Experimental Performance Analysis of Job Scheduling Algorithms on Computational Grid using Real Workload Traces

Syed Nasir Mehmood Shah[1, *], Ahmad Kamil Mahmood[2], Saddaf Rubab[3], Mohd Fadzil Hassan[4]
{dr.shah@kicsit.edu.pk[1,*], kamilmh@petronas.com.my[2],
saddaf_g02754@utp.edu.my[3], mfadzil_hassan@petronas.com.my[4]}

Department of Computer Sciences, Dr. A. Q. Khan Institute of Computer Sciences & Information Technology, Kahuta, Pakistan[1]
Department of Computer and Information Sciences,
Universiti Teknologi PETRONAS, Seri Iskandar, Perak Darul Ridzuan, 32610, Malaysia[2, 3, 4]

Abstract. Grid, an infrastructure for resource sharing, currently has shown its importance in many scientific applications requiring tremendously high computational power. Grid computing, whose resources are distributed, heterogeneous and dynamic in nature, introduces a number of fascinating issues in job scheduling. Grid scheduler is the core component of a grid and is responsible for efficient and effective utilization of heterogeneous and distributed resources. This paper presents comparative performance analysis of our proposed job scheduling algorithm with other well known job scheduling algorithms considering the quality of service parameters. The main thrust of this work was to conduct a quality of service based experimental performance evaluation of job scheduling algorithms on computational Grid in true dynamic environment. Experimental evaluation confirmed that proposed scheduling algorithms possess a high degree of optimality in performance, efficiency and scalability. This paper includes statistical analysis of real workload traces to present the nature and behavior of jobs.

Keywords: Distributed systems, Cluster, Grid computing, Grid scheduling, Workload modeling, Performance evaluation, Simulation, Load balancing, Task synchronization, Parallel processing

1 Introduction

Job scheduling plays a vital role in an efficient and effective management of grid resources. Grid scheduling is divided into three phases; which are namely resource discovery, resource allocation and job execution. Resource discovery is a mechanism which generates a pool of available resources. Resource allocation deals with selection of best resources and the allocation of jobs to the selected resources accordingly. Job execution manages the execution of jobs on available resources. Resource allocation is an NP-complete problem[1, 2]

In our previous research, we presented two Grid job scheduling algorithms which are Multilevel Dual Queue Scheduling Algorithm (MDQ) and the Multilevel Hybrid Scheduling Algorithm (MH) with objective of optimizing the performance of Grid[2, 3]. We also proposed dynamic variants of MH and MDQ to manage scheduling of high computing jobs in truly scalable and dynamic computational Grid environment in our recent work [4, 5].

Scalability testing is an important success factor in the design and development of a job scheduling algorithm for Grid computing environment. Scalability is measured by analyzing the application performance by increasing and decreasing the computing power provided to it in the form of cores or processors. In [4], the concept of performance and scalability is highlighted by the authors. The terms performance and scalability are usually grouped together. Performance is defined as a speed measure with which a single application can be processed by the computing system. While, the scalability measures the capability of a application to maintain its performance under increased processing power and workload.

Two fundamental issues are addressed in performance analysis of new Grid scheduling algorithms. Firstly, representative workload traces are used to produce trustworthy results. Secondly, a good testing environment should be established. Simulation is the widely adopted strategy for evaluation of job scheduling algorithms[5].

Grid scheduling introduces a number of fascinating issues which makes the implementation of system in dynamic, scaleable and real time system a very difficult problem. The grid resources allocation is an NP-hard problem[6]. The near optimal solution for job scheduling has been presented as a heuristic in the literature [7-11]. The problem of scheduling in grid computing has been discussed in Section 2. Main focus of this research is to evaluate the performance, efficiency and scalability of scheduling algorithms in an optimized way.

The structure of the paper is as follows: Section 2 provides a brief literature review of Grid scheduling methodologies. Section 3 presents the proposed scheduling algorithms and Section 4 is about the statistical analysis of real workload traces. Section 5 illustrates the scheduling simulator design. In Section 6, the experimental setup is discussed and Section 7 describes the performance analysis of the Grid scheduling algorithms. Section 8 concludes the paper.

2 Related Research

The computation of grid resources is difficult to ensure throughout the job execution because of heterogeneous and dynamic nature of grid resources. The job scheduling for grid problem has already been discussed in the literature has attempted to minimize the makespan and computation cost. Among the most popular ones, Heterogeneous Earliest Finish Time (HEFT) [12, 13] is heuristic used to schedule the workflow applications on heterogeneous resources. HEFT assigns the tasks derived from workflow applications based on the individual task priorities.

A multi-criteria based accelerated genetic algorithm has been used in [9]. The authors have selected job response time and success rate as scheduling criterions. There are a few other job scheduling heuristics proposed to achieve the high performance from grid, which includes, Max-Min[11, 14] , Min-Min[11, 14, 15], BHEFT [16] and GA[9, 17]. A comparison of many scheduling algorithms has been presented by Chandak et al. [18] and presented a classification of scheduling algorithms in groups such as economic, meta-heuristic and population based etc.

A work presented in [19] developed three scheduling algorithms to schedule and reschedule the jobs on grid resources by considering dynamics of resources and applications. The three algorithms follow the incremental, divide and conquer and genetic algorithm principle. The study [19] suggests continuous monitoring of tasks in execution and resources

executing the scheduled tasks, if the performance is degraded rescheduling of tasks will be performed.

An adaptive scoring based job scheduling algorithm [20] has been proposed to schedule compute and data intensive independent tasks on grid resources. The status of grid resources is updated using the local and global scheduler before scheduling the new jobs but it does not reschedule the jobs already scheduled.

3 Scheduling Algorithms

In [2, 21, 22] we have proposed two scheduling algorithms- MH and MDQ. They are based on a fixed time quantum. The two flavors of MH have been introduced in [23] and two flavors of MDQ have been presented in [3]. In this paper we present a performance comparison of MH and MDQ scheduling algorithms on small scaled computational Grid using real workload traces.

3.1 Multilevel Hybrid Scheduling Algorithm (MH)

MH is based on master/slave architecture and uses the RR allocation strategy for job distribution among the slave processors; and the Hybrid Scheduling Algorithm (H) is used on each slave processor for computation.

For MH the ready queue is maintained in order of CPU burst length, with the least burst length at the head of the queue. Two numbers are maintained. The first number, t_{large}, shows the burst length of the largest process in the ready queue while the second one, t_{exec}, represents a running total of the execution time of all processes (since a reset was made). A new process submitted to the system is linked to the queue in accordance with its CPU burst length. MH dispatches processes from the head of the ready queue for execution by the CPU. Processes being executed are preempted on expiry of a time quantum, which is a system-defined variable. Following preemption, t_{exec} is updated as follows:

$$t_{exec} = t_{exec} + quantum \quad (1)$$

The numbers are then compared. If $t_{exec} < t_{large}$ then the preempted process is linked to the tail of the ready queue. The next process is then dispatched from the head of the ready queue. If $t_{exec} \geq t_{large}$ then the process with the largest CPU burst length is given a turn for execution. Upon preemption, the ready queue is sorted on the basis of SJF.

The value of t_{large} is reset to the burst length of the largest PCB, which is lying at the tail of the queue, and t_{exec} is reset to 0. The next process is then dispatched from the head of the ready queue. When a process has completed its task it terminates and is deleted from the system. t_{exec} is updated as follows:

$$t_{exec} = t_{exec} + time\ to\ complete \quad (2)$$

The numbers are then compared and the actions taken are the same as those for a preempted process. The two flavors of MHQ (i.e., Dynamic Multilevel Hybrid Scheduling Algorithm using Median (MHM) and Dynamic Multilevel Hybrid Scheduling Algorithm using Square Root (MHR)) and of MDQ (i.e., Dynamic Multilevel Dual Queue Scheduling Algorithm using Median (MDQM) and Dynamic Multilevel Dual Queue Scheduling Algorithm using Square Root (MDQR)) are discussed in [3, 23].

3.2 Multilevel Dual Queue Scheduling Algorithm (MDQ)

The MDQ algorithm operates similar to the MHQ where the only difference is maintaining two queues at ready queue. The ready queue comprises two queues – the waiting queue and the execution queue. The waiting queue is maintained as an FIFO queue. A new process submitted to the slave is linked to the tail of the waiting queue. Whenever the execution queue is empty, all processes in the waiting queue are moved to the execution queue, leaving the waiting queue empty. The execution queue is maintained in order of CPU burst length, with the shortest burst length at the head of the queue. The two numbers t_{large} and t_{exec} are maintained for MDQ as well and are updated using Eq. 1 and 2. If $t_{exec} < t_{large}$ then the preempted process is linked to the tail of the execution queue and next process is dispatched from the head of the execution queue. If $t_{exec} \geq t_{large}$ then the process with the largest CPU burst length is given a turn for execution. Upon preemption, all processes in the waiting queue are moved to the execution queue, leaving the waiting queue empty. The execution queue is then sorted on the basis of SJF. The value of t_{large} is reset to the burst length of the largest PCB and t_{exec} is reset to 0. The next process is then dispatched from the head of the execution queue.

4 Statistical Analysis of Real Workload Traces

[24] represents a comprehensive statistical analysis of real workload traces to study their dynamic behavior. Real workload traces i.e.; LCG1 were collected from the CERN. We reproduced the graphs of[24]using our developed web-based simulator ; SyedWSim [28]; to study the behavior of 'LCG1' workload[25, 26]. The total numbers of jobs in LCG1 is 188041. '64' second period is taken as job interval in our analysis. The number of jobs arriving in each interval is called 'job count'.

Figure 1 shows the division of job counts and run time processing demand for the real workload trace. Next we performed an autocorrelation of the job counts at different lags and then Fourier analysis, which are presented in Fig. 2.

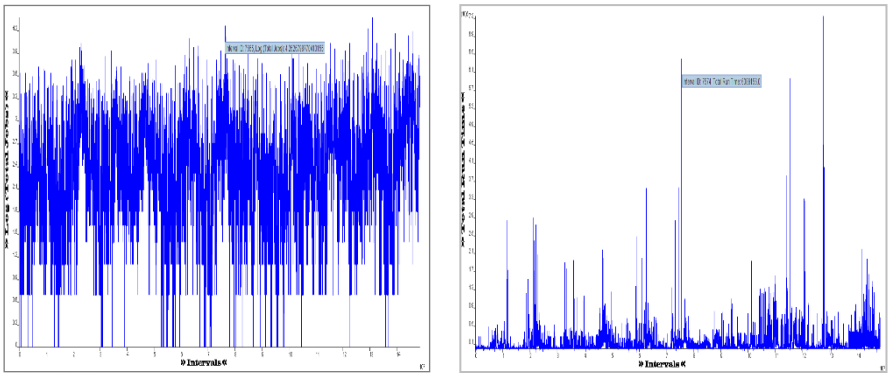

Fig. 1. The sequence plot and run time demand for the count process of LCG1

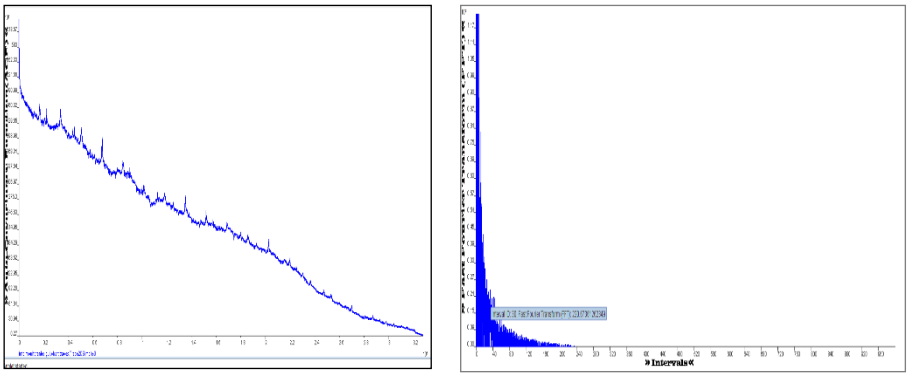

Fig. 2. The autocorrelation function(ACF) and Fast Fourier transformation(FFT) for the count process of LCG1

Fig 1 and 2 represent that job arrival show a diversity of correlation structures. Short range dependence, long range dependence, and pseudo periodicity are counted in analysis. If long range dependency is found in job arrival pattern then it predicts large performance degradation. Job pattern analysis is significant in evaluation of job scheduling algorithms.

Fig 1 and 2 demonstrate that Grid workload LCG1 shows scaling behavior and rich correlation structures, which are different from workload produced by the conventional parallel machines. Such types of behavior cannot be captured by simple models such as Poisson or other distribution based methods [24]. Fig 1 and 2 show that self-similarity and long range dependency are the characteristics of LCG1 jobs. LCG1 will be used in experiments for comparative performance analysis of proposed scheduling algorithms with other well-known job scheduling algorithms in truely Computational Grid environment.

5 Scheduling Simulator Design and Development

In this paper we used the same software development strategy for scheduling simulator, which was used in[2]. MPJ-express library is widely adopted Java API for parallel and distributed programming. This Java message passing library allows writing and implementing parallel applications for multicore and distributed systems.

We designed and developed a Java based simulator using MPJ-express API to evaluate the performance and efficiency of our proposed scheduling algorithms. The metadata for each process includes its Process ID, its arrival time and CPU demand. This simulator works on Master-Slave model. This simulator takes the number of slaves (processing elements) as input; and then the job is divided among slaves accordingly. The simulator considers the arrival time for each process and then submits processes to the system. The software has two main modules. One module runs on the master node (SimM). The other module runs on each slave processor (SimS).

A number of scheduling algorithms including the newly developed ones, MH, MHM, MHR, MDQ, MDQM and MDQR, as well as established ones, FCFS, SPN, SRTF, RR and P are programmed in the simulator. The user can select one from a range of scheduling algorithms as input. All slaves (compute nodes) use the same scheduling algorithm, which is

input by the user. The purpose of the simulator is to produce performance measures against each scheduling algorithm for given real workload trace LCG1.

6 Experimental Setup

The experiments made use of a Computational Grid of High Performance Cloud Computing Centre at Universiti Teknologi PETRONAS. We ran our experiment using a cluster of 16 to 32 processors. A detailed experimental setup is shown in Figure 3.

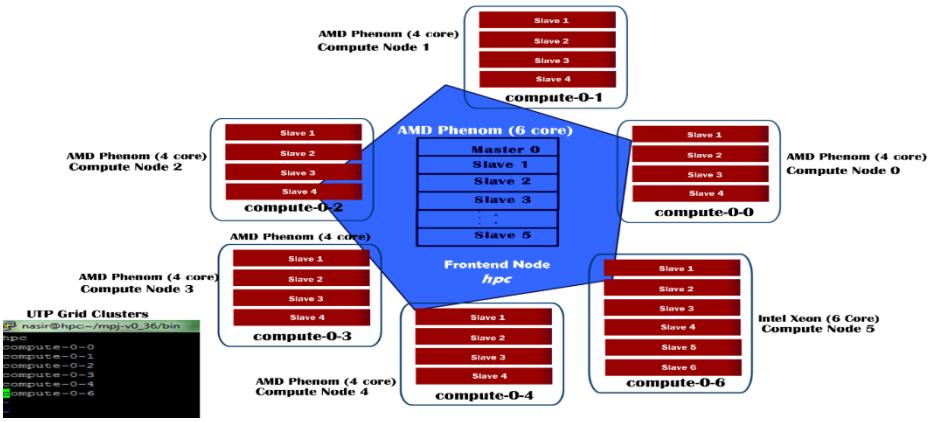

Fig. 3 Experimental Setup

7 Performance Analysis of Grid Scheduling Algorithms

This section presents the comparative performance analysis of scheduling algorithms using LCG1. Our experiments include the scalability test of scheduling algorithms under an increasing number of processors. The 'runtime' attribute is given for each process in 'LCG1'.The 'runtime' is taken as CPU time in our experiment. We used '5' units as the fixed time quantum. This section describes a comparative performance analysis of our proposed algorithms with the established ones.

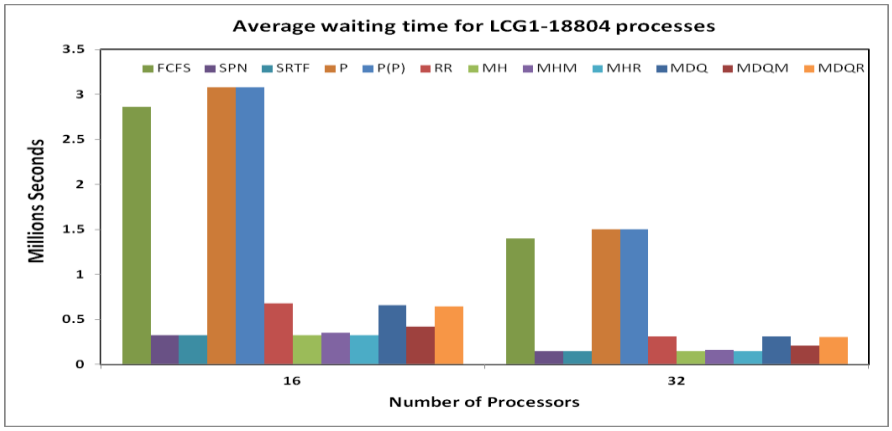

Fig. 4 Average Waiting Time

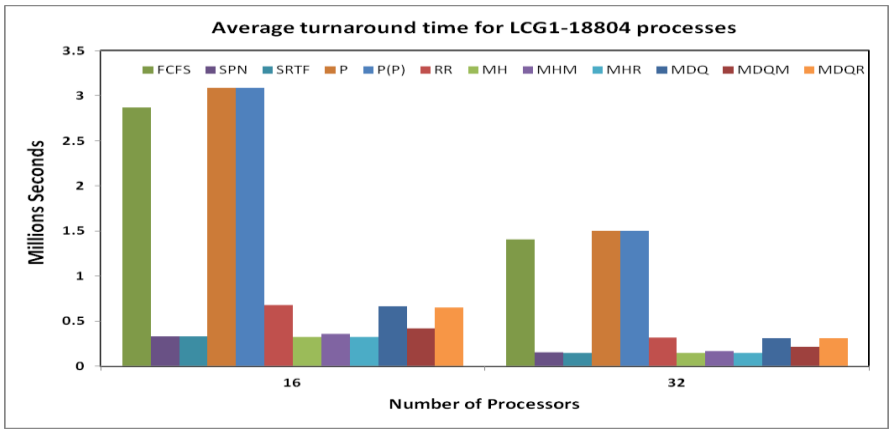

Fig. 5 Average Turnaround Time

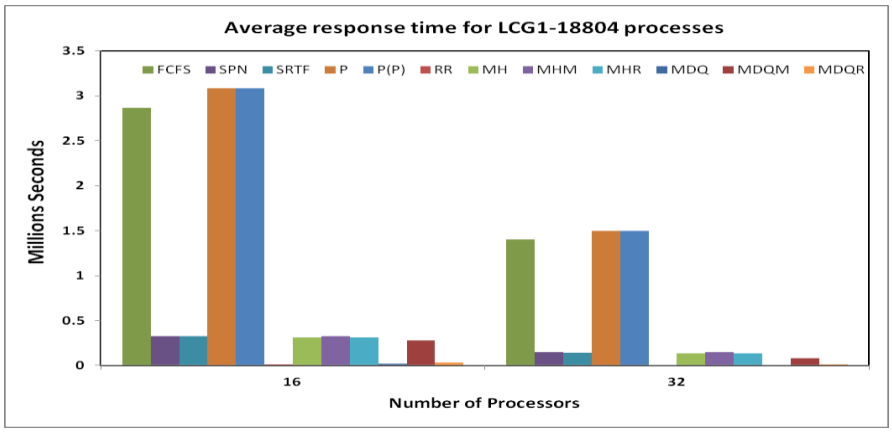

Fig. 6 Average Response Time

7.1 Average Waiting Times Analysis

Fig 4 depicts that the average waiting times calculated by each scheduling algorithm on Computational Grid for given real workload trace of LCG1. SRTF, MH, MHR and MDQM scheduling algorithms result in the least average waiting times as compared to the other scheduling algorithms. The average waiting time computed by SRTF scheduling algorithm is slightly shorter than the value computed by the MH and MDQR scheduling algorithms. Under the increased number of processing elements (CPUs), each algorithm shows the relative improvement in performance. MHM demonstrate better results as compared to the FCFS, RR and MDQ algorithms. All scheduling algorithms show that the relative performance is independent of the job nature, the job demand and the number of CPUs used for computation.

7.2 Average Turnaround Times Analysis

Figure 5 shows the pictorial view of the average turnaround times computed on Computational Grid for the scheduling algorithms using real workload traces of LCG1. The average turnaround times computed by the SRTF, MH and MDQM scheduling algorithms are less than the values computed for the other scheduling algorithms. Under the increased number of CPUs, each algorithm has shown improved average turnaround time, and supports scalability. Experimental results present that SRTF, MH and MHR are at the same performance level in terms of average turnaround times. Figure 5 also shows that the average turnaround times computed for MHM and MDQM are better than the values computed for the MDQ, RR and FCFS but slightly longer than those for the MHR and SJF scheduling algorithms. Moreover, all scheduling algorithms show that the relative performance, which is independent of the job nature, the job size and the number of CPUs used for the computation.

7.3 Average Response Times Analysis

Figure 6 presents that MDQ and MDQR result in the least average response times as compared to the MH and MHR scheduling algorithms. The average response times computed by MDQ are slightly longer than the values computed for RR and slightly shorter than those for MH and MHR. The SPN and SRTF scheduling algorithms show poor response times as compared to all other scheduling algorithms. Moreover, all scheduling algorithms present that the relative performance is independent of the nature of jobs, the job size and the number of CPUs used in experimentation. MDQ gives consistently good response time measures for given workload of LCG1 under increased numbers of processors.

8 Conclusion

In this paper we performed comparative performance analysis of job scheduling algorithms on campus based Computational Grid using LCG1 real workload traces. Our proposed algorithms introduce new dynamic time quantum strategy. Proposed algorithms generate the time quantum based on nature of jobs and execute the processes accordingly. We evaluated our proposed and existing job scheduling algorithms on a simulator running on Computational Grid using LCG1 workload traces under the increased number of CPUs. Statistical analysis of LCG1 workload traces is also conducted in this work to study the dynamic nature of jobs.

We conclude that MH and MDQM are scheduling policies from the system perspective; they satisfy the system requirements (i.e. less Average Waiting Time and less Turnaround Time). MDQ and MDQR are scheduling policies from the user perspective due to their shorter Average Response Time. Moreover, proposed MH, MDQR, MDQM and MDQ are scalable, i.e. the relationship between each performance measure (e.g. average turnaround time) and the workload size is almost linear.

References

[1] D. Fernández-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, pp. 1427-1436, 1989.

[2] S. N. M. Shah, A. K. B. Mahmood, and A. Oxley, "Development and Performance Analysis of Grid Scheduling Algorithms," in *International Conference on Advances in Information Technology*, 2009, pp. 170-181.

[3] S. N. M. Shah, A. K. B. Mahmood, and A. Oxley, "Dynamic multilevel dual queue scheduling algorithms for grid computing," in *International Conference on Software Engineering and Computer Systems*, 2011, pp. 425-440.

[4] S. Haines, *Pro Java EE 5 Performance Management and Optimization*: Apress, 2006.

[5] H. Li and R. Buyya, "Model-driven simulation of grid scheduling strategies," in *e-Science and Grid Computing, IEEE International Conference on*, 2007, pp. 287-294.

[6] M. R. Garey and D. S. Johnson, *Computers and intractability* vol. 29: wh freeman New York, 2002.

[7] F. Xhafa and A. Abraham, "Computational models and heuristic methods for Grid scheduling problems," *Future generation computer systems*, vol. 26, pp. 608-621, 2010.

[8] H. Izakian, A. Abraham, and B. T. Ladani, "An auction method for resource allocation in computational grids," *Future Generation Computer Systems*, vol. 26, pp. 228-235, 2// 2010.

[9] K. Z. Gkoutioudi and H. D. Karatza, "Multi-criteria job scheduling in grid using an accelerated genetic algorithm," *Journal of Grid Computing*, vol. 10, pp. 311-323, 2012.

[10] Y.-H. Lee, S. Leu, and R.-S. Chang, "Improving job scheduling algorithms in a grid environment," *Future generation computer systems*, vol. 27, pp. 991-998, 2011.

[11] M.-Y. Tsai, P.-F. Chiang, Y.-J. Chang, and W.-J. Wang, "Heuristic Scheduling Strategies for Linear-Dependent and Independent Jobs on Heterogeneous Grids," in *Grid and Distributed Computing: International Conference, GDC 2011, Held as Part of the Future Generation Information Technology Conference, FGIT 2011, Jeju Island, Korea, December 8-10, 2011. Proceedings*, T.-h. Kim, H. Adeli, H.-s. Cho, O. Gervasi, S. S. Yau, B.-H. Kang, *et al.*, Eds., ed Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 496-505.

[12] H. Topcuoglu, S. Hariri, and M.-y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE transactions on parallel and distributed systems*, vol. 13, pp. 260-274, 2002.

[13] D. M. Abdelkader and F. Omara, "Dynamic task scheduling algorithm with load balancing for heterogeneous computing system," *Egyptian Informatics Journal*, vol. 13, pp. 135-145, 2012.

[14] H. Hussain, S. U. R. Malik, A. Hameed, S. U. Khan, G. Bickler, N. Min-Allah, *et al.*, "A survey on resource allocation in high performance distributed computing systems," *Parallel Computing*, vol. 39, pp. 709-736, 2013.

[15] K. Etminani and M. Naghibzadeh, "A min-min max-min selective algorihtm for grid task scheduling," in *Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia on*, 2007, pp. 1-7.

[16] W. Zheng and R. Sakellariou, "Budget-deadline constrained workflow planning for admission control," *Journal of grid computing*, vol. 11, pp. 633-651, 2013.

[17] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Scientific Programming*, vol. 14, pp. 217-230, 2006.

[18] A. V. Chandak, B. Sahoo, and A. K. Turuk, "Heuristic task allocation strategies for computational grid," 2011.

[19] H. Sanjay and S. S. Vadhiyar, "Strategies for rescheduling tightly-coupled parallel applications in multi-cluster grids," *Journal of Grid Computing*, vol. 9, pp. 379-403, 2011.

[20] R.-S. Chang, C.-Y. Lin, and C.-F. Lin, "An adaptive scoring job scheduling algorithm for grid computing," *Information Sciences*, vol. 207, pp. 79-89, 2012.

[21] S. N. M. Shah, A. K. B. Mahmood, and A. Oxley, "Hybrid scheduling and dual queue scheduling," in *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*, 2009, pp. 539-543.

[22] S. N. Mehmood Shah, A. K. B. Mahmood, and A. Oxley, "Analysis and evaluation of grid scheduling algorithms using real workload traces," in *Proceedings of the International Conference on Management of Emergent Digital EcoSystems*, 2010, pp. 234-239.

[23] S. N. M. Shah, A. K. B. Mahmood, and A. Oxley, "Dynamic multilevel hybrid scheduling algorithms for grid computing," *Procedia Computer Science*, vol. 4, pp. 402-411, 2011.

[24] H. Li, "Workload dynamics on clusters and grids," *The Journal of Supercomputing*, vol. 47, pp. 1-20, 2009.

[25] (24 Oct 2011). *Worldwide LHC Computing Grid*. Available: http://lcg.web.cern.ch/lcg/

[26] (25 December 2016). *Trace analysis report LCG*. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-11-lcg/report/