Effect of Job Size Characteristics on Job Scheduling Performance

Kento Aida
Department of Computational Intelligence and Systems Science,
Tokyo Institute of Technology
4259, Nagatsuta, Midori-ku, Yokohama-shi 226-8502, JAPAN
phone: +81-45-924-5168 fax: +81-45-924-5165
E-mail: aida@dis.titech.ac.jp

Abstract

A workload characteristic on a parallel computer depends on an administration policy or a user community for the computer system. An administrator of a parallel computer system needs to select an appropriate scheduling algorithm that schedules multiple jobs on the computer system efficiently. The goal of the work presented in this paper is to investigate mechanisms how job size characteristics affect job scheduling performance. For this goal, this paper evaluates the performance of job scheduling algorithms under various workload models, each of which has a certain characteristic related to the number of processors requested by a job, and analyzes the mechanism for job size characteristics that affect job scheduling performance significantly in the evaluation. The results showed that: (1) most scheduling algorithms classified into the first-fit scheduling showed best performance and were not affected by job size characteristics, (2) certain job size characteristics affected performance of priority scheduling significantly. The analysis of the results showed that the LJF algorithm, which dispatched the largest job first, would perfectly pack jobs to idle processors at high load, where all jobs requested power-of-two processors.

1 Introduction

Space sharing is one of the job scheduling strategies that are often used on large-scale parallel computers. For instance, in pure space sharing among rigid jobs, a submitted job requests a certain number of processors, and a job scheduler dispatches the job to idle processors. Here, the number of processors that execute the job is the same as that requested by the job. A job scheduler needs to select a job to dispatch in appropriate order so as to execute multiple jobs efficiently. Many job scheduling algorithms, *e.g.* conventional FCFS (First Come First Served), LJF [1], Backfilling [2, 3], Scan [4], etc., have been proposed, and performance of these algorithms have been evaluated.

Many previous performance evaluation works assumed that characteristics of parallel jobs, or a parallel workload, followed a simple mathematical model. However, recent analysis of real workload logs, which are collected from many large-scale parallel computers in production use, shows that a real parallel workload has more complicated characteristics [5, 6, 7, 8]. For instance, (1) a percentage of *small jobs*, which request small number of processors, is higher than that of *large jobs*, which request a large number of processors, (2) a percentage of jobs that request power-of-two processors is high, etc.

These job characteristics seem to affect job scheduling performance. Particularly, the effect of job size characteristics seems to be strong in pure space sharing among rigid jobs. Many researchers investigated performance of job scheduling algorithms under more realistic workloads, which have above job size characteristics [5, 6, 7, 9]. For instance, Lo, Mache and Windisch compared performance of job scheduling algorithms under various workload models. They showed that the ScanUp algorithm [4] performed well, or increased processor utilization, as the proportion of jobs requesting power-of-two processors in the workload increased. However, mechanisms how the job size characteristics affect performance of job scheduling algorithms have not yet been clear.

This paper presents performance evaluation of job scheduling algorithms under various workload models, each of which has a certain characteristic related to the number of processors requested by a job. The goal of the work presented in this paper is to investigate mechanisms how the job size characteristics affect job scheduling performance. For this goal, this paper also analyzes the mechanism for job size characteristics that affected job scheduling performance significantly in the evaluation. First, this paper shows five workload models used in the performance evaluation. Each model has a single characteristic for job size. For instance, a percentage of small jobs is high in one workload model, and all jobs request power-of-two processors in the other one. Because a real workload has a combination of the multiple characteristics, the workload models in this paper are not suitable to investigate practical performance of job scheduling algorithms. However, it is meaningful to investigate the effect by the individual characteristic on job scheduling performance in order to investigate the mechanisms how the job size characteristics affect job scheduling performance. Next, the paper classifies job scheduling algorithms into three groups by techniques mainly used in the algorithms. These groups are FCFS, priority scheduling and first-fit scheduling. Finally, the paper shows performance evaluation results by simulation and analyzes the result that showed most significant change. The evaluation results showed that: (1) most scheduling algorithms classified into the first-fit scheduling showed best performance and were not affected by job size characteristics, (2) certain job size characteristics affected performance of priority scheduling significantly. The analysis of the results showed that the LJF algorithm, which dispatched the largest job [1] first, would perfectly pack jobs to idle processors at high load, where all jobs requested power-of-two processors.

Generally, a workload characteristic on a parallel computer depends on an administration policy or a user community. An administrator of a parallel computer needs to select an appropriate scheduling algorithm that schedules multiple jobs on the computer efficiently. The results in this paper will be useful information for administrators of parallel computers.

The rest of the paper is organized as follows. Section 2 gives a job scheduling model including workload models assumed in this paper. Section 3 describes job scheduling algorithms evaluated in the paper. Section 4 presents and discusses performance evaluation results. Finally, Section 5 presents conclusions and future work.

[1]The largest job indicates the job that requests the largest number of processors.

2 Job Scheduling Model

This section gives a job scheduling model including workload models assumed in this paper.

2.1 Parallel Computer

A parallel computer assumed in this paper is a multiprocessor system that consists of m processors. The processors are connected equally by a crossbar switch network or a multistage interconnection network. Thus, this paper assumes that locations of processors that execute a single job affect the execution time of the job negligibly, and that the locations of other jobs executed on the computer affect the execution time of the job negligibly too.

2.2 Job Scheduler

Figure 1 illustrates a model of a job scheduler assumed in this paper. In the figure, a job submitted by a user arrives at the shared job queue, and the job scheduler obtains the number of processors requested by the job (job size) [2]. No other information is given to the job scheduler.

The job scheduler gathers status of processors, idle or busy, and dispatches a job in the shared job queue to idle processors. Here, there are two policies to execute a job. In the first policy, a job scheduler dispatches a job to S processors and guarantee the job to be executed on S processors until its completion. Here, S denotes job size. In the second policy, a job scheduler does not guarantee the job to be executed on S processors, that is, the number of processors to execute the job depends on a congestion level of jobs on the parallel computer. A job executed in the first policy is called a rigid job [10]. In this paper, all jobs that arrive at the shared job queue will be executed as rigid jobs.

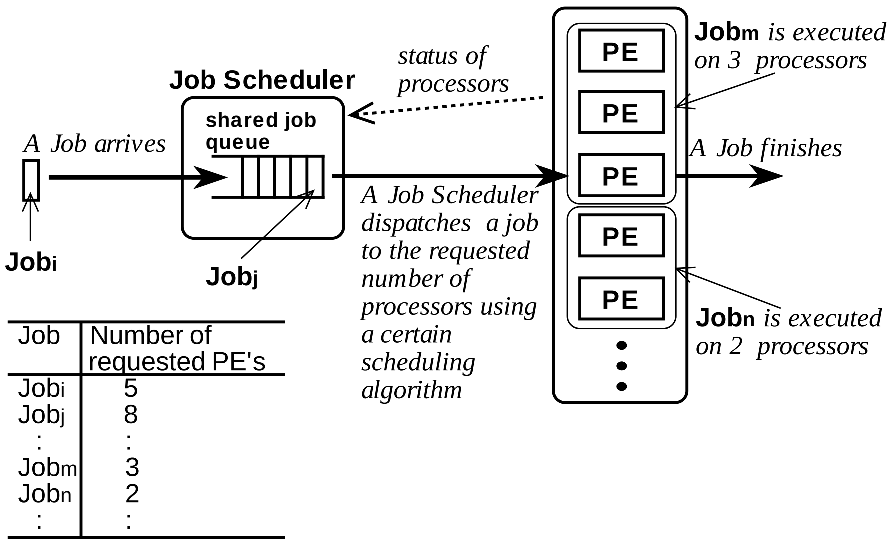
2.3 Workloads

A job assumed in this paper has three parameters: (1) the number of processors that a job requests, or job size, (2) execution time and (3) arrival time.

2.3.1 Job Size

Recent analyses of real workload logs, which are collected from many large-scale parallel computers in production use, show that job size in real parallel workloads has following characteristics [5, 6, 7, 8]:

[2]This paper assumes that job size is specified by a user.


Job Scheduler
shared job queue
A Job arrives
Jobi
Jobj
status of processors
A Job Scheduler dispatches a job to the requested number of processors using a certain scheduling algorithm
PE
PE
PE
PE
PE
Jobm is executed on 3 processors
A Job finishes
Jobn is executed on 2 processors

Job	Number of requested PE's
Jobi	5
Jobj	8
:	:
Jobm	3
Jobn	2
:	:

Figure 1: A Model for a Job Scheduler

(1) A percentage of small jobs, which request a small number of processors, is higher than that of large jobs, which request a large number of processors.

(2) A percentage of jobs that request power-of-two processors is high.

(3) A percentage of jobs that request square of n processors is high.

(4) A percentage of jobs that request multiples of 10 processors is high.

In order to discuss an effect of each characteristic on job scheduling performance, this paper uses following workload models in the performance evaluation. Each model is distinguished from others by distribution of job size. These models except the Uniform model were created from the Feitelson 1996 Model [5, 8].

(a) Uniform model
Job size is an integer that follows the Uniform distribution within the range [1,m]. This model is very simple synthetic model that used in many previous performance evaluation works.

(b) Harmonic model
Job size is an integer that follows the Harmonic distribution within the range [1,m]. The probability that $jobsize = n$ is proportional to $1/n^{1.5}$. This model represents the job size characteristic (1) in the above.

(c) Power2 model
Job size is an integer that is calculated by 2^k within the range [1,m]. (k is an integer.) The probability of each value is uniform. This model represents the job size characteristic (2).

(d) Square model
Job size is an integer that is calculated by k^2 within the range [1,m]. (k is an integer.) The probability of each value is uniform. This model represents the job size characteristic (3).

(e) Multi10 model
Job size is an integer that is calculated by $10 \cdot k$ within the range [1,m]. (k is an integer.) The probability of each value is uniform. This model represents the job size characteristic (4).

2.3.2 Execution Time

Execution time of a job follows the 3 Stage Hyper-exponential Distribution, which is defined in the Feitelson 1996 Model, in all workload models.

2.3.3 Arrival Time

Job arrival is assumed to be Poisson in all workload models. This paper represents a congestion level of jobs on a parallel computer by *load*, which is defined by the following formula.

$$load = \frac{\lambda \cdot p}{m \cdot \mu}$$

Table 1: A Summary of Job Scheduling Algorithms

algorithm	technique	
	priority	first-fit
FCFS	*no*	*no*
LJF	*yes*	*no*
SJF	*yes*	*no*
FCFS/First-Fit	*no*	*yes*
LJF/First-Fit	*yes*	*yes*
SJF/First-Fit	*yes*	*yes*

Here, λ denotes an arrival rate of a job, and μ indicates mean service rate on a parallel computer, that is, $1/\mu$ represents mean execution time of a job. Also, p and m denote mean job size in a workload and the number of processors on a parallel computer respectively.

3 Job Scheduling Algorithm

This section describes job scheduling algorithms evaluated in this paper. This paper classifies job scheduling algorithms into three groups by techniques mainly used in the algorithms. These groups are FCFS, priority scheduling and first-fit scheduling. Table 1 summarizes classification of the algorithms, and details of the algorithms are as follows.

3.1 FCFS

FCFS (First Come First Served) is a simple and conventional algorithm. In the FCFS scheduling, a job scheduler dispatches a job at the top of a shared job queue to idle processors whenever enough idle processors to execute the job are available.

3.2 Priority Scheduling

Scheduling algorithms classified into the priority scheduling give certain priority to each job in a shared job queue and dispatches the job with highest priority to idle processors. This paper evaluates two algorithms classified into this groups: LJF (Largest Job First) and SJF (Smallest Job First). Both algorithms give each job a priority that is calculated by the job size. From an implementation point of view, a job scheduler sorts jobs in a shared job queue by the job size, and then dispatches a job at the top of the shared job queue to idle processors whenever enough idle processors to execute the job are available. The LJF sorts jobs to the non-increasing order, that is a job scheduler dispatches *the largest job*, the job with the largest job size, first. On the other hand, the SJF sorts jobs to the non-decreasing order, or a job scheduler dispatches *the smallest job* first.

LJF [1], Scan [4] and Subhlok96 [6] can be classified into this group, because a job scheduler dispatches the largest/smallest job first in these algorithms.

3.3 First-Fit Scheduling

Scheduling algorithms classified into the first-fit scheduling dispatch jobs in a *first-fit* manner, or a job scheduler searches jobs in a shared job queue and dispatches the first job for which enough idle processors are available. This paper evaluates three algorithms classified into this group: FCFS/First-Fit, LJF/First-Fit and SJF/First-Fit. In the FCFS/First-Fit, a job scheduler searches jobs in a shared job queue from the top to the bottom, and dispatches the first job with job size that is not greater than the number of idle processors.

The LJF/First-Fit and the SJF/First-Fit use techniques both in the priority scheduling and in the first-fit scheduling. In the LJF/First-Fit, a job scheduler sorts jobs in a shared job queue to the non-increasing order of job size, and then dispatches jobs in the same way as FCFS/First-Fit. The SJF/First-Fit dispatches job in the same way as the LJF/First-Fit except that a job scheduler sorts jobs to the non-decreasing order of job size.

Backfilling [2, 3], FCFS-fill [11], LSF-RTC [2] and FPFS [12] can be classified into this group.

4 Performance Evaluation

This section shows simulation results to evaluate performance of job scheduling algorithms under the workload models described in Section 2. The simulation model follows the model described in Section 2. Here, the number of processors on a parallel computer, or m, is 128, and maximum execution time of a job is limited to 12 hours. Scheduling overhead is assumed to be negligible.

Performance of job scheduling algorithms is measured by two metrics: processor utilization and slowdown ratio. Processor utilization is the percentage that processors are busy over entire simulation. Slowdown ratio, SR, shows normalized data for mean response time, and it is derived by the following formula [9].

$$SR = \frac{T_{mean\_response}}{T_{mean\_execution}} \qquad (1)$$

Here, $T_{mean\_response}$ and $T_{mean\_execution}$ denote mean response time of a job and mean execution time of a job respectively. For instance, let us suppose that 10000 jobs were executed in an experiment. The mean response time of these 10000 jobs was 5 hours, and their mean execution time on processors was 2 hours. Then, the slowdown ratio is 2.5.

Job scheduling algorithms evaluated by the simulation are those in Table 1 and Backfilling [2, 3]. The Backfilling is a similar algorithm to the FCFS/First-Fit except that it dispatches jobs in a more conservative way. In the Backfilling, a job scheduler estimates the time when a job waiting at the top of a shared job queue will be dispatched, and then dispatches other waiting jobs for which enough idle processors are available, so long as they do not delay the start time of the waiting job at the top of the shared job queue [3]. The Backfilling can not be performed in the job scheduling model described in Section 2 because a job scheduler needs to obtain execution time of each job in advance [4]. However, this paper assumes that a job scheduler under the Backfilling obtains execution time of each job when a job arrives, and shows results for the Backfilling in order to see difference between the performance of the Backfilling and that of the FCFS/First-Fit. In addition, scheduling performed by the SJF/First-Fit is same as that by the SJF. Thus, results of the SJF/First-Fit are omitted from the paper.

Results shown in this section represent the average of 100 experiments, each of which runs 10000 jobs. The *load* does not vary during the experiment. The results have confidence intervals of $\pm 10\%$ or less at 95% confidence level. Performance evaluation results under each job model described in Section 2 are as follows:

4.1 Uniform

Figure 2 and Figure 3 show processor utilization and slowdown ratio respectively under the Uniform model. Both for processor utilization and for slowdown ratio, algorithms both in the priority scheduling and in the first-fit scheduling showed better performance as compared with the FCFS, except that the LJF slightly degraded slowdown ratio. In the priority scheduling algorithms, the LJF showed better processor utilization than the SJF. The maximum improvement by the LJF on the FCFS was 16% [5]. The algorithm that showed the best performance in the first-fit scheduling was the LJF/First-Fit, and it improved processor utilization compared with the FCFS by 34% maximum.

When job size of a job at the top of a shared job queue is larger than the number of idle processors, the FCFS *blocks* dispatch of jobs even when there are enough idle processors for other jobs in the shared job queue. This paper calls this situation *blocking*. The reason for low performance of the FCFS is that frequent blocking left many processors to be idle, that is, a job scheduler wasted many processor resources. In other words, the reason why the priority scheduling and the first-fit scheduling showed better performance than the FCFS is that they avoided wasting processor resources by sorting jobs or by searching jobs. The rest of this section discusses these reasons in detail.

First, this section discusses performance of the LJF. A scheduling problem discussed here can be regarded as the one-dimensional bin-packing problem, in which a job scheduler attempts to pack jobs into the idle processor space [6]. In the one-dimensional bin-packing problem, it is proved that waste of space in a bin is reduced by putting items into the non-increasing order of the item's size [14]. The LJF improved processor utilization because of this nature of the one-dimensional bin-packing problem. On the other hand, the LJF blocks dispatch of many small jobs by giving priority to larger jobs. Consequently, the LJF made response time of many small jobs longer, and then slowdown ratio under the LJF was slightly higher than that under the FCFS

Next, the reason why the SJF showed better performance compared with the FCFS is that the SJF reduced the occurrence of blocking. Blocking is caused by the situation that a larger job at the top of a shared job queue waits for dispatch while smaller jobs, for which enough idle processors are available, exist in the queue. In the SJF, a job scheduler gave priority to smaller jobs, and it prevented that a larger job blocked dispatch of many smaller jobs. For this reason, the SJF improved processor utilization. Also, the reason why slowdown ratio under the SJF was much lower than those both under the FCFS and under the LJF is that a large number of smaller jobs were dispatched prior to a small number of larger jobs.

Job scheduling algorithms in the first-fit scheduling including the Backfilling showed best performance under the Uniform workload. The reason for this best performance is that they prevented the occurrence of blocking efficiently by searching jobs in a shared job queue and dispatching a job for which

[3]There are variations of the Backfilling. The algorithm used in this paper is called aggressive backfilling [13].

[4]Recall that information that a job scheduler obtains from a job is only the job size in this model.

[5]The improvement indicated by "%" means $\frac{higher\ utilization - lower\ utilization}{lower\ utilization}$ [%] in this paper.

[6]This paper use the term "idle processor space" to refer to a bin that has capacity of the number of idle processors.

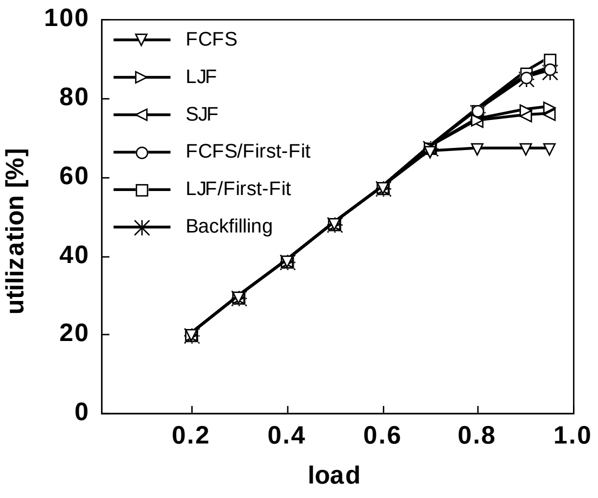


Figure 2: Processor Utilization (Uniform)

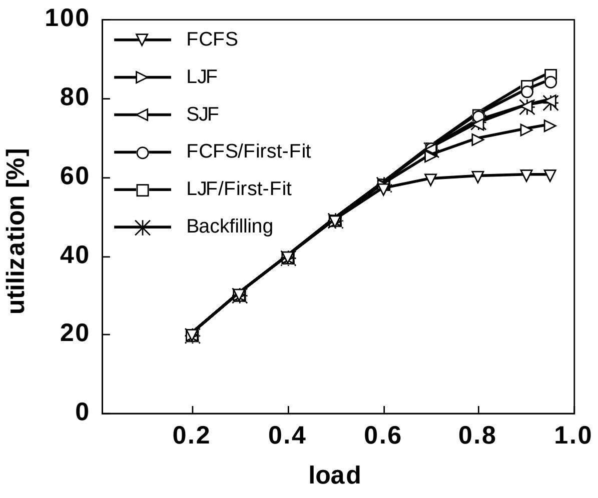


Figure 4: Processor Utilization (Harmonic)

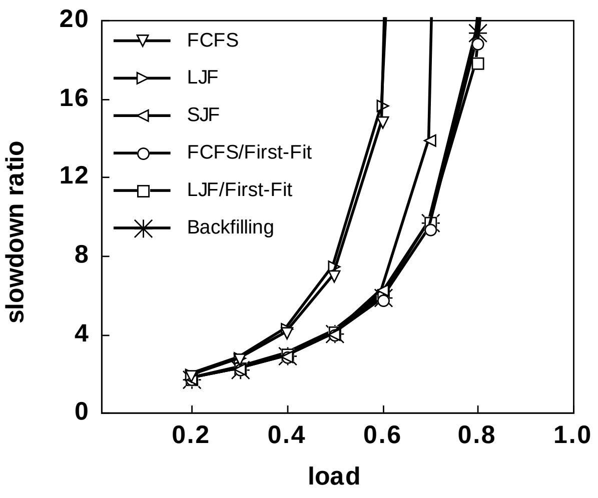


Figure 3: Slowdown Ratio (Uniform)

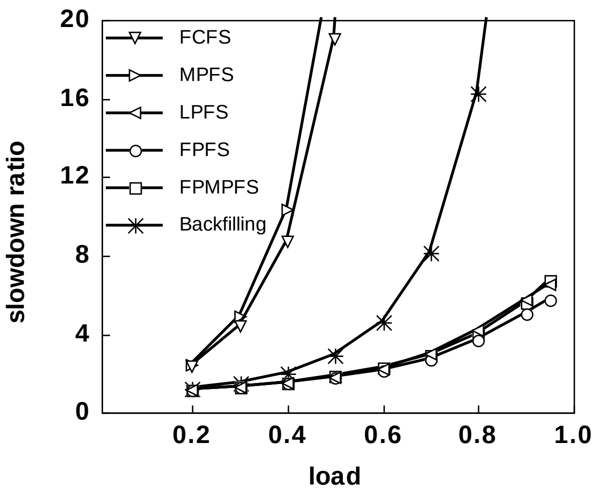


Figure 5: Slowdown Ratio (Harmonic)

enough idle processors are available. Although the priority scheduling also improved performance, they could not prevent blocking as sufficiently as the first-fit scheduling. For further analysis for performance of the first-fit scheduling under the Uniform model, see [12].

4.2 Harmonic

Figure 4 and Figure 5 show processor utilization and slowdown ratio respectively under the Harmonic model. There are several differences between results under the Uniform model and those under the Harmonic model. The major difference is observed in performance of the SJF. Processor utilization by the SJF under the Harmonic model was higher than that under the Uniform model. It showed better performance than the LJF. The SJF improved processor utilization compared with FCFS by 32% maximum, while the improvement under the Uniform model was 13%.

The reason for the performance difference is as follows: In order to improve processor utilization, it is effective to pack multiple jobs efficiently and to execute them concurrently, e.g. packing a large job with multiple small jobs, packing multiple small jobs and etc., so that waste of processor resources will be reduced. In the Uniform model, job size of jobs that arrive at a shared job queue follows the uniform distribution. A job scheduler dispatches small jobs prior to large jobs. Consequently, when the job scheduler dispatches a large job that remains in the shared job queue, there are not enough small jobs to be packed with the large job in the queue. On the other hand, under the Harmonic model, an arrival rate of small jobs is much higher than that of large jobs. It means that the probability that small jobs arrive soon after the job scheduler dispatches a large job is much higher. Thus, these small jobs will be packed with the large job. For this reason, the SJF under the Harmonic model packed multiple jobs more efficiently compared with that under the Uniform model.

In addition, slowdown ratio by the SJF under the Harmonic model was much lower than that under the Uniform model. It was close to slowdown ratio by the FCFS/First-Fit and the LJF/First-Fit, which were also much lower than that under Uniform Model. Their slowdown ratio remained small at higher *load*, e.g. lower than six at $load = 0.9$, while their slowdown ratio under the Uniform model went to infinity at $load > 0.8$. The reason why these scheduling algorithms showed lower slowdown ratio is that these dispatched a large number of small jobs prior to a small number of large jobs, where the number of small jobs under the Harmonic model was much greater than that under the Uniform model. For instance, 90% of jobs requested 16 processors or less and 58% of jobs requested two processors or less in the experiment.

The next major difference observed in the results is performance of the Backfilling. While the Backfilling showed almost same performance as other algorithms in the first-fit scheduling under the Uniform Model, the performance of the Backfilling was lower than the others under the Harmonic model. For instance, the Backfilling improved processor utilization compared with the FCFS by 31% maximum, while the improvement by the FCFS/First-Fit was 40%. It is supposed that the reason is due to the conservative way to dispatch jobs in the Backfilling. Further discussion about this reason is required but it will be the author's future work.

4.3 Power2

Figure 6 and Figure 7 show processor utilization and slowdown ratio respectively under the Power2 model. The major difference between results under the Uniform model and those under the Power2 model is performance of the LJF. Processor utilization by the LJF under the Power2 model was higher than that under the Uniform model. The LJF improved processor utilization compared with the FCFS by 48%

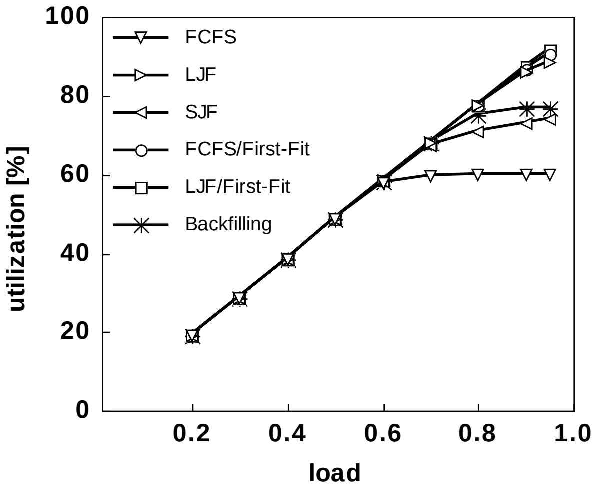


Figure 6: Processor Utilization (Power2)

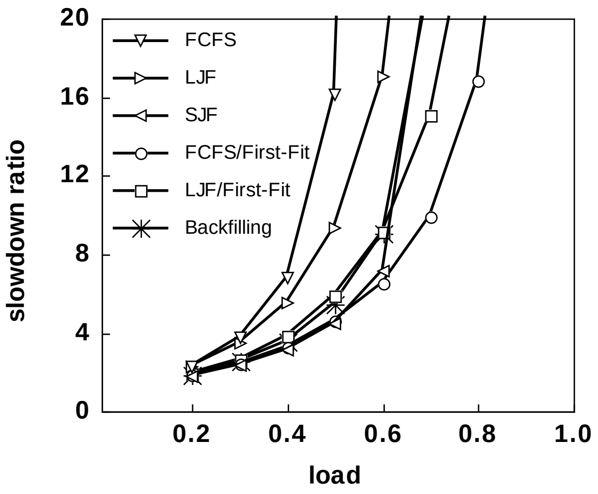


Figure 7: Slowdown Ratio (Power2)

maximum, while the improvement under the Uniform model was 16%. It was close to that of the FCFS/First-Fit, which improved processor utilization compared with the FCFS by 51% maximum.

Lo, Mache and Windisch compared the performance of job scheduling algorithms under different workloads, each of which has a different proportion of jobs that request power-of-two processors. They showed that the ScanUp algorithm [4] performed well, or increased processor utilization, as the proportion of jobs requesting power-of-two processors in a workload increased. Both results by Lo and by this paper are consistent from the point of view that a job scheduling algorithm performs well under the workload in which the proportion of jobs requesting power-of-two jobs is high.

The reason for the high processor utilization by the LJF under the Power2 model is that the LJF packed jobs very efficiently. In order to show the reasons more precisely, this paper regards the scheduling problem discussed here as the one-dimensional bin-packing problem again. The goal is to pack items, or jobs, into the bin, or the idle processor space, efficiently. Here, job size is 2^n (n is an integer within the range [0,7]), and capacity of the idle processor space is 2^7.

Let us assume that there are jobs, J_i's. The subscript denotes the order of the dispatch, that is, J_i is dispatched prior to J_{i+1}. P_i denotes job size of J_i, and IP indicates the number of idle processors. Let us suppose that s jobs has been dispatched by the LJF algorithm and they are still in execution, that is, processors are executing $J_1 ... J_s$. Then, the number of processors that are currently idle is derived by formula (2) [15].

$$IP = c_1 \cdot min(P_1, ..., P_s) \leq 128 \qquad (2)$$
$$c_1 \; is \; an \; integer \; and \; c_1 \geq 0$$

Because the LJF dispatches a larger job prior to a smaller job, J_s is the smallest job among jobs in execution. Thus,

$$IP = c_1 \cdot P_s. \qquad (3)$$

Here, there is a relation represented by (4) between job size of the job at the top of a shared job queue, namely P_{s+1}, and job size of J_s, or P_s.

$$P_s = 2^{c_2} \cdot P_{s+1} \leq 128 \qquad (4)$$
$$c_2 \; is \; an \; integer \; and \; c_1 \geq 0$$

Therefore, (5) is derived from both (3) and (4).

$$IP = c \cdot P_{s+1} \leq 128 \qquad (5)$$
$$c = c_1 \cdot 2^{c_2}$$

The formula (5) indicates that the number of idle processors will always be an integer multiple (possibly zero) of job size of the job at the top of a shared job queue, when the LJF performs job scheduling. Consequently, while enough number of jobs wait in a shared job queue, the LJF will pack jobs so as to fill idle processors perfectly.

The discussion mentioned above is valid for off-line bin-packing, or the assumption that all jobs have arrived at a shared job queue before a job scheduler starts to perform the scheduling. The job scheduling model in this paper could not be regarded as the off-line bin-packing. However, simulation log indicated that scheduling performed in the simulation showed partially same behavior as the discussion above when *load* was high, that is, many jobs were dispatched like off-line bin-packing at high *load*. Thus, the reason why processor utilization by LJF under the Power2 model was high can be explained by the above discussion.

While processor utilization by the LJF was high, slowdown ratio was not so good. The reason is same as that under the Uniform model, that is, the LJF blocked dispatch of many small jobs by giving priority to larger jobs. Also, the reason why slowdown ratio by the LJF/First-Fit was worse than that by the FCFS/First-Fit is the same as the previous reason.

In addition, as the results under the Harmonic model, performance of the Backfilling was also lower than the FCFS/First-Fit under the Power2 model. The Backfilling improved processor utilization compared with the FCFS by 28% maximum, while the improvement by the FCFS/First-Fit was 51%.

4.4 Square and Multi10

Figure 8 and Figure 9 show processor utilization under the Square model and the Multi10 model respectively. Both results were almost same as that under the Uniform model. Also, slowdown ratio under these both models were also similar to that under the Uniform model. This paper omits graphs for the slowdown ratio.

4.5 Feitelson Model

Finally, Figure 10 shows processor utilization under the Feitelson 1996 model [5, 8]. The Feitelson 1996 model is a well-known realistic workload model that is created by logs of six large-scale parallel computers in production use. A workload in this model has all characteristics that previous five models have.

Under this model, processor utilization was similar to that under the Power2 model. It mean that the characteristic, a percentage of jobs that request power-of-two processors is high, significantly affected performance of job scheduling algorithms in the Feitelson model. In other words, it seems that the proportion of jobs requesting power-of-two processors has strong effect on job scheduling performance in the real workload.

5 Conclusions

The goal of the work presented in this paper is to investigate mechanisms how job size characteristics affect job scheduling performance. This paper presented overall performance evaluation to show effect

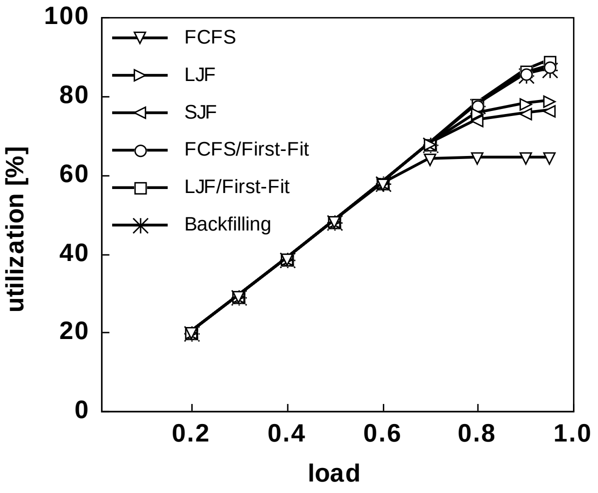


Figure 8: Processor Utilization (Squares)

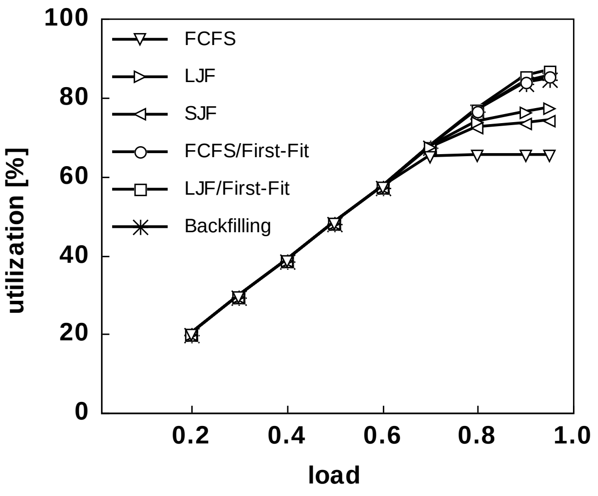


Figure 9: Processor Utilization (Multi10)

of an individual job size characteristic on job scheduling performance, and analyzed the evaluation result that showed most significant change.

The evaluation results showed that:

(1) The first-fit scheduling except the Backfilling showed best performance and were not affected by job size characteristics.

(2) Job size characteristics modeled by the Harmonic model and that by the Power2 model affected performance of priority scheduling significantly. Particularly, the effect of the job size characteristic under the Power2 model on the LJF performance was most significant among the results. It improved processor utilization compared with the FCFS by 48% maximum, while the improvement under the Uniform model was 16%.

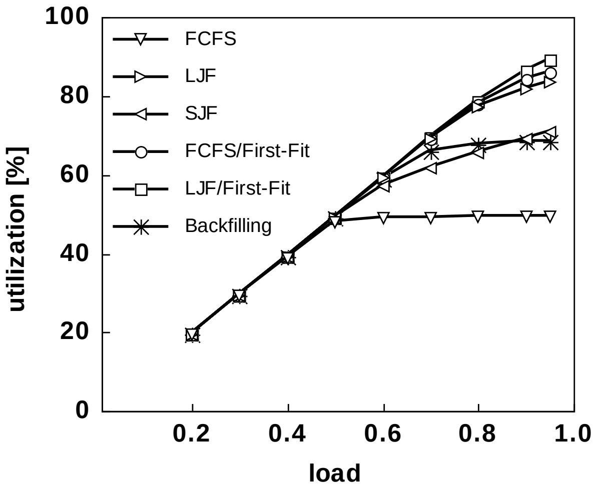


Figure 10: Processor Utilization (Feitelson)

(3) The analysis of scheduling performed by the LJF under the Power2 model showed that the LJF would perfectly pack jobs to idle processors under the Power2 model, when *load* on a parallel computer was high.

The analysis for the results in this paper has not yet been complete in order to achieve the goal. There are results that need more precise analysis, e.g. the effect on performance of the Backfilling. Furthermore, Scheduling algorithms evaluated in this paper, except the FCFS and the Backfilling, are not starvation-free. Thus, these scheduling algorithms need to be performed with some aging technique in order to improve practicability. However, the aging may affect performance of the job scheduling algorithms. It seems that this is a reason why the Backfilling showed lower performance than the FCFS/First-Fit. The author has evaluated the effect of an aging technique on the performance of the FCFS/First-Fit [12]. In this FCFS/First-Fit with aging, a job scheduler suppress to search jobs in a shared job queue when there exists a job waiting for long time (more than the pre-defined threshold), and dispatches this job first. The results showed that processor utilization of the FCFS/First-Fit with aging was degraded compared with that without aging. There are many aging techniques [4, 12] for job scheduling algorithms, and effects of the aging techniques on job scheduling performance seems to be various. Discussion about the effects is the author's future work.

References

[1] K. Li and K. Cheng. Job Scheduling in a Partitionable Mesh Using a Two-Dimensional Buddy System Partitioning Scheme. *IEEE Trans. on Parallel and Distributed Systems*, 2(4):413–422, 1991.

[2] D. A. Lifka. The ANL/IBM SP Scheduling System. In *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science 949*, pages 295–303. Springer-Verlag, 1995.

[3] J. S. Skovira, W. Chan, and H. Zhou. The EASY - LoadLeveler API Project. In *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science 1162*, pages 41–47. Springer-Verlag, 1996.

[4] P. Krueger, T. Lai, and V. A. Dixit-Radiya. Job Scheduling Is More Important than Processor Allocation for Hypercube Computers. *IEEE Trans. on Parallel and Distributed Systems*, 5(5):488–497, 1994.

[5] D. G. Feitelson. Packing Scheme for Gang Scheduling. In *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science 1162*, pages 89–110. Springer-Verlag, 1996.

[6] J. Subhlok, T. Gross, and T Suzuoka. Impact of Job Mix on Optimizations for Space Sharing Scheduler. In *Proc. of Supercomputing '96*, 1996.

[7] A. B. Downey. A parallel workload model and its implications for processor allocation. In *Proc. the 6th International Symposium of High Performance Distributed Computing*, pages 112–123, 1997.

[8] Parallel Workloads Archive. http://www.cs.huji.ac.il/labs/parallel/workload/.

[9] V. Lo, J. Mache, and K. Windisch. A Comparative Study of Real Workload Traces and Synthetic Workload. In *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science 1459*, pages 25–46. Springer-Verlag, 1998.

[10] D. G. Feitelson and L. Rudolph. Toward Convergence in Job Schedulers for Parallel Supercomputers. In *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science 1162*, pages 1–26. Springer-Verlag, 1996.

[11] R. Gibbons. A Historical Application Profiler for Use by Parallel Schedulers. In *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science 1291*, pages 58–77. Springer-Verlag, 1997.

[12] K. Aida, H. Kasahara, and S. Narita. Job Scheduling Scheme for Pure Space Sharing Among Rigid Jobs. In *Job Scheduling Strategies for Parallel Processing, Lecture Notes in Computer Science 1459*, pages 98–121, 1998.

[13] H. Franke, J. Jann, J. E. Moreira, P. Pattnaik, and M. A. Jette. An Evaluation of Parallel Job Scheduling for ASCI Blue-Pacific. In *Proc. SC99*, 1999.

[14] E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation Algorithms for Bin-packing - An Updated Survey. In *Algorithm Design for Computer System Design*, pages 49–106. Springer-Verlag, 1984.

[15] E. G. Coffman, M. R. Garey, and D. S. Johnson. Bin Packing with Divisible Item Sizes. *Journal of Complexity*, 3:406–428, 1987.