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

Conference Paper · January 2017

DOI: 10.4108/eai.27-2-2017.152264

4 authors:

Syed Nasir Mehmood Shah
Universiti Teknologi PETRONAS

Ahmad Kamil Mahmood
Universiti Teknologi PETRONAS

Saddaf Rubab
National University of Sciences and Technology

Mohd Fadzil Hassan
Universiti Teknologi PETRONAS

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

[1,*]dr.shah@kicsit.edu.pk, [2]kamilmh@petronas.com.my, [3]saddaf_g02754@utp.edu.my, [4]mfadzil_hassan@petronas.com.my

Abstract. Grid, an infrastructure for resource sharing, currently has shown its importance in many scientific applications requiring tremendously high computational power. Grid computing enables sharing, selection and aggregation of resources for solving complex and large-scale scientific problems. 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 the comparative performance analysis of our proposed job scheduling algorithm with other well known job scheduling algorithms considering the quality of service parameters like waiting time, turnaround time and response. The main thrust of this work was to conduct a quality of service based experimental performance evaluation of job scheduling algorithms on a computational Grid in true dynamic environment. Experimental evaluation confirmed that the proposed scheduling algorithms possess a high degree of optimality in performance, efficiency and scalability. This paper also includes a 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 grid resource management. There are three main phases of Grid scheduling. Phase one is resource discovery, which provides a list of available resources. Phase two is resource allocation, which involves the selection of feasible resources and the mapping of jobs to the resources. The third phase includes job execution. In the second phase, the selection of the best match of jobs to resources is an NP-complete problem [1, 2]

In our previous research work, we proposed two Grid scheduling algorithms (the Multilevel Hybrid Scheduling Algorithm (MH) and the Multilevel Dual Queue Scheduling Algorithm (MDQ) with a view to improved performance [3]. We also presented dynamic flavors of MH and MDQ to manage scheduling of Grid jobs in dynamic and scalable computational Grid environment in our recent work [4, 5].

Scalability testing is a significant success factor in the design and development of a Grid scheduling algorithm. Scalability means a measure of optimizing the application to use more processing power given to it in the form of additional processors or cores. In [6], the authors defined the concept of performance and scalability as "The terms 'performance' and 'scalability' are commonly used interchangeably, but the two are distinct: performance measures the speed with which a single request can be executed, while scalability measures the ability of a request to maintain its performance under increasing load."

Two fundamental issues have to be considered for the performance evaluation of new Grid scheduling algorithms. Firstly, representative workload traces are required to produce dependable results. Secondly, a good testing environment should be set up, which is the commonly done by simulations mostly [7].

Grid scheduling presents several challenges that make the implementation of practical systems a very difficult problem. The job scheduling on grid resources is an NP-hard problem [8]. The near optimal solution for job scheduling has been presented as a heuristic in the literature [9-13]. 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) [14, 15] 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 [11]. 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[13, 16] , Min-Min[13, 16, 17], BHEFT [18] and GA[11, 19]. A comparison of many scheduling algorithms has been presented by Chandak et al. [20] and presented a classification of scheduling algorithms in groups such as economic, meta-heuristic and population based etc.

A work presented in [21] 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 [21] 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 [22] 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 [3, 23, 24] 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 [4] and two flavors of MDQ have been presented in [5]. 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:

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

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 [4, 5].

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

In [25] a comprehensive statistical analysis has been carried out for a variety of workload traces on clusters and Grids. We reproduced the graphs of [25] to study the behavior of the dynamic nature of workload 'LCG1' [26, 27], using SyedWSim [28]. The total numbers of jobs in LCG1 is 188041. We looked at the number of jobs arriving in each 64 second period. The number of jobs arriving in a particular period is its 'job count'.

Figure 1 shows the distribution of job counts and run time demand for the whole trace. Next we performed an autocorrelation of the job counts at different lags followed by Fourier analysis, presented in Fig. 2.

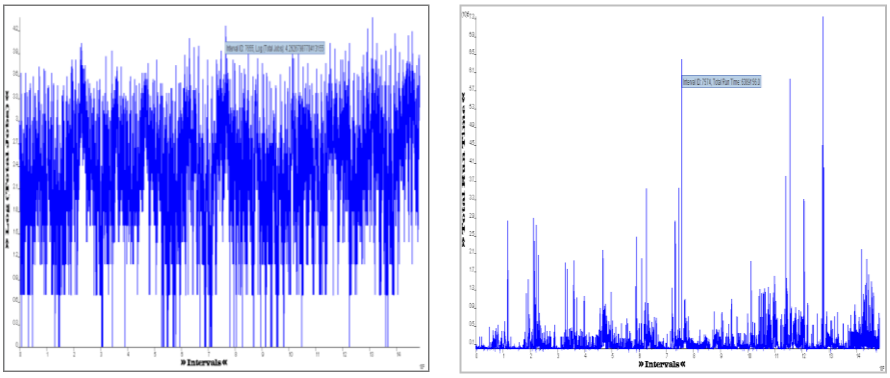

Fig. 1. The sequence plot and run time demand for the count process of LCG1

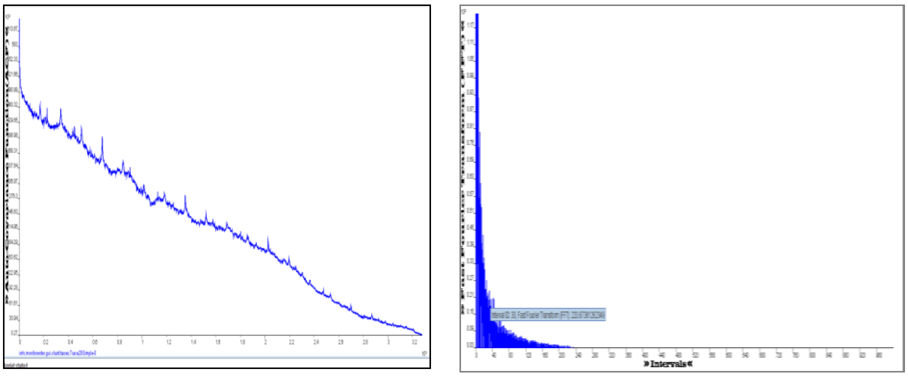

Fig. 2. The autocorrelation function(ACF) and Fast Fourier transformation(FFT) for the count process of LCG1

Fig 1 and 2 depicts that job arrival show a diversity of correlation structures, including short range dependence, pseudo periodicity, and long range dependence. Long range dependencies can results in a large performance degradation whose effects should be taken into consideration for evaluation of scheduling algorithms.

Grid workload LCG1 is shown to have rich correlation and scaling behavior, which are different from conventional the parallel workload and cannot be captured by simple models such as Poisson or other distribution based methods [25]. Self-similarity and long range dependency are the characteristics of LCG1 jobs. LCG1 will play a key role in the performance evaluation of our proposed scheduling algorithms in comparison to other well-known Grid scheduling algorithms.

5 Scheduling Simulator Design and Development

In this paper we used the same development strategy as we discussed in [5]. The MPJ-express is widely used Java message passing library that allows writing and executing parallel applications for distributed and multicore systems. We developed a

Java based simulator using MPJ-express API to evaluate the efficiency of our proposed scheduling algorithms. The metadata for each process includes its ID, its arrival time, its CPU time and the number of slaves that the job is to be divided between. The simulation software encounters the arrival time for each process and then submits processes to the system. The software has two main programs. One program runs on the master node (SimM). The other program runs on each slave processor (SimS).

All slaves use the same scheduling algorithm, which is input by the user of SimM. The user can select one of a range of algorithms including the newly developed ones, MHM, MHR, MH, MDQ, MDQM and MDQR, as well as established ones, FCFS, SPN, SRTF, RR and P. The purpose of the simulator is to produce a comparative performance analysis of scheduling algorithms.

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.

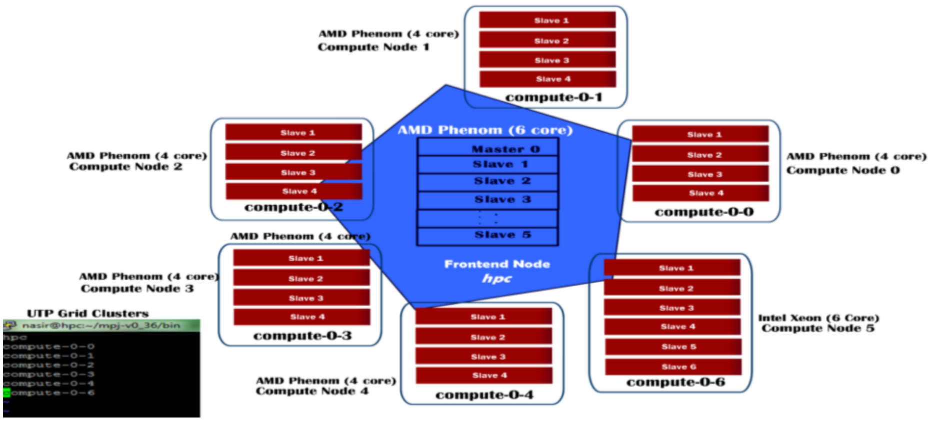

Fig. 3 Experimental Setup

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.

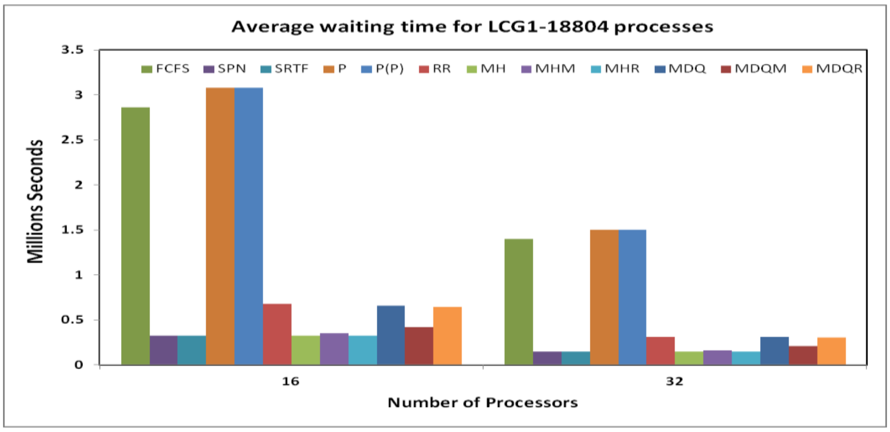

Fig. 4 Average Waiting Time

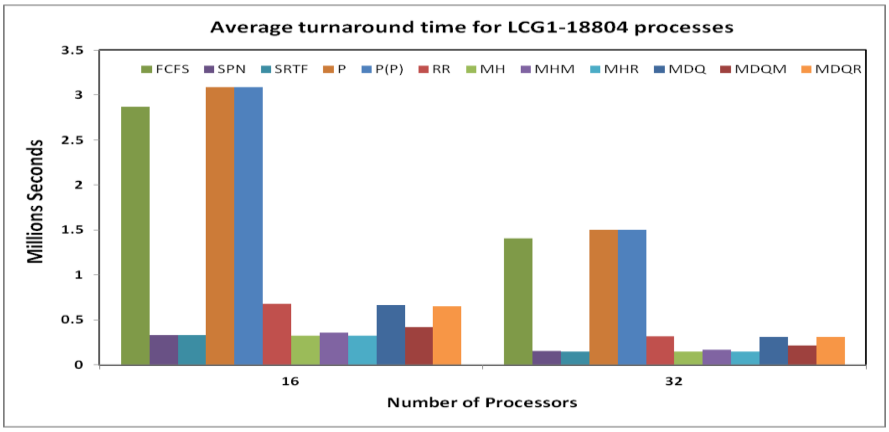

Fig. 5 Average Turnaround Time

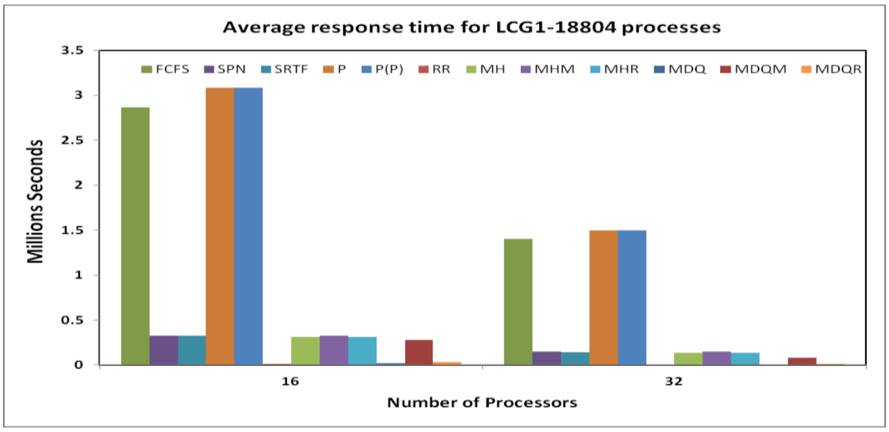

Fig. 6 Average Response Time

7.1 Average Waiting Times Analysis

Fig 4 shows that the average waiting times computed by each scheduling algorithm for each real workload trace. Figure 4 illustrates that the SRTF, MH, MHR and MDQM scheduling algorithms produce the shortest average waiting times as compared to the other scheduling algorithms. The average waiting time computed for SRTF is slightly shorter than the value computed for the MH and MDQR scheduling algorithms. By increasing the number of CPUs, each algorithm shows the relative improvement in performance, except for the MHM algorithm. MHM shows better results as compared to the MDQ, RR and FCFS algorithms. All scheduling algorithms, with the exception of MHM, show that the relative performance is independent of the nature of the workload, the workload size and the number of CPUs used for computation.

7.2 Average Turnaround Times Analysis

Figure 5 presents the pictorial view of the average turnaround times computed for the scheduling algorithms using real workload traces. Figure 5 illustrates that the average turnaround time computed by the SRTF, MH and MDQM scheduling algorithms are shorter than the other Grid scheduling algorithms. By increasing the number of CPUs, each algorithm has an improved average turnaround time, except for the MHM scheduling algorithm. Experimental results show that SRTF, MH and MHR are at the same performance level as regards average turnaround time. Figure 5 also shows that the average turnaround times computed for MHM and MDQM are slightly longer than those for the MHR and SJF scheduling algorithms but better than the values computed for the MDQ, RR and FCFS scheduling algorithms. Moreover, all scheduling algorithms, with the exception of MHM, show that relative performance is independent of the nature of the workload, the workload size and the number of CPUs used in the experiment.

7.3 Average Response Times Analysis

Figure 6 shows that MDQ and MDQR produce the shortest average response times as compared to the MH and MHR scheduling algorithms. The average response times computed for MDQ are slightly longer than those for RR and slightly shorter than those for MH and MHR. The SPN and SRTF scheduling algorithms result in poor response times as compared to the other scheduling algorithms. All scheduling algorithms, except for MHM, show that the relative performance is independent of the nature of the workload, the workload size and the number of CPUs. MDQ gives consistently good results for different workloads and numbers of CPUs.

8 Conclusion

In this paper we present performance comparisons of job scheduling algorithms using LCG1 real workload traces on Computational Grid. Our proposed algorithms compute the time quantum dynamically and execute the processes accordingly. We have evaluated these algorithms on a simulator running on Computational Grid using a wide range of CPUs and 'LCG1' workload traces. In this paper we also performed a statistical analysis of 'LCG1' workload traces to study the dynamic nature of jobs.

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

References

1. Roehrig, M., W. Ziegler, and P. Wieder, Grid scheduling dictionary of terms and keywords. GFD-I. 11, Grid Scheduling Dictionary WG (SD-WG), 2002.
2. Fernández-Baca, D., Allocating modules to processors in a distributed system. IEEE Transactions on Software Engineering, 1989. 15(11): p. 1427-1436.
3. Shah, S.N.M., A.K.B. Mahmood, and A. Oxley. Development and Performance Analysis of Grid Scheduling Algorithms. in International Conference on Advances in Information Technology. 2009. Springer.3.
4. Shah, S.N.M., A.K.B. Mahmood, and A. Oxley, Dynamic multilevel hybrid scheduling algorithms for grid computing. Procedia Computer Science, 2011. 4: p. 402-411.
4. Shah, S.N.M., 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. Springer.
6. Haines, S., Pro Java EE 5 Performance Management and Optimization. 2006: Apress.
7. Li, H. and R. Buyya. Model-driven simulation of grid scheduling strategies. in e-Science and Grid Computing, IEEE International Conference on. 2007. IEEE.
8. Garey, M.R. and D.S. Johnson, Computers and intractability. Vol. 29. 2002: wh freeman New York.
9. Xhafa, F. and A. Abraham, Computational models and heuristic methods for Grid scheduling problems. Future generation computer systems, 2010. 26(4): p. 608-621.
10. Izakian, H., A. Abraham, and B.T. Ladani, An auction method for resource allocation in computational grids. Future Generation Computer Systems, 2010. 26(2): p. 228-235.
11. Gkoutioudi, K.Z. and H.D. Karatza, Multi-criteria job scheduling in grid using an accelerated genetic algorithm. Journal of Grid Computing, 2012. 10(2): p. 311-323.
12. Lee, Y.-H., S. Leu, and R.-S. Chang, Improving job scheduling algorithms in a grid environment. Future generation computer systems, 2011. 27(8): p. 991-998.

13. Tsai, M.-Y., et al., 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, et al., Editors. 2011, Springer Berlin Heidelberg: Berlin, Heidelberg. p. 496-505.
14. Topcuoglu, H., S. Hariri, and M.-y. Wu, Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE transactions on parallel and distributed systems, 2002. 13(3): p. 260-274.
15. Abdelkader, D.M. and F. Omara, Dynamic task scheduling algorithm with load balancing for heterogeneous computing system. Egyptian Informatics Journal, 2012. 13(2): p. 135-145.
16. Hussain, H., et al., A survey on resource allocation in high performance distributed computing systems. Parallel Computing, 2013. 39(11): p. 709-736.
17. Etminani, K. 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. IEEE.
18. Zheng, W. and R. Sakellariou, Budget-deadline constrained workflow planning for admission control. Journal of grid computing, 2013. 11(4): p. 633-651.
19. Yu, J. and R. Buyya, Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. Scientific Programming, 2006. 14(3-4): p. 217-230.
20. Chandak, A.V., B. Sahoo, and A.K. Turuk, Heuristic task allocation strategies for computational grid. 2011.
21. Sanjay, H. and S.S. Vadhiyar, Strategies for rescheduling tightly-coupled parallel applications in multi-cluster grids. Journal of Grid Computing, 2011. 9(3): p. 379-403.
22. Chang, R.-S., C.-Y. Lin, and C.-F. Lin, An adaptive scoring job scheduling algorithm for grid computing. Information Sciences, 2012. 207: p. 79-89.
23. Shah, S.N.M., 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. IEEE.
24. Shah, S.N.M., 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. ACM.
25. Li, H., Workload dynamics on clusters and grids. The Journal of Supercomputing, 2009. 47(1): p. 1-20.
26. Worldwide LHC Computing Grid. 24 Oct 2011]; Available from: http://lcg.web.cern.ch/lcg/
27. Trace analysis report LCG. 25 May 2015]; Available from: http://gwa.ewi.tudelft.nl/datasets/gwa-t-11-lcg/report/.
28. Shah, S.N.M., A.K.B. Mahmood, and A. Oxley, SYEDWSIM: A Web Based Simulator for Grid Workload Analysis, in International Conference on Software Engineering and Computer Systems. 2011. Springer.