Scheduling Grid Jobs Using Priority Rule Algorithms and Gap Filling Techniques

Zafril Rizal M Azmi[1], Kamalrulnizam Abu Bakar[2], Mohd Shahir Shamsir[3], Wan Nurulsafawati Wan Manan[4], Abdul Hanan Abdullah[5]

[1,4]Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang
[2,5]Faculty of Computer Science and Information System, Universiti Teknologi Malaysia
[3]Faculty of Biosciences and Bioengineering, Universiti Teknologi Malaysia

zafril@ump.edu.my, kamarul@fsksm.utm.my, shahir@fbb.utm.my, safawati@ump.edu.my, hanan@fsksm.utm.my

Abstract

Over the past decade, scheduling in grid computing system has been an active research. However, it is still difficult to find an optimal scheduling algorithm in order to achieve load balancing. Gap filling or backfilling is one of the techniques used to optimize First Come First Serve (FCFS) and have been used widely in scheduling area. This paper introduced an improved backfilling technique which not only applied to FCFS but to others Priority Rule algorithms. Experimental results have shown significant improvement compared to the original Priority Rule algorithms.

Keywords: *Grid scheduling, priority rule algorithms, backfilling, performance.*

1. Introduction

Grid computing can have variety of heterogeneous resources connected with each other. However, the available resources such as the network and computational power are continually changing with respect to every available node. The constantly changing characteristic of the heterogeneous resources is known as dynamic resources. To make it more critical, the jobs that need to be process by these resources are arriving from different length of time and not knowing by the system until the particular jobs arrive to the system.

In order to utilize these dynamic resources and jobs optimally, a scheduling strategy should be able to continually adapt to the changes and properly distribute the workload and data amounts scheduled to each node. Unfortunately, several researchers have stated that serious difficulty in concurrent programming of a grid computing has occurred in terms of dealing with scheduling and load balancing of such a system, which may consist of heterogeneous computers [1, 2]. The current popular techniques in grid scheduling are schedule-based algorithms that adapting optimization technique such as Genetic Algorithm, Ant Colony Optimization and Tabu Search. The corresponding authors of these techniques have proof that their techniques are much better compared to the priority rules algorithms. Priority rules scheduling has been widely used in the popular scheduling production systems such as Condor and PBS.

Although schedule-based algorithms were proven to be more effective (based on objective the researchers want to achieve), priority rules techniques are still very important because most of the schedule-based algorithms will have to apply priority rules algorithms for initial schedule in their schedule-based algorithms.

Each priority rule algorithms have their own strength. For example, Shortest Job First (SJF) is optimal for minimizing flowtime. Similarly, Longest Job First (LJF) and Minimum Time To Due Date (MTTD) are optimal for makespan and total tardiness respectively. However, each of these algorithms has its own drawbacks. SJF might have a very poor makespan while LJF may suffer from extremely high flowtime, total tardiness and low resource utilization. To strike a balance between the pro and the cons, this paper proposed a combination of priority rule algorithms with backfilling or gap filling technique.

2. Priority Rule Algorithms

Priority rules also referred as Queue-based [3]. Instead of guaranteeing optimal solution, these techniques aim to find reasonable solutions in a relatively short time. Although it is a suboptimal algorithms, it is yet the most frequently used for solving scheduling problem in real world because of the easiness to implement and their low time complexity. In this study, we used six priority rules algorithms

2.1 First Come First Serve (FCFS)

First Come First Serve (FCFS) or also known as First In First Out (FIFO) is the simplest and the most fundamental of grid scheduling that involves client-server interaction. In grid scheduling, FCFS policy manages the jobs based on their arrival time, which means that the first job will be processed first without other biases or preferences. This concept has been used by several well known enterprise scheduler such as MAUI [4] and PBS [5].

2.2 Earliest Deadline First (EDF)

Earliest Deadline First (EDF) is a policy that schedule all the incoming jobs according to the specified due date or deadline. Incoming jobs will be processed or put in the queue based on the chronology indicate by the deadline. The job with the earliest deadline will be placed first in the processing queue.

2.3 Shortest Job First (SJF)

Shortest job First (SJF) also known as Shortest Job Next (SJN) or Shortest Process Next (SPN) is a scheduling technique that selects the job with the smallest execution time. The jobs are queued with the smallest execution time placed first and the job with the longest execution time placed last and given the lowest priority. In theory, the best strategy to minimize the response time is to schedule the shortest job on the fastest resource [6]. Since this policy gives preference to some groups of jobs over other group of jobs, this policy is unfair when compared to FCFS policy. In extreme cases, when jobs with shorter execution time continue to arrive, jobs with longer execution period may never get a chance to execute and would have to wait indefinitely. This is known as ‘starvation’ and would pose a serious problem and reflect the low degree of fairness for this policy [7]. In addition, SJF is believed to have the maximum makespan time compared to other algorithms because of this characteristic.

2.4 Longest Job First (LJF)

Longest Job First (LJF) have the contradiction behavior of SJF. While shortest job is believe will reduce the response time, processing longest job first on the fastest resource according to Abraham in [6] will minimize the makespan time. However, LJF will be suffering due to slightly increase in the response time.

2.5 Earliest Release Date (ERD)

Earliest Release Date (ERD) put the highest priority to the job that has the earliest release date in the queue. Release date is the start time of each and every job and it can be different or same. If there are two or more jobs that have the same release date, FCFS rule will be applied. Studies have also shown that if there is only a few numbers of jobs in the queue, the ERD performance will be similar to FCFS but when the number of jobs increases, the results will defer.[8].

2.6 Minimum Time to Due Date (MTTD)

Minimum Time to Due Date (MTTD) is a scheduling algorithm which put the priority on the jobs according to the time that can be considered for the job to be executed with minimum tardiness [8]. To achieve this objective, MTTD define the time as follow:

(Deadline-Release Date) (1)

3. Backfilling Strategies

Gap or Hole filling technique was originally developed from the backfilling algorithm introduced in EASY [9]. The purpose of backfilling is to improve system utilization of scheduler that used First Come First Serve (FCFS). Although FCFS is a simple policy and have been widely used, it suffers from the low system utilization [10]. This happens because there is a gap between two jobs that make the resource idle [11] Backfilling improves resource utilization by allowing small job to fill in those gaps. Job which is lower in the queue is moved to the idle machines without delaying the execution of the job at the top of the queue. Many backfilling variants have been suggested, but most of them consider jobs candidate both for execution and for backfilling according to a FCFS strategy [12]. This paper will try to extend this technique into several other PR algorithms other than FCFS.

Originally, backfill such as EASY utilize the system by moving small job. However there also some research that combined backfill with other PR algorithms. Mishra et al., in [13] have successfully implement SJF-Backfill scheduling algorithms. The algorithms works by rearrange existing jobs queue based on increasing order of the execution time of the jobs. Candidate job to be moved forward is the one that is also in the queue. Similarly, works conducted by Dan et al., in [11] also combined SJF with backfill but with different approach. While keep preserving the order using FCFS, the SJF only applied to the job that want to be moved. These two technique and several other backfill-PR algorithms have one common which is computationally expensive. This is because scheduler has to reconstruct new queue each time new job arrive to the system.

Contradict to the above technique, Klusacek et al., in [14] have come out with a solution for the problem by proposing an incremental technique for the backfilling scheduler. It works by using the last computed schedule as a starting point for building a new and up to date schedule. This technique avoids unreasonable cost from building a schedule from scratch. The proposed algorithm named Earliest Gap-Earlier Deadline First (EG-EDF) utilize EDF algorithm using the backfill technique. There are five important steps in the EG-EDF algorithm which are:

Step 1: Determines which particular machine is suitable to execute the job.

Step 2: Places new job to the particular suitable machines schedule.

Step 3: Check the current machines schedule whether a suitable gap for the new job exists.

Step 4: If more than one suitable gap, use the earliest one (EG policy).

Step 5: Evaluate the new schedule (acceptance criterion). If the new schedule better than current schedule, accept.

4. Earliest Gap-Priority Rule (EG-PR) Algorithms

The proposed algorithms introduced in this section which are EG-FCFS, EG-SJF, EG-LJF, EG-ERD and EG-MTTD were designed by combining backfilling like procedure with priority rule algorithms. The procedure were adapt from the EG-EDF [14] mentioned earlier. The formal algorithm for the proposed scheduler is given in Algorithm 1.

EG-PR scheduler is designed to manage newly arrived jobs submitted by the grids user to the grid system. The incoming new jobs are sorted using First Come First Serve (FCFS) in the *Waiting Queue*. This *raw* queue then is checked whether the first job in the queue can fit in into the first gap found in the *machine$_n$ schedule*. Since the gap only appear in the queue when more than one job exist at the same cycle time, the priority rules will always be the initial scheduler used to allocate the job to selected *machine$_n$ schedule*. However, if there exists even one gap that can be fit in by a new job, the backfilling approach will be used. Unlike traditional method, this backfilling algorithm considers not only small jobs but apply to all new job that entering the system.

The mechanism used to evaluate each decision is based on the makespan and prediction of job that meet the deadline objective functions [15]. To achieve this, the weight for both objectives is calculated and if the *total weight* is more then *0*, the current *job-machine* mapping is considered the best schedule so far, compared to the previous match.

Algorithm 1: Backfilling- Priority rules

```
1  resourceSize = Get total number of resource in the grids
2  for i=0 to resourceSize do
3   if number of PU < number of PU requested by $job_m$ then
4    break operation
5   else $machine_i$ is suitable to perform $job_m$
6   if there exist suitable gap in $machine_i$ schedule then
7    insert $job_m$ into $machine_i$ schedule
8   end if
8   else if no suitable gap in $machine_i$ schedule then
9    insert $job_m$ into $machine_i$ schedule based on selected priority rule algorithms
     (FCFS/SJF/LJF/ERD/MTTD)
10  end if
11  total weight=weight for makespan + weight for jobs that meet deadline
12  if total weight < 0.0 then
13   remove $job_m$ from $machine_i$ schedule
```

```
14 else
15   put job_m into machine_i schedule
16 end if
17 end for
```

5. Biggest Hole-Priority Rule (BG-PR) Algorithms

In [14], new algorithm namely Earliest Gap Earliest Deadline First (EG-EDF) that select the first gap in the scheduler to be fill by a new job have been introduced. Although the EG-EDF algorithms have been proof to outperform the backfilling algorithm and several other policy mentioned in [14], it is still open for further improvement.

In order to improve the *Earliest Gap* technique introduced in EG-EDF [14] and also fully manipulate the gap filling technique, this paper has introduced Biggest Hole (BH) strategy. One of the main ideas of Grid Computing is to make a large process faster [16]. Large process is the process that usually involved many calculations and huge data that consumes a lot of time [16]. While traditional backfilling intends to fill the holes with smaller jobs and EG-EDF fill the first holes with possible jobs that can fix in, Biggest Hole search for the biggest holes in the queue and match them with any jobs that can fix in. In the case of EG-EDF, if the hole or gap does not match the jobs or too small for the job, the system has to search again and the possibility of the candidate gap to be lost are higher because as time pass by, the candidate gap can be shrinks smaller or even lost from the system. This is not happening with BH technique. BH-PR carries out the same procedure as EG-PR algorithms. However in line 6 of Algorithm 1, instead of accepting the earliest gap in the queue, BH-PR search for the biggest gap in the *machine$_i$ schedule* at that particular time. This is carried out by sorting the gap decreasingly based on the gap size as shown in Algorithm 2 (sorting the gap ID based on the gap size).

Algorithm 2 Algorithm to Find the Biggest Hole (Sorting Algorithm)

```
1  if total hole is more than 1 which mean more than 1 gap detected then
2    h1 = first hole (hole_0)
3    for i=1 to holes size do
4      get the next hole (h2=hole_i)
5      if  h1.size>h2.size then
6        x=h2
7        h2=h1
8        h1=x
9        i++
10       end if
11    end for
12 end if
```

6. Experimental Results

This section will have three subsection presenting and discussing results obtained from the experimentations. The first part will cover results from experimentation using EG-PR scheduler. The second subsection will focused on the comparison of EG-PR with original PR under heavy load grid system. Finally the comparison of EG-PR with BH-PR also under heavy load grid system will be discussed.

These experiments considered six different PR algorithms namely First Come First Serve (FCFS), Shortest Job First (SJF), Longest Job First (LJF), Earliest Release Date (ERD), Minimum Time to Deadline (MTTD) and Earliest Deadline First (EDF) as mentioned earlier. The experiment focused on 5 performance metrics; total tardiness, makespan time, flowtime, number of delayed jobs and the resource utilization.

Minimizing Total Tardiness: One of the main objectives of the scheduling procedure is the completion of all jobs before their agreed due dates. Failure to keep that promise has negative effects on the credibility of the service provider. If we define lateness of job *i* as the difference between its completion time C_i and the corresponding due date d_i, then the tardiness of the job is calculated from the following expression:

$$T_i = max(0, C_i - d_i) \qquad (2)$$

In other words, tardiness represents the positive lateness of a job. In a single machine environment, we define the total tardiness problem as follows:

A number of jobs $J_1, J_2 ...., J_n$ are to be processed in a single facility. Each job is available for processing at time zero, and is completely identified by its processing time *p*, and its due date d_i. We seek to find the processing sequence that minimizes the sum of tardiness of all jobs:

$$\sum_{i=1}^{n} max(0, Ci - di) \qquad (3)$$

Where C_i is the completion time of job *i*. The total tardiness problem is a special case of the weighted total tardiness problem. Both problems are not easy to solve, especially for large values of *n*.

Minimizing Makespan Time: Makespan is a standard performance metric to evaluate scheduling algorithms. Small values of makespan mean that the scheduler is providing good and efficient planning of jobs to resources. The makespan of a schedule can be defined as the time it takes from the instant the first task begins execution to the instant at which the last task completes execution [17]. In a simplest words, makespan is the time when finishes the latest job.

Minimizing Response Time: Response time also known as flow time. Response time is the sum of finalization times of all the tasks [17]. Response time and makespan are the two major objectives to be minimized in research involving scheduling. However, minimization of makespan always results in the maximization of response time [6].

Minimizing Number of Delayed Jobs: Delayed Jobs means jobs that fail to meet their deadline [14]. Deadline or due date is a period of time a job must be completed [18]. The goal typically in such problems is to complete the maximum number of jobs by their deadlines. According to Klusacek and Rudova in [14] a higher machine usage fulfills resource owner's expectations, while a higher number of non delayed jobs guarantees a higher Quality of Service (QoS) provided for the users. By reducing the number of delayed jobs, QoS for the system that using the proposed scheduling technique also will be improve.

Maximizing Resource Utilization: Maximizing machine usage or resource utilization of the grid system is another important performance metrics. Utilization is the percentage of time that a resource is actually occupied, as compared with the total time that the resource is available for use. Low utilization means a resource is idle and wasted.

The datasets used consists of five folders. Each folder contains 20 different jobs file and each file has 3000 different jobs with specific inter-arrival time generated from negative exponential distribution. The lesser job inter-arrival time, the higher the contention in the system will be. DataSets-1 contained jobs with inter-arrival 1, DataSets-2 contained jobs with inter-arrival 2 and so on. There are total 150 heterogeneous machines used in this experiment.

6.1 Earliest Gap with Priority Rules Algorithms

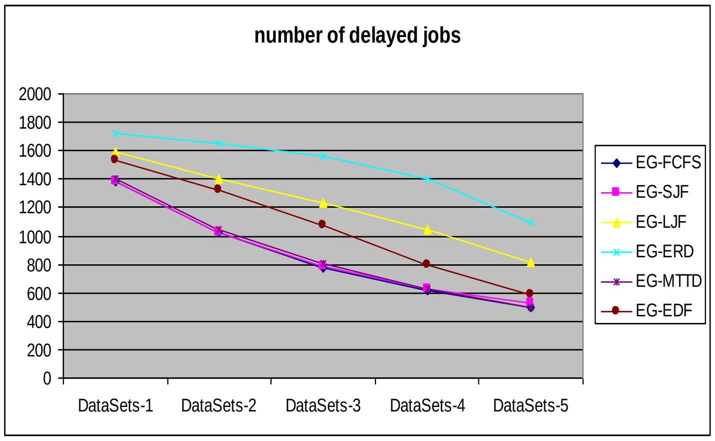

Figure 1. Graph Comparing the Number of Delayed Jobs between EG-Priority Rule Algorithms.

Figure 1 outline the total number of delayed jobs. From the graph, it is obvious that when system contention is high (DataSets-1=interarrival 1), possibility for jobs to be delayed is higher compared to the situation when system are less busy. EG-MTTD, EG-SJF and EG-FCFS outperform the other scheduler with EG-FCFS recorded the least delayed jobs for high contention system while EG-MTTD is at its best when the system have low loads. With limited and fixed resources (150 machines), there is not enough processing power to process jobs before its due date especially when the system contention is high. As can be seen in Figure 4, the proposed scheduler manage to utilize up to 90% of the machines processing power which is considered as high but it is still not enough to make sure all jobs able to finish on time.

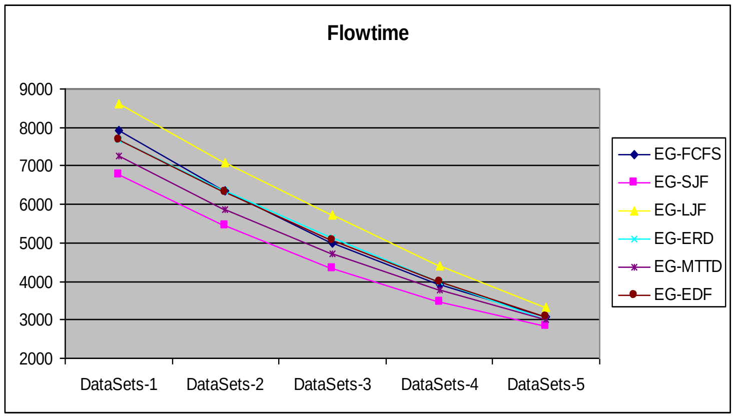

Figure 2. Graph Comparing the Flowtime between EG- Priority Rule Algorithms

Flowtime results shows in Figure 2 indicate that grid with heavy load system have higher flowtime compared to low load system. This is because the possibility of jobs have to wait longer in the queue when entering heavy load system is higher compared to low load system. This will contribute to longer waiting time and leads to higher number of delayed jobs and tardiness as depicted in Figure 1 and Figure 5.

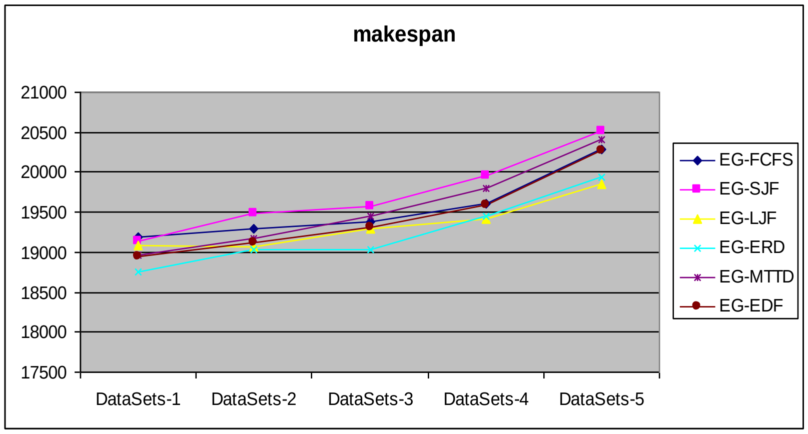

Figure 3. Graph Comparing the Makespan between EG-Priority Rule Algorithms

Figure 3 show an increasing trend for makespan from high load system to low load system. This is actually the opposite pattern of flowtime. Makespan time is lower in high contention grid system for all EG-PR algorithms because the final jobs finished earlier in the high contention grid compared to low contention.

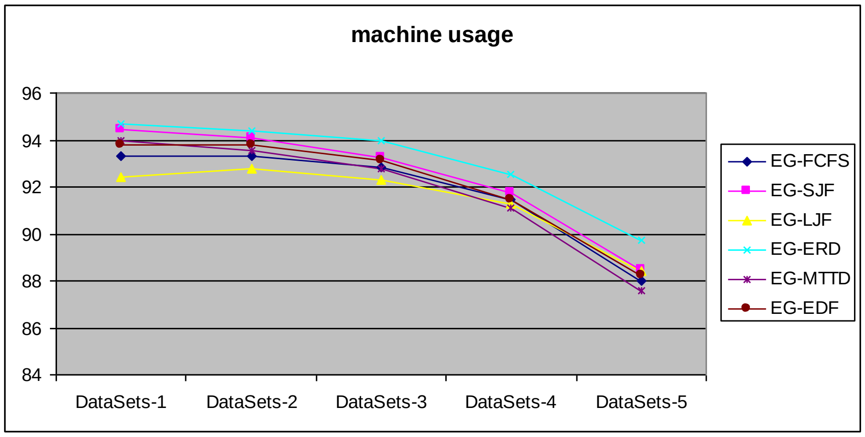

Figure. 4 Graph Comparing the Machine Usage between EG- Priority Rule Algorithms

It is not surprising to see high load grid system records higher percentage in the graph for machine usage in Figure 4. In high load grid system, more jobs can be fill in the queue gaps compared to low load system.

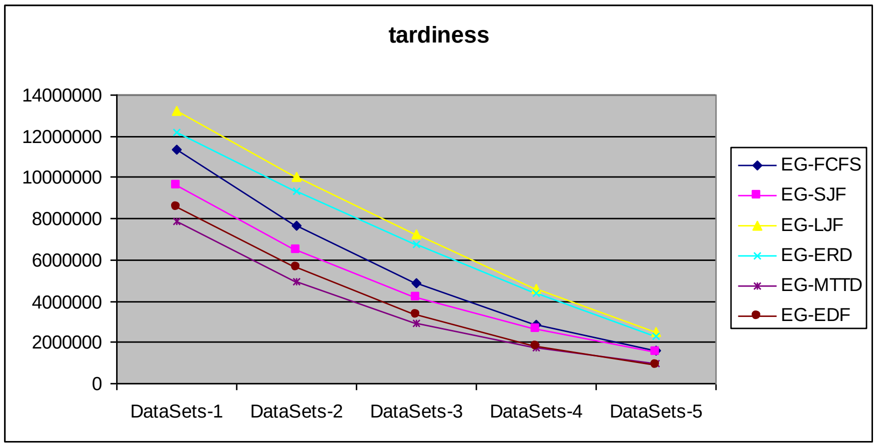

Figure 5. Graph Comparing the Tardiness between EG- Priority Rule Algorithms

As mentioned previously, when the flowtime is high, the possibility of jobs to be delayed also higher. Therefore, tardiness which presenting the overall delayed time of jobs also increased as mentioned in Figure 5.

6.2 Performance Comparison of Priority Rule Algorithms With and Without EG Under Heavy Load

To get a clear picture on the improvement made by the EG-PR algorithms, this section will compared EG-PR with original PR. The discussion will focused on the algorithms performance on heavy load grid systems.

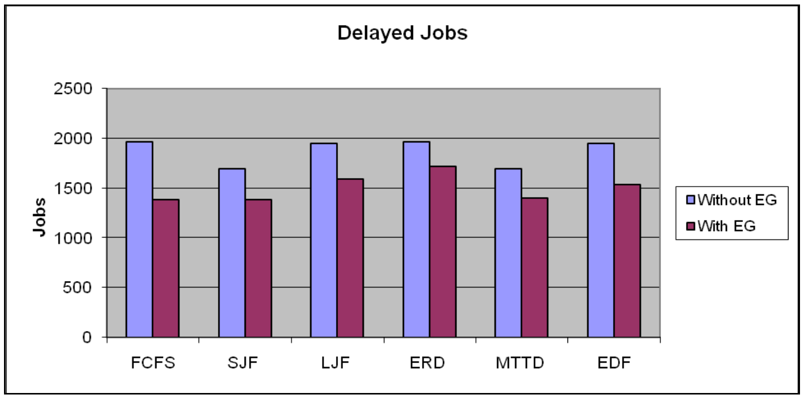

Figure 6. Graph Delayed Jobs of Priority Rule Algorithms With and Without EG based on DataSets-1

Figure 6 shows comparison of total delayed jobs between Priority Rule with Earliest Gap (EG-PR) and without EG Scheduling Algorithm. The graph shows the number of jobs meet their deadline can be increased when combining EG with PR. For instance, EG-FCFS not only shown the lowest number of delay jobs but also the highest improvement (29.5%) compare to other algorithm. EG-ERD shows the highest number of delayed jobs but the percentage of improvement is still considered as good with 12.5% compared to original ERD. The reason why there is significant improvement for all PR scheduling algorithms when combined with EG is mainly because the original PR produced many gaps. The EG algorithm will utilize this gap with a new job which resulting in more jobs can be processed before the deadline. EG-FCFS have the most improvement because the original FCFS produced more gaps then other PR scheduler.

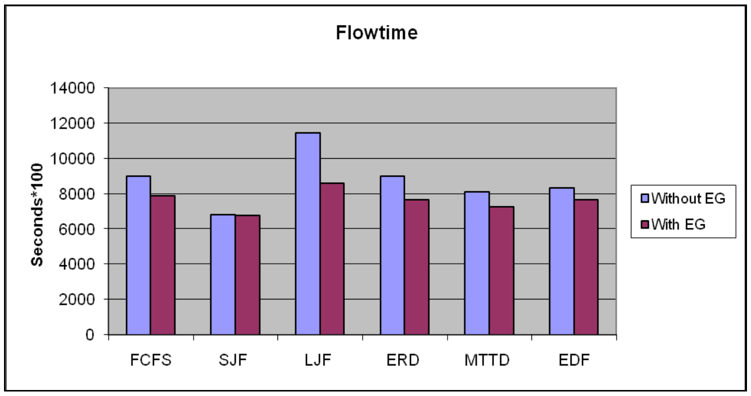

Figure 7. Graph Flowtime of Priority Rule Algorithms With and Without EG based on DataSets-1

Figure 7 shows flowtime comparison between PR algorithms with EG and without EG. From the result obtained, LJF shows the highest percentage of improvement (24.90%) after being integrate with EG. Other algorithms are below 15% of improvement while EG-SJF being the lowest (0.5%). However, SJF still has the lowest flowtime with or without EG. According to [6], to minimize flowtime is by schedule the shortest job first in the fastest resource. With minimum flowtime, more jobs can be finish prior to the deadline [19].

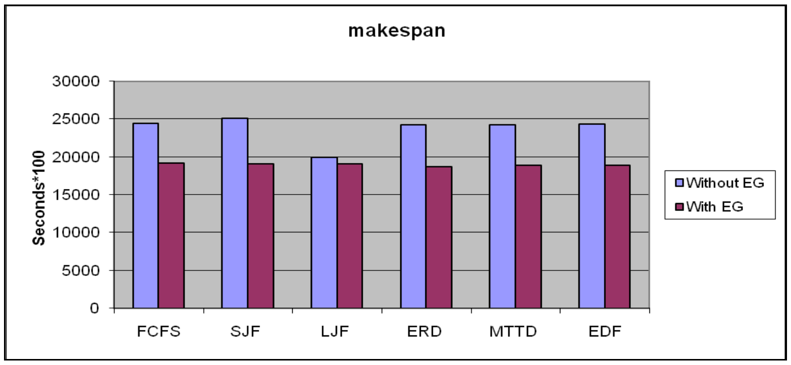

Figure 8. Graph Makespan of Priority Rule Algorithms With and Without EG based on DataSets-1

Figure 8 shows an improvement made by makespan for all PR algorithms after combined with EG. SJF with EG has shown the highest percentage with 23.76% of improvement. While others recorded 20-23% of improvement, EG-LJF only 4.3 % compared to LJF. This is because the original LJF already produced a minimum makespan scheduler [6] compared to other PR scheduler. However, after applying EG to other PR, the makespan is leveled between every scheduler.

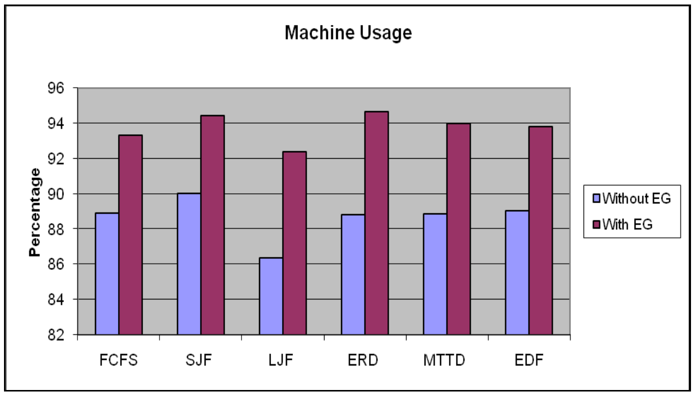

Figure 9. Graph Machine Usage of Priority Rule Algorithms With and Without EG based on DataSets-1

Figure 9 shows machine usage of priority rule algorithm with EG and without EG. Machine usage or resource utilization had increased gradually after applying simulation process with EG. Figure 9 depicts that EG-ERD algorithm shown the highest percentage of machine usage (94.67%) with 6.6% improvement compared to original ERD. The lowest machine usage for EG-PR is EG-LJF although the percentage of improvement is the highest among all (6.97%). Higher machine usage indicates less idle and wasted resources processing power. It also reflects the number of delayed jobs which means more jobs can be processed if the resources are fully utilized.

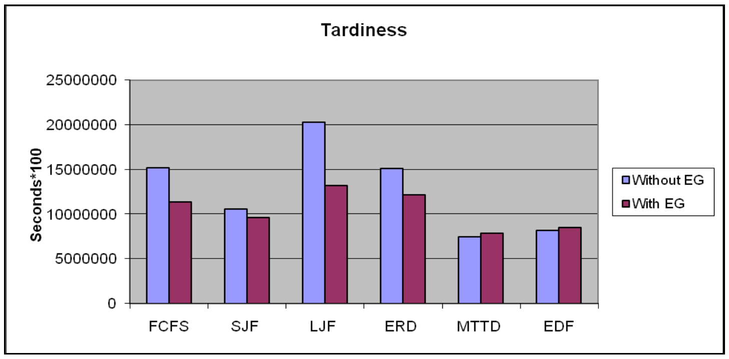

Figure 10. Graph Tardiness of Priority Rule Algorithms With and Without EG based on DataSets-1

While delayed jobs metric determine the number of jobs that failed to meet the due date, tardiness of jobs is the amount of time its completion time exceeds its due date. In Figure 10, most of the PR tardiness has been improved when integrate with EG. EG-LJF algorithm shows the highest percentage (34.85%) of improvement, followed by FCFS with 25% of improvement. While other algorithms shown improvement in prioritize their jobs, EG-MTTD and EG-ERD algorithm have show an opposite result. However, despite the drawback with -4.5% for EG-MTTD and -4.1% for EG-EDF, their tardiness is still the lowest compared to other algorithms.

6.3 Comparison of Priority Rule Algorithms With EG and With BH

The performance comparison made in this section is stressed on heavy load grid system. The performance of BH-PR algorithms will be discussed based on their percentage of better or worst compared to EG-PR.

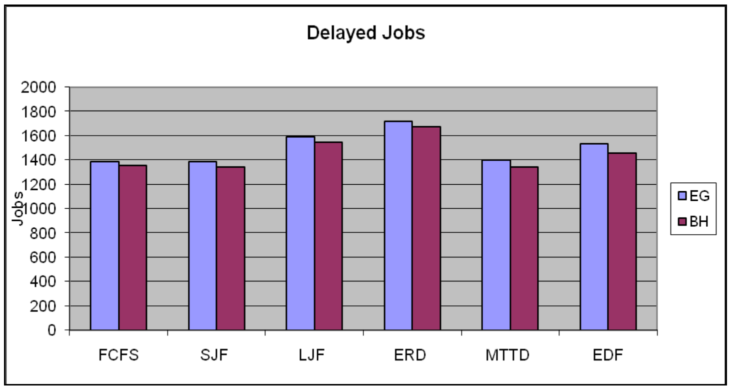

Figure 11. Graph Delayed Jobs of Priority Rule Algorithms With EG and With BH based on DataSets-1

Figure 11 shows delayed jobs of Priority Rule Scheduling Algorithm with EG and BF. Figure 11 denotes that the number of delayed jobs is lower for BH-PR compared to EG-PR. ERD is still having the highest delayed jobs, however, with BH it is better by 2.6% compared to with EG. The lowest number of delayed jobs recorded is from BH-SJF and BH-MTTD, 3.4% and 4.1% better then EG-SJF and EG-MTTD. Delayed jobs are lower for all BH-PR compared to EG-PR because BH algorithms utilize the gaps more effective than EG.

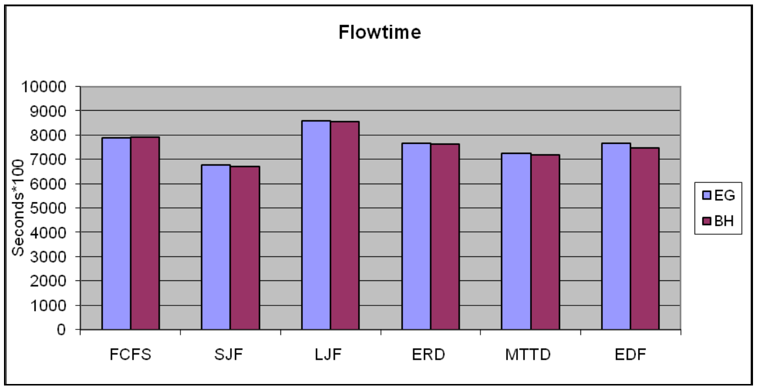

Figure 12. Graph Flowtime of Priority Rule Algorithms With EG and With BH based on DataSets-1

Figure 12 shows flowtime comparisons between Priority Rules Algorithms with EG and with BH. Generally, except for FCFS (-0.13%), other PR flowtime is its best with BH compared to EG. Based on the results obtained for BH, the pattern is still similar with EG, with LJF still having the highest flowtime and SJF the lowest. However BH proved to be better with small margin, 0.5% for LJF and 0.9% for SJF. BH-EDF algorithm shows the highest percentage of different, 2.24% better than EG-EDF. Other BH algorithms are below 1%.

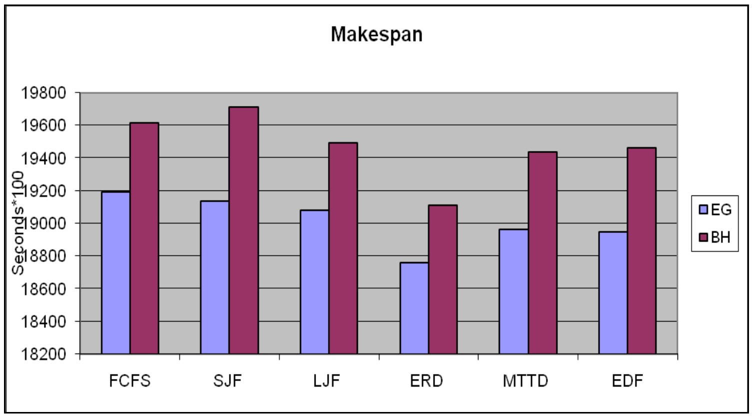

Figure 13. Graph Makespan of Priority Rule Algorithms With EG and With BH based on DataSets-1

Based on Figure 13, it is clear that makespan for a BH-PR algorithm is more than EG-PR. BH-SJF indicates not only the highest makespan but also the highest different percentage (3.0%). The lowest is still ERD with 1.9% over by BH compared to EG.

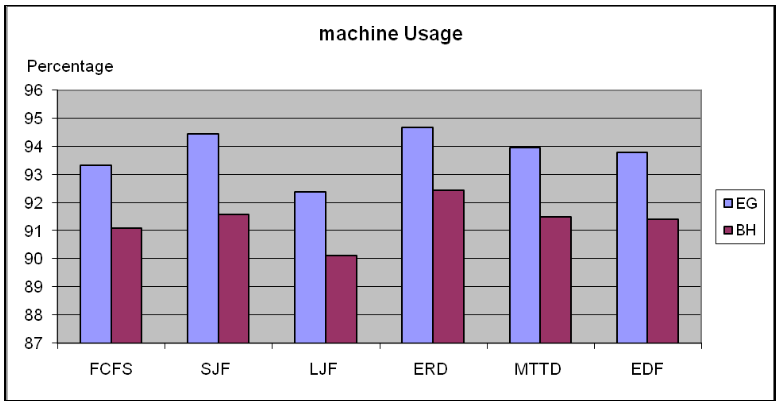

Figure 14. Graph Machine Usage of Priority Rule Algorithms With EG and With BH based on DataSets-1

Figure 14 shows comparison on machine usage between EG-PR and BH-PR. The reason why machine usage is significantly lower for BH-PR compared to EG-PR because; if two gaps were found in *machine$_i$ schedule*, and the second gap happen to be bigger than the first , the BH-PR scheduler will simply abandon the first or smaller gap and concentrate filling on second gap. The scheduler only fills the first gap with suitable job if the gap still exists on the following iteration. However, low machine usage or system utilization for BH-PR based scheduler indicates that most of the gap remains idle or unfilled. BH-LJF has the lowest machine usage (90.13%), only 2.4% lower than EG-LJF, while BH-ERD has the highest machine usage. Despite the drawback, all BH-PR algorithms still utilize over 90% of the resource which still can be considered as good performance.

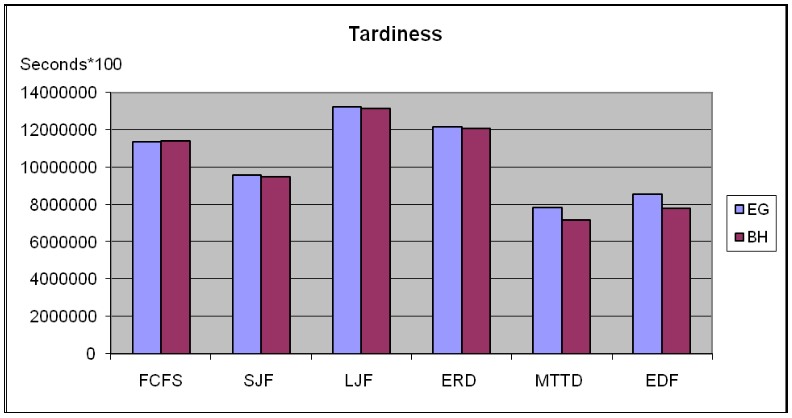

Figure 15. Graph Tardiness of Priority Rule Algorithms with EG and With BH based on DataSets-1

In Figure 15, the tardiness of Priority Rule Algorithms has been reduced when BH is used. While EG increase the tardiness of original MTTD and EDF by 4.5% and 4.1% (Figure 10), BH on the other hand manage to reduce it by 4.4% and 5.0%. It is 8.8% and 8.7% way better compared to EG-PR.

7. Conclusions

The results have proven that the proposed backfilling have improved all aspects tested based on the performance metrics measurement. This paper also introduced the BH technique which improves the flowtime, total tardiness and number of delayed jobs. However it has a drawback on makespan and machine usage.

References

[1] Boeres, C., A. Lima, and V.E. Rebello. *Hybrid Task Scheduling: Integrating Static and Dynamic Heuristics.* . in *In Proceedings of the 15th Symposium on Computer Architecture and High Performance Computing* 2003. Washington, DC: IEEE Computer Society.

[2] Chronopoulos, A.T., et al. *A class of loop self-scheduling for heterogeneous clusters*. in *Cluster Computing, 2001. Proceedings. 2001 IEEE International Conference on*. 2001.

[3] Klusacek, D., L. Matyska, and H. Rudová, *Alea – Grid Scheduling Simulation Environment*, in *Parallel Processing and Applied Mathematics*, R. Wyrzykowski, et al., Editors. 2008, Springer Berlin / Heidelberg. p. 1029-1038.

[4] Jackson, D., Q. Snell, and M. Clement, *Core Algorithms of the Maui Scheduler*, in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson and L. Rudolph, Editors. 2001, Springer Berlin / Heidelberg. p. 87-102.

[5] Henderson, R., *Job scheduling under the Portable Batch System*, in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson and L. Rudolph, Editors. 1995, Springer Berlin / Heidelberg. p. 279-294.

[6] Abraham, A., R. Buyya, and B. Nath, *Nature's Heuristics for Scheduling Jobs on Computational Grids*, in *The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000)*. 2000: Cochin, India.

[7] Garrido, J.M., *Performance modeling of operating systems using object-oriented simulation: a practical introduction*. 2000, Norwell, MA: Kluwer Academic Publishers

[8] Aysan Rasooli, O. *Introduction of Novel Rule Based Algorithms for Scheduling in Grid Computing Systems*. in *Proceedings of the 2008 Second Asia International Conference on Modelling & Simulation (AMS)*. 2008: IEEE Computer Society

[9] Lifka, D., *The ANL/IBM SP scheduling system*, in *Job Scheduling Strategies for Parallel Processing*, D. Feitelson and L. Rudolph, Editors. 1995, Springer Berlin / Heidelberg. p. 295-303.

[10] Srinivasan, S., et al. *Characterization of backfilling strategies for parallel job scheduling*. in *Parallel Processing Workshops, 2002. Proceedings. International Conference on*. 2002.

[11] Dan, T., *Backfilling Using System-Generated Predictions Rather than User Runtime Estimates.* IEEE Transactions on Parallel and Distributed Systems, 2007. 18: p. 789-803.

[12] Danelutto, M., et al., *Backfilling Strategies for Scheduling Streams of Jobs On Computational Farms*, in *Making Grids Work*. 2008, Springer US. p. 103-115.

[13] Mishra, A., S. Mishra, and D.S. Kushwah, *An Improved Backfilling Algorithm: SJF-BF*. Int. J. on Recent Trends in Engineering & Technology, 2011. Vol. 05 (1).

[14] Klusacek, D. and H. Rudova, *Improving QoS in computational Grids through schedule-based approach.* , in *Scheduling and Planning Applications Workshop at the Eighteenth International Conference on Automated Planning and Scheduling (ICAPS 2008)*. 2008: Sydney, Australia

[15] Klusacek, D., et al., *Comparison Of Multi-Criteria Scheduling Techniques*, in *Grid Computing*, S. Gorlatch, P. Fragopoulou, and T. Priol, Editors. 2008, Springer US. p. 173-184.

[16]. Yu, J. and R. Buyya, *A taxonomy of scientific workflow systems for grid computing.* SIGMOD Rec., 2005. **34**(3): p. 44-49.

[17] Xhafa, F. and A. Abraham, *Meta-heuristics for Grid Scheduling Problems*, in *Metaheuristics for Scheduling in Distributed Computing Environments*, F. Xhafa and A. Abraham, Editors. 2008, Springer Berlin / Heidelberg. p. 1-37.

[18] Brucker, P., *Scheduling Algorithms*. 5th ed. 2007, Berlin Heidelberg: Springer

[19] Enns, S.T., *Finite capacity scheduling systems: Performance issues and comparisons.* Computers & Industrial Engineering, 1996. **30**(4): p. 727-739.

Authors

Zafril Rizal M Azmi obtained his BSc and MSc in Computer Science from Universiti Teknologi Malaysia. Currently he is a PhD student working on Grid Scheduling.

Kamalrulnizam Abu Bakar obtained his Ph.D degree from Aston University (Birmingham, UK) in 2004. Currently, he is an Associate Professor in Computer Science at Universiti Teknologi Malaysia (Malaysia) and member of the “Pervasive Computing” research group. He involves in several research projects and is the referee for many scientific journals and conferences. His specialization includes mobile and wireless computing, information security and grid computing.

Mohd Shahir Shamsir obtained his BSc in Biotechnology and Plant Sciences from University of Sheffield in 1996 and PhD in Biological Sciences (Computational Biology) from University of Exeter in 2005. Currently he is a Senior Lecturer at Faculty of Bioscience & Bioengineering, Universiti Teknologi Malaysia. His research focuses on molecular dynamics and biodiversity databases.

Wan Nurulsafawati Wan Manan received her Bcs in computer science major in networking from University Technical Malaysia Malacca (UTeM) in 2006 and Master in Information Technology (Network) at Queensland University of Technology (QUT) on 2009. Currently she is a lecturer at Faculty of Computer System & Software Engineering, University Malaysia Pahang (UMP). Her research interest focuses on grid and cloud computing, network security and wireless network.

Abdul Hanan Abdullah received his BSc and MSc in Computer Science from the University of San Francisco and his PhD in the same field from the Aston University, UK in 1995. Currently he is a professor at Faculty of Computer Science and Information System, Universiti Teknologi Malaysia. His research interest includes network security, grid computing and active network.