3), 698-714.

Cheng, C.-T., and Li, Z.-J. 2006. Parallel algorithm for grid resource allocation based on nash equilibrium. Paper presented at the Machine Learning and Cybernetics, 2006 International Conference on. IEEE. pp. 4383-4388.

Czajkowski, K., Fitzgerald, S., Foster, I., and Kesselman, C. 2001. Grid information services for distributed resource sharing. Paper presented at the High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on. IEEE. pp. 181-194.

Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., and Tuecke, S. 1998. A resource management architecture for metacomputing systems. Paper presented at the Job Scheduling Strategies for Parallel Processing.

Ding, D., Luo, S., and Gao, Z. 2010. A greedy double auction mechanism for grid resource allocation. Paper presented at the Job Scheduling Strategies for Parallel Processing.

Dong, F., and Akl, S. G. 2006. Scheduling algorithms for grid computing: State of the art and open problems: Technical report.

Foster, I., Kesselman, C., and Tuecke, S. 2001. The anatomy of the grid: Enabling scalable virtual organizations. International journal of high performance computing applications. 15(3), 200-222.

Hamscher, V., Schwiegelshohn, U., Streit, A., and Yahyapour, R. 2000. Evaluation of job-scheduling strategies for grid computing Grid Computing-GRID 2000 (pp. 191-202): Springer.

Ismail, L. 2007. Dynamic resource allocation mechanisms for grid computing environment. Paper presented at the Testbeds and Research Infrastructure for the Development of Networks and Communities, 2007. TridentCom 2007. 3rd International Conference on. IEEE. pp. 1-5.

Kannasoot, N. 2011. Grid resource allocation and scheduling in optical networks: THE UNIVERSITY OF TEXAS AT DALLAS.

Khokhar, A. A., Prasanna, V. K., Shaaban, M. E., and Wang, C.-L. 1993. Heterogeneous computing: Challenges and opportunities. Computer. 26(6), 18-27.

Kołodziej, J., Khan, S. U., Wang, L., Kisiel-Dorohinicki, M., Madani, S. A., Niewiadomska-Szynkiewicz, E., . . . Xu, C.-Z. (2014). Security, energy, and performance-aware resource allocation mechanisms for computational grids. Future Generation Computer Systems. 31, 77-92.

Manavalasundaram, V. and Duraiswamy, K. 2012. Association based grid resource allocation algorithm. Eur. J. Sci. Res. 78(2), 248-258.

Manimala, R., and Suresh, P. 2013. Load balanced job scheduling approach for grid environment. Paper presented at the Information Communication and Embedded Systems (ICICES), 2013 International Conference on. IEEE. pp. 336-339.

Qureshi, M. B., Dehnavi, M. M., Min-Allah, N., Qureshi, M. S., Hussain, H., Rentifis, I., . . . Xu, C.-Z. 2014. Survey on grid resource allocation mechanisms. Journal of Grid Computing. 12(2), 399-441.

Saleh, A. I., Sarhan, A. M., and Hamed, A. M. 2012. A new grid scheduler with failure recovery and rescheduling mechanisms: discussion and analysis. Journal of Grid Computing, 10(2), 211-235.

Saravanakumar, E., and Prathima, G. 2010. A novel load balancing algorithm for computational grid. Paper presented at the Innovative Computing Technologies (ICICT), 2010 International Conference on. IEEE. pp. 1-6.

Siegel, H. J., Dietz, H. G., and Antonio, J. K. 1996. Software support for heterogeneous computing. ACM Computing Surveys (CSUR), 28(1), 237-239.

Wu, T., Ye, N., and Zhang, D. 2005. Comparison of distributed methods for resource allocation. International Journal of Production Research. 43(3), 515-536.

Zhi-jie, L., and Cun-rui, W. 2012. Resource allocation optimization based on load forecast in computational grid. Int. J. Eng. Res. Appl. (IJERA), 2(3), 1353-1358.