Characterization of Backfilling Strategies for Parallel Job Scheduling*

Srividya Srinivasan Rajkumar Kettimuthu Vijay Subramani P. Sadayappan

Department of Computer and Information Science
The Ohio State University
{srinivas,kettimut,subraman,saday}@cis.ohio-state.edu

Abstract

Although there is wide agreement that backfilling produces significant benefits in scheduling of parallel jobs, there is no clear consensus on which backfilling strategy is preferable e.g. should conservative backfilling be used or the more aggressive EASY backfilling scheme; should a First-Come First-Served(FCFS) queue-priority policy be used, or some other such as Shortest job First(SF) or eXpansion Factor(XF); In this paper, we use trace-based simulation to address these questions and glean new insights into the characteristics of backfilling strategies for job scheduling. We show that by viewing performance in terms of slowdowns and turnaround times of jobs within various categories based on their width (processor request size), length (job duration) and accuracy of the user's estimate of run time, some consistent trends may be observed.

1 Introduction

Effective job scheduling schemes are important for super computer centers in order to improve system metrics like utilization, and user metrics like turnaround time. Most of the studies in literature have reported these metrics averaged over all jobs of simulated traces. When comparing different scheduling strategies, many studies have concluded that the relative effectiveness of different schemes often depends on the job mix [1]. In order to gain greater insight into the relative effectiveness of different scheduling strategies, we group jobs into categories and study the effect of various schemes on the different categories. Three important factors that affect the scheduling of a job are - the length (run time of the job), width (number of nodes requested by the job) and the accuracy of the user's estimated runtime. By classifying jobs along these dimensions, and interpreting metrics like slowdown for various job categories instead of just a single average for the entire job trace, we are able to obtain new insights into the performance of different scheduling schemes.

* Supported in part by a grant from Sandia Laboratories

We address a number of issues in this paper:

- How does conservative backfilling [8] compare with aggressive (EASY) [7, 10] backfilling?
- The most commonly used priority policy for backfilling is First-Come First-Served(FCFS) [1,3,5]. How does it compare with other priority-policies, such as Shortest job First(SF) and eXpansion Factor(XF)?
- What is the effect of inaccurate user estimates of job runtime? A recent study [1] concluded that with FCFS, performance is actually enhanced by worse user estimates, suggesting that it might be desirable for supercomputer centers to systematically multiply user-specified wall-clock limits by some factor. We study this issue, considering different scheduling priorities and backfilling policies, viewing the results by grouping them based on the extent of over-estimation.

The paper is organized as follows. We provide background about job scheduling strategies in Section 2 and information about the job traces in Section 3. In Section 4, we compare conservative and aggressive backfilling under the assumption of exact user estimates of runtime. In Section 5, we study the effect of inaccurate estimation of job runtime. In Section 6, we provide conclusions and discuss future work.

2 Background

Scheduling of parallel jobs is usually viewed in terms of a 2D chart with time along one axis and the number of processors along the other axis. Each job can be thought of as a rectangle whose height is the user estimated run time and width is the number of processors required. Parallel job scheduling strategies has been widely studied in the past [1, 2, 5, 6, 9, 11, 12]. The simplest way to schedule jobs is to use the First-Come-First-Served (FCFS) policy. This approach suffers from low system utilization. Backfilling [7, 8, 13] was proposed to improve system utilization and

Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)
1530-2016/02 $17.00 © 2002 IEEE

has been implemented in several production schedulers [4]. Backfilling works by identifying "holes" in the 2D chart and moving forward smaller jobs that fit those holes. There are two common variations to backfilling - conservative and aggressive (EASY)[8, 10]. In conservative backfill, every job is given a reservation when it enters the system. A smaller job is moved forward in the queue as long as it does not delay any previously queued job. In aggressive backfilling, only the job at the head of the queue has a reservation. A small job is allowed to leap forward as long as it does not delay the job at the head of the queue. Under FCFS, the priority of a job is its wait time. Other priority policies like Shortest job First (SF), eXpansion Factor(XF) can be used. Under SF, the priority of a job is inversely proportional to its user estimated run time. Under the XFactor priority scheme, the priority of a job is its XFactor which is defined as follows:

XFactor = (Wait time + Estimated Run time) / Estimated Run time

Some of the common metrics used to evaluate the performance of scheduling schemes are the average turnaround time and the average bounded slowdown. We use these metrics for our studies. The bounded slowdown of a job is defined as follows:

Bounded Slowdown = (Wait time + Max(Run time, 10))/ Max(Run time, 10)

The threshold of 10 seconds is used to limit the influence of very short jobs on the metric.

3 Workload Characterization

The simulation studies were performed using a locally developed scheduler with workload logs from supercomputer centers. From the collection of workload logs available from Feitelson's archive [3], the CTC workload trace and the SDSC workload trace were used to evaluate the various schemes. The CTC trace is from the 430 node Cornell Theory Center. The SDSC trace is from the 128 node IBM SP2 system at the San Diego Supercomputer Center.

To gain a better understanding of the performance of various schemes, we classify the jobs into various categories based on the runtime and the number of processors requested, and analyze the average slowdown and turnaround time for each category. In the initial part of the study we compare the performance of the different schemes assuming accurate user estimates. Later, we present the results taking user estimate inaccuracy in to account. Simulation studies were performed under both normal and high loads. A high load condition was simulated by shrinking the inter-arrival times of jobs. Similar trends were observed under both loads. The trends are pronounced under high load. Hence we present the results for high load in subsequent sections.

To analyze the performance of jobs of different sizes and lengths, jobs were classified into 4 categories: two categories based on their run time - Short(S) and Long(L) and two categories based on the number of processors requested - Narrow(N) and Wide(W). The criteria used for job classification is shown in Table 1. The distribution of jobs in the various traces, corresponding to the four categories is given in Tables 2 and 3.

Job Categorization Criteria

	<=8 Processors	>8 Processors
<=1Hr	SN	SW
>1Hr	LN	LW

Table 1. Categorization of jobs based on their runtime and width

Job Distribution

	N	*W*
S	45.06%	11.84%
L	30.26%	12.84%

Table 2. CTC Trace

	N	*W*
S	47.24%	21.44%
L	20.94%	10.38%

Table 3. SDSC Trace

4 Evaluation with Exact Estimates of Runtime

Previous studies have been inconclusive about which backfill strategy is preferable: conservative or aggressive. Here we attempt to characterize the performance of these backfill policies under different priority schemes by using trace based simulation. Super computer schedulers make scheduling decisions based on the user estimated run time. Thus, aborted jobs and the poorly estimated jobs can skew the average slowdown. Therefore to eliminate such effects, we initially study the performance of the various schemes assuming accurate user estimates. Later, in Section 5, we study the impact of inaccurate user estimates of runtime.

4.1 Priority Equivalence

Under conservative backfilling, if the user estimates are accurate, irrespective of the priority policy used, the resulting schedules are exactly the same. This is because,

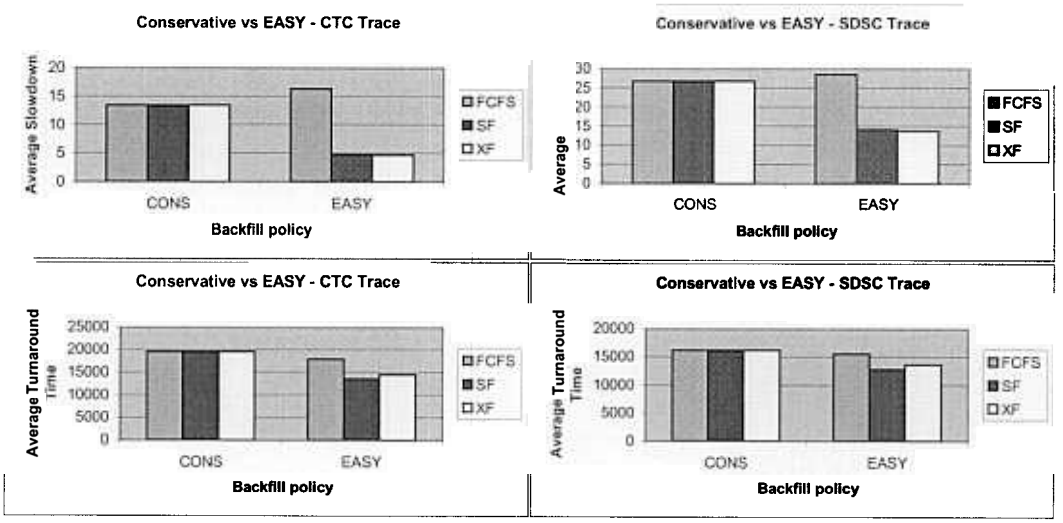

Figure 1. Comparison of conservative vs. EASY backfilling under different priority policies: Accurate user estimates. Under conservative backfilling, all priority policies are equivalent. EASY backfilling with SF or XF priority outperforms conservative backfilling.

under conservative backfilling, every job is given a reservation at the time it enters the system. Thus these start time guarantees are given in arrival order. Therefore, irrespective of the priority scheme used for ordering the jobs in the idle queue, the reservations will be exactly the same. The difference between the priority schemes occurs only when backfilling occurs to fill holes created by jobs that complete earlier than expected. When jobs abort, or complete before the user estimated time, jobs in the idle queue are considered for backfill in the priority order and hence different priority schemes can produce different schedules. But if the user estimates are accurate, no new "holes" are created, and no new backfilling opportunities arise for any queued job. Hence all priority schemes result in exactly the same schedule under conservative backfilling.

4.2 Conservative versus Aggressive Backfilling

Figure 1 shows the overall average slowdown and turnaround time for the different backfill policies under different priority schemes for the two traces. We observe that EASY backfilling with XF or SF priority policy clearly outperforms conservative backfilling with XF or SF. To explain the above observation we first compare conservative and EASY backfilling under FCFS priority. In conservative backfilling, a newly arriving job is given a reservation at the earliest time that will not violate any previously existing guarantees. The existing reservations act as roofs in the schedule that prevent later arriving jobs from backfilling easily. The longer the job is the more difficult it is for it to get a reservation ahead of previously arrived jobs. Therefore long jobs find it difficult to backfill under conservative backfilling. EASY backfilling relaxes this constraint by maintaining only one reservation at any point of time. The presence of only one blocking reservation in the schedule helps long jobs to backfill more easily. Wide jobs find it difficult to backfill because they cannot find enough free processors easily. Conservative backfill helps such wide jobs by guaranteeing them a start time when they enter the system. In EASY backfill, since these jobs are not given a reservation until they reach the head of the idle queue, more jobs can backfill ahead of them. Thus jobs in the Long Narrow(LN) category benefit from EASY backfilling, while jobs in the Short Wide(SW) category benefit from conservative backfilling. As far as the Short Narrow(SN) jobs are concerned, there is no clear trend between conservative and EASY backfilling because these jobs backfill quickly in both the schemes. Similarly, for the Long Wide(LW) jobs, there is no clear advantage in one scheme over the other because conservative backfilling provides these with the advantage of reservation while EASY backfilling offers better backfilling opportunities due to fewer blockades in the schedule. Figure 2 shows a category-wise comparison of EASY versus conservative backfilling for the CTC trace. The relative change in slowdown for EASY backfilling is presented, relative to conservative backfilling. It can be observed that the above expectations for the different job categories are clearly borne out. Thus the overall performance of conservative versus EASY backfilling will depend on the relative number of jobs in each of the categories.

When we compare conservative and EASY backfilling

Proceedings of the International Conference on Parallel Processing Workshops (ICPPW'02)
1530-2016/02 $17.00 © 2002 **IEEE**

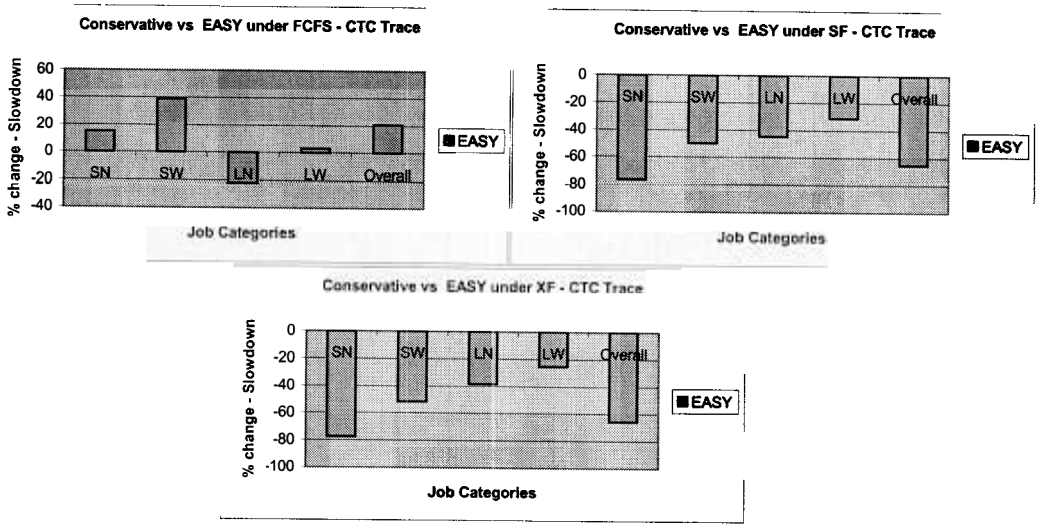

Figure 2. Category-wise performance comparison of conservative vs. EASY backfilling under different priority policies: Accurate user estimates. Under EASY backfilling with SF or XF priority policy, the short(SN and SW) jobs and the long narrow(LN) jobs benefit.

under SF priority, we find that the short jobs (SN and SW) benefit under EASY backfilling because they receive higher priority. Further the Long Narrow (LN) jobs benefit from the increased backfill opportunities provided by EASY as explained above. Because of the improvements in each of these categories, the overall average tends to decrease under EASY backfilling with SF priority compared to conservative backfill with SF priority. A similar observation is true when comparing conservative and EASY backfill under the XF priority policy also which implicitly gives higher priority to the short jobs because their XFactors tend to rise quickly compared to long jobs. These trends are visible in Figure 2. Figure 1 indicates that SF and XF priority policies perform comparably with respect to overall average slowdown. However, the worst-case turnaround time under EASY backfilling is worse compared to conservative backfilling. This can be observed from Table 4. The lack of a bound in the delay that a job could incur under EASY backfilling manifests itself as high values of worst case turnaround time.

	Worst-case Turnaround Time
Conservative	125044
FCFS EASY	127135
XF EASY	169960
SF EASY	221571

Table 4. Comparison of worst-case turnaround time in seconds: CTC Trace

5 Impact of Inaccurate Estimates of Runtime

5.1 Systematic Overestimation

Previous studies [5, 8, 14] suggest that inaccurate user estimates of runtime may be beneficial because of better backfilling possibilities. These studies evaluate the effect of inaccurate user estimates by multiplying the runtime of all jobs by an over estimation factor. Presented below are the results of such systematic overestimation. Tables 5 and 6 contain results for R=1 where the user estimates are accurate, R=2 where the user estimated run time is twice the actual run time and R=4 where the user estimated run time is four times the actual run time. As pointed out by [5] when systematic overestimation is used, the overall slowdown decreases significantly compared to the slowdowns when the user estimates are accurate. This is because of the larger holes created in the schedule when jobs finish earlier than expected. This effect is more pronounced under conservative backfilling, where early job completions create holes in the schedule and better backfilling opportunities. With EASY backfilling, the difference is less significant because EASY backfilling provides good backfilling opportunities even when user estimates are accurate.

5.2 Actual User Estimates

We next present the results using actual user estimates. We observe from Figure 3 that although the overall slowdown improves with systematic overestimation, with actual user estimates the overall slowdown deteriorates. In order to get better insights, we categorize the jobs into two

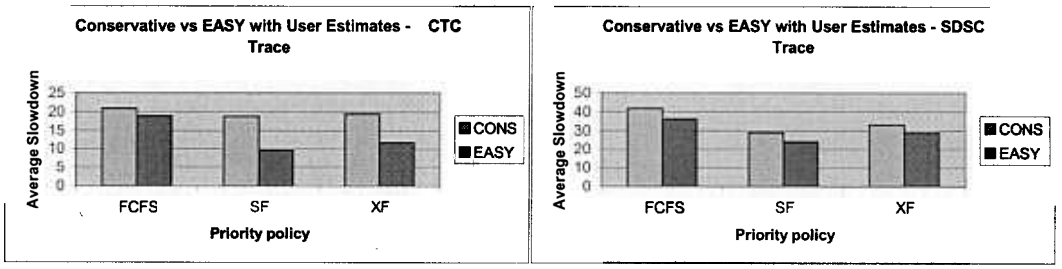

Figure 3. Comparison of conservative vs. EASY backfilling under different priority policies: Actual user estimates. EASY backfilling has lower overall average slowdown compared to conservative backfilling for all priority policies.

	R=1	R=2	R=4
FCFS	13.47	6.75	6.71
SF	13.47	3.84	3.62
XF	13.47	3.71	4.53

Table 5. Systematic overestimation: Conservative

	R=1	R=2	R=4
FCFS	16.27	11.98	12.34
SF	4.73	3.93	4.68
XF	4.63	3.97	4.53

Table 6. Systematic overestimation: EASY

categories - well estimated jobs and poorly estimated jobs. Jobs whose user estimated run time is less than or equal to twice their actual run time are considered to be well estimated and jobs whose user estimated run time is greater than twice their actual run time are considered to be poorly estimated. Figure 4 shows a comparison of the average slowdown for the well and poorly estimated jobs with that of the same set of jobs when all user estimates are accurate for the CTC trace. Similar trends were observed for the SDSC trace.

The average slowdown of the well estimated jobs decreases compared to their slowdown when all user estimates are accurate. On the other hand, the average slowdown of the poorly estimated jobs increases compared to when all user estimates are accurate. This is because the well estimated jobs benefit from the backfilling opportunities due to the holes created by the badly estimated jobs. The poorly estimated jobs owing to their increased length have difficulty backfilling and hence have higher slowdowns compared to when all user estimates are accurate. As before, we observe that this effect is more pronounced under conservative compared to EASY. This is because EASY backfilling has good backfilling opportunities even under accurate user estimates. Therefore, when user estimates are not accurate, the well estimated jobs do not benefit as much as under conservative backfilling and the poorly estimated jobs do not worsen as much as under conservative backfilling, because even the seemingly long jobs have good backfilling opportunities under EASY backfilling.

When the job trace contains a mix of both well and poorly estimated jobs, performance of conservative backfilling is comparable to that of EASY backfilling for the well estimated jobs under the various priority policies. However, if all the jobs in the trace were very well estimated, then performance of conservative backfilling would be much worse than that of the EASY backfilling (as shown in Section 4, where the user estimates are assumed to be accurate). EASY backfilling performs better than conservative backfilling under both SF and XF priority policies with respect to overall average slowdown.

Table 7 shows the worst-case turnaround time for the two backfilling schemes under different priority policies. We observe as before that the worst-case turnaround time under EASY is worse compared to conservative.

	Conservative	EASY
FCFS	139221	151818
XF	138265	175610
SF	150268	171737

Table 7. Comparison of worst-case turnaround time in seconds: CTC Trace

6 Conclusions and Future Work

In this paper we studied the effect of various backfilling schemes on different priority policies. We observed that even though the overall slowdown is trace dependent, on finer categorization of the jobs in a trace, consistent category-wise trace independent trends become ev-

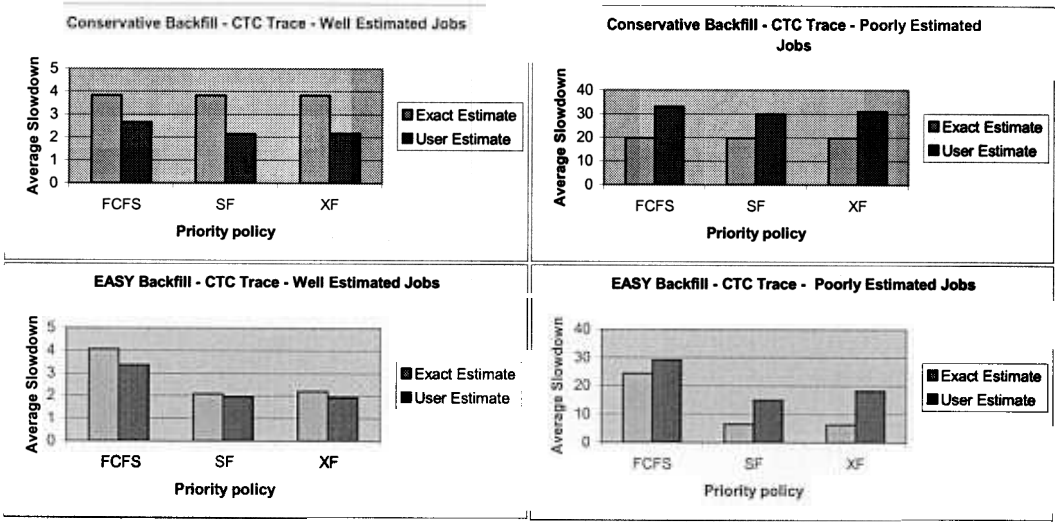

Figure 4. Comparison of the average slowdown under accurate user estimates vs. actual user estimates. The average slowdown of well estimated jobs decreases while that of the poorly estimated jobs increases.

ident. Inaccurate estimates can cause significant deterioration of the overall slowdown. We observed that conservative backfilling, by providing reservations to all jobs, limits backfilling opportunities. EASY backfilling on the other hand, improves backfilling opportunities but the jobs that have difficulty backfilling (e.g wide jobs) get a reservation only when they manage to get to the head of the queue, thus resulting in significant worsening in the worst-case turnaround time. Therefore, instead of the non-selective nature of reservations with both conservative and aggressive backfilling, we are investigating a selective backfilling strategy, wherein jobs do not get reservation until their expected slowdown exceeds some threshold, whereupon they get a reservation. By doing so, if the threshold is chosen judiciously, few jobs should have reservations at any time, but the most needy of jobs get assured reservations.

Acknowledgments

We would like to thank the anonymous referees for their helpful suggestions on improving the presentation of this paper.

References

[1] K. Aida. Effect of job size characteristics on job scheduling performance. In *Proceedings of JSSPP*, pages 1–17, 2000.

[2] O. Arndt, B. Freisleben, T. Kielmann, and F. Thilo. A comparative study of online scheduling algorithms for networks of workstations. *Cluster Computing*, 3(2):95–112, 2000.

[3] D. G. Feitelson. Logs of real parallel workloads from production systems. http://www.cs.huji.ac.il/labs/parallel/workload/logs.html.

[4] D. Jackson, Q. Snell, and M. J. Clement. Core algorithms of the maui scheduler. In *JSSPP*, pages 87–102, 2001.

[5] P. J. Keleher, D. Zotkin, and D. Perkovic. Attacking the bottlenecks of backfilling schedulers. *Cluster Computing*, 3(4):245–254, 2000.

[6] R. Kettimuthu, V. Subramani, S. Srinivasan, T. B. Gopalsamy, and P. Sadayappan. Selective preemption strategies for parallel job scheduling. To appear in Proceedings of the International Conference on Parallel Processing, 2002.

[7] D. Lifka. The ANL/IBM SP scheduling system. In *JSSPP*, pages 295–303, 1995.

[8] A. W. Mu'alem and D. G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the ibm sp2 with backfilling. In *IEEE Transactions on Parallel and Distributed Computing*, volume 12, pages 529–543, 2001.

[9] D. Perkovic and P. J. Keleher. Randomization, speculation, and adaptation in batch schedulers. *Cluster Computing*, 3(4):245–254, 2000.

[10] J. Skovira, W. Chan, H. Zhou, and D. Lifka. The easy - loadleveler API project. In *JSSPP*, pages 41–47, 1996.

[11] A. Streit. On job scheduling for hpc-clusters and the dynp scheduler. In *HiPC*, pages 58–67, 2001.

[12] V. Subramani, R. Kettimuthu, S. Srinivasan, and P. Sadayappan. Distributed job scheduling on computational grids using multiple simultaneous requests. In *Proceedings of the 11th High Performance Distributed Computing Conference*, 2002.

[13] D. Talby and D. Feitelson. Supporting priorities and improving utilization of the ibm sp scheduler using slack-based backfilling. In *In Proceedings of the 13th International Parallel Processing Symposium, 1999.*, 1999.

[14] D. Zotkin and P. Keleher. Job-length estimation and performance in backfilling schedulers. In *Proceedings of the 8th High Performance Distributed Computing Conference*, 1999.