Selective Reservation Strategies for Backfill Job Scheduling[⋆]

Srividya Srinivasan, Rajkumar Kettimuthu, Vijay Subramani, and P. Sadayappan

The Ohio State University, Columbus, OH, USA
{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 - should conservative backfilling be used or the more aggressive EASY backfilling scheme. Using trace-based simulation, we show that if performance is viewed within various job categories based on their width (processor request size) and length (job duration), some consistent trends may be observed. Using insights gleaned by the characterization, we develop a selective reservation strategy for backfill scheduling. We demonstrate that the new scheme is better than both conservative and aggressive backfilling. We also consider the issue of fairness in job scheduling and develop a new quantitative approach to its characterization. We show that the newly proposed schemes are also comparable or better than aggressive backfilling with respect to the fairness criterion.

1 Introduction

Effective job scheduling schemes are important for supercomputer centers in order to improve system metrics like utilization, and user metrics like slowdown and turn around time. It is widely accepted that the use of backfilling in job scheduling results in significant improvement to system utilization over non-backfilling scheduling approaches [8]. However, when comparing different backfilling strategies, many studies have concluded that the relative effectiveness of different schemes depends on the job mix [10], [12]. The two main variants are conservative backfilling [6] and aggressive (EASY) [6], [13] backfilling. With conservative backfilling, each job is given a reservation when it arrives in the queue, and jobs are allowed to move ahead in the queue as long as they do not cause any queued job to get delayed beyond its reserved start-time. With aggressive backfilling, only the job at the head of the queue is given a reservation. Jobs are allowed to move ahead of the reserved job as long as they do not delay that job. There is no consensus on which of these two backfilling schemes is better.

In order to gain greater insight into the relative effectiveness of conservative and aggressive backfilling, we group jobs into categories and study their effect on

[⋆] Supported in part by a grant from Sandia National Laboratories jobs in the different categories. Two important factors that affect the scheduling of a job are the length (run time of the job) and width (number of nodes requested by the job). 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 conservative and EASY backfilling. We show that very consistent trends are observed with four different traces from Feitelson's archive [4].

We observe that conservative and aggressive backfilling each benefit certain job categories while adversely affecting other categories. Conservative backfilling allows less backfilling than aggressive backfilling due to the constraints on the schedule by the reservations of all waiting jobs. Although aggressive backfilling enables many more jobs to be backfilled, those jobs (e.g. wide jobs) that do not easily backfill suffer since they might have to wait till they get to the head of the queue before they get a reservation.

We propose a selective reservation scheme intended to obtain the best characteristics from both strategies while avoiding the drawbacks. The main idea is to provide reservations selectively, only to jobs that have waited long enough in the queue. By limiting the number of reservations, the amount of backfilling is greater than conservative backfilling; but by assuring reservations to jobs after a limited wait, the disadvantage of potentially unbounded delay with aggressive backfill is avoided. We show that the new strategy is quite consistently superior to both conservative and aggressive backfilling.

Finally, we address the issue of fairness in job scheduling. We propose a new model for quantitative characterization of fairness in job scheduling and show that the new schemes are comparable or better than aggressive backfilling.

The paper is organized as follows. In Section 2, we provide some background information pertinent to this paper. Section 3 addresses the comparison of conservative and aggressive backfilling. The new selective backfilling schemes are presented and evaluated in Section 4. In Section 5, we develop a new model for characterizing the fairness of a job scheduling scheme. Related work is presented in Section 6. Concluding remarks are provided in Section 7.

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 length is the user estimated run time and width is the number of processors required. Parallel job scheduling strategies have been widely studied in the past [1], [2], [3], [9], [15]. The simplest way to schedule jobs is to use the First-Come-First-Served (FCFS) policy. This approach suffers from low system utilization. Backfilling [11], [12], [16] was proposed to improve system utilization and has been implemented in several production schedulers [7] . Backfilling works by identifying "holes" in the 2D chart and moving forward smaller jobs that fit those holes. There are two common variants to backfilling - conservative and aggressive (EASY)[12], [13]. With 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. With 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.

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 [6] of a job is defined as follows:

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

A threshold of 10 seconds is used to limit the influence of very short jobs (which usually are due to aborted jobs) on the metric.

2.1 Workload Characterization

The simulation studies were performed using a locally developed simulator with workload logs from several supercomputer centers. From the collection of workload logs available from Feitelson's archive [4], the CTC workload trace, the SDSC workload trace, the KTH workload trace and the LANL workload trace were used to evaluate the various schemes. The CTC trace was logged from a 430 node IBM SP2 at the Cornell Theory Center, the KTH trace from a 100 node IBM SP2 system at the Swedish Royal Institute of Technology, the SDSC trace from a 128 node IBM SP2 system at the San Diego Supercomputer Center, and the LANL trace from a 1024 node CM-5 system at the Los Alamos National Laboratory.

Table 1. Job categorization criteria - CTC, KTH and SDSC traces

	≤8 Processors	**>8 Processors**
≤1Hr	SN	SW
>1Hr	LN	LW

Table 2. Job categorization criteria - LANL trace

	≤64 Processors	**>64 Processors**
≤1Hr	SN	SW
>1Hr	LN	LW

Any analysis that is based on the aggregate slowdown of the system as a whole alone does not provide insights into the variability within different job categories. Therefore in our discussion, we classify the jobs into various categories

Table 3. Job distribution by category

Trace	**SN**	**SW**	**LN**	**LW**
CTC	45.06%	11.84%	30.26%	12.84%
KTH	53.78%	19.52%	16.50%	10.20%
SDSC	47.24%	21.44%	20.94%	10.38%
LANL	70.80%	11.72%	9.42%	8.06%

based on the run time 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 under the idealistic assumption of accurate user estimates. In later sections, we present the results using the actual user estimates from the workload logs.

To analyze the performance of jobs of different sizes and lengths, jobs were grouped into 4 categories: based on their run time - Short(S) vs. Long(L); and the number of processors requested - Narrow(N) vs. Wide(W). The criterion used for job classification for the CTC, SDSC and KTH traces are shown in Table 1. For the LANL trace, since no job requested less than 32 processors, the classification criterion shown in Table 2 was used. The distribution of jobs in the various traces, corresponding to the four categories is given in Table 3.

The choice of the partition boundaries for the categories is somewhat arbitrary; however, we show in the next section that the categorization permits us to observe some consistent trends that are not apparent when only the overall averages for the entire trace are computed. We find that the same overall trends are observed if the partition boundaries are changed.

3 Conservative versus EASY backfilling

Previous studies [10], [12] have concluded that the relative performance of EASY and conservative backfill policies is trace and metric dependent and that no consistent trend can be observed. However on finer categorization of the jobs in a trace, consistent category-wise trends become evident under the assumption of exact user run time estimates using FCFS priority.

With conservative backfilling, when a job is submitted, it is given a reservation to start at the earliest time that does not violate any previously existing reservations. The existing reservations constrain later arriving jobs from backfilling easily. The longer the job is, the more difficult it is for it to get a reservation ahead of the 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, even jobs having lower priority than these can backfill ahead of them, if they find enough free processors.

Thus the jobs in the Long Narrow (LN) category benefit from EASY backfilling, while the jobs in the Short Wide (SW) category benefit from conservative backfilling. As far as the Short Narrow (SN) jobs are concerned, there is no consistent trend between EASY and conservative because these jobs backfill very 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 reservations, while EASY backfilling provides these with better backfilling opportunities due to fewer "blockades" in the schedule. Thus the overall performance of EASY versus conservative backfilling will depend on the relative mix of the jobs in each of the categories. Fig. 1 compares the slowdowns and turnaround times of jobs in the different categories, for EASY and conservative backfilling, for the four traces. The average slowdown and turnaround time for EASY backfilling are shown, as a percentage change compared to the corresponding average for the same set of jobs under conservative backfill scheduling. For example, if the average slowdown of jobs in the SW category were 8.0 for conservative backfill and 12.0 for EASY backfill, the bar in the graph would show +50%. Therefore negative values indicate better performance. The figures indicate that the above mentioned trends are observed irrespective of the job trace used and the metric used. Fig. 2 shows a comparison of the two schemes for the CTC trace under high system load (obtained by multiplying each job's run time by a factor of 1.3). We find that the same trends are observed and that differences between the schemes are more pronounced under high load.

The data above highlights the strengths and weaknesses of the two backfilling schemes:

- Conservative backfilling provides reservations to all jobs at arrival time and thus limits the slowdown of jobs that would otherwise have difficulty getting started via backfilling. But it is indiscriminate and provides reservations to all jobs, whether they truly need it or not. By providing reservations to all jobs, the opportunities for backfilling are decreased, due to the blocking effect of the reserved jobs in the schedule.
- EASY backfilling provides a reservation to only the job at the head of the job queue. Thus it provides much more opportunity for backfilling. However, jobs that inherently have difficulty backfilling (e.g. wide jobs) suffer relative to conservative backfilling, because they only get a reservation when they manage to get to the head of the queue.

4 Proposed Schemes

4.1 Selective Reservation Schemes

Instead of the non-selective nature of reservations with both conservative and aggressive backfilling, we propose a selective backfilling strategy: jobs do not get

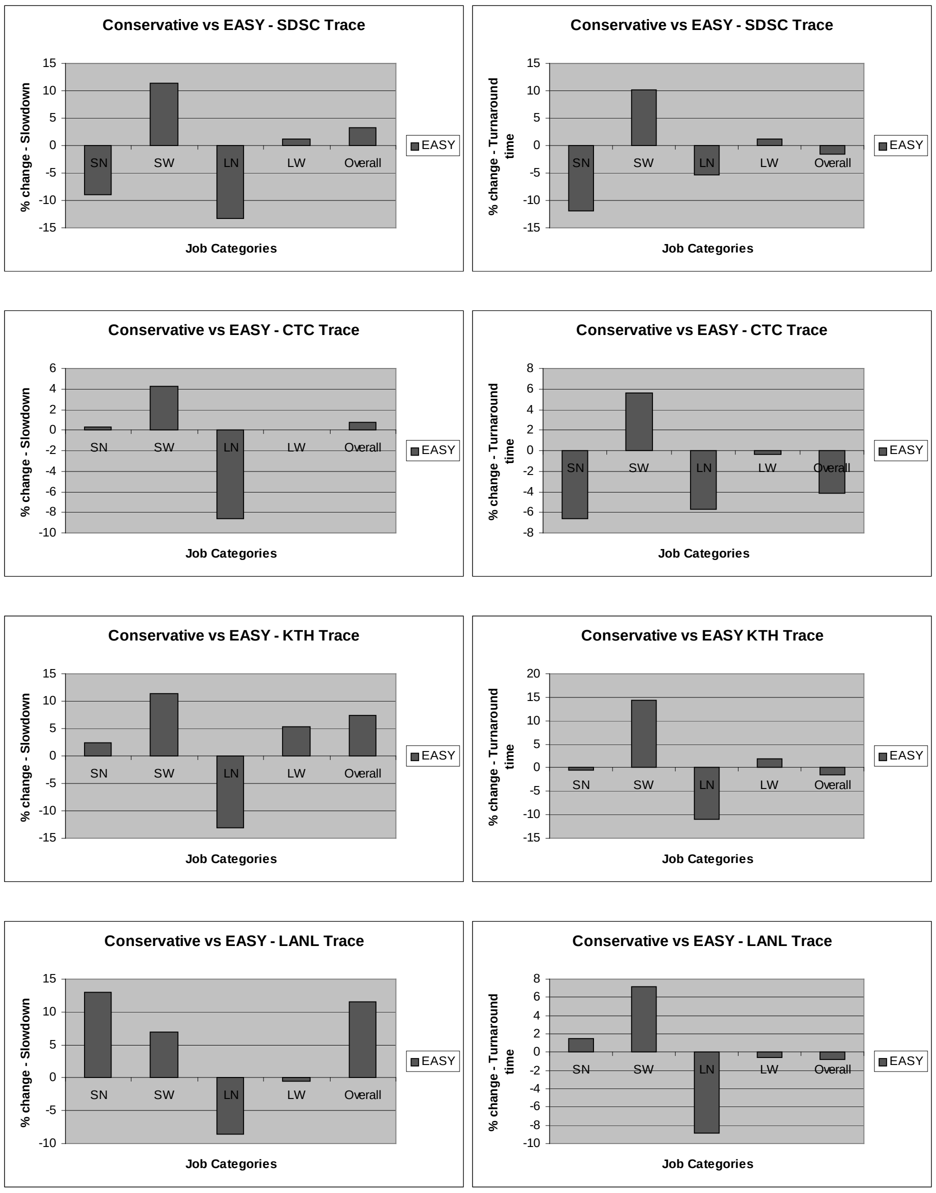
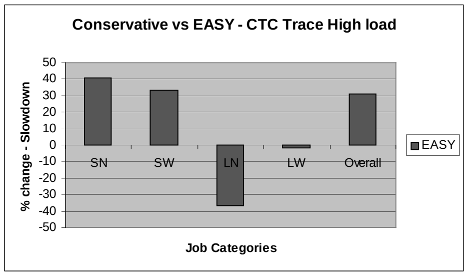


Fig. 1. Category-wise performance comparison of conservative vs. EASY backfilling: normal load. The SW jobs have better slowdowns under conservative backfilling while the LN jobs have better slowdowns under EASY backfilling. This trend is consistent across different traces



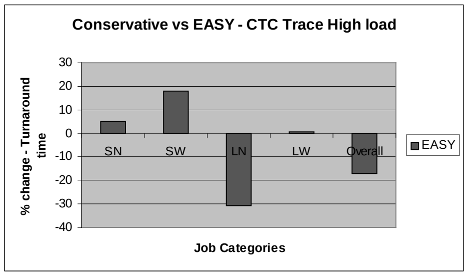


Fig. 2. Comparison of conservative and EASY backfilling: high load. The trends for the SW and the LN jobs are more pronounced under high load compared to normal load

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 are assured of getting reservations.

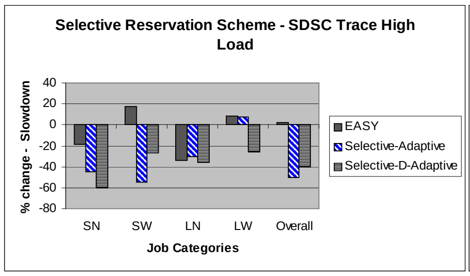
It is convenient to describe the selective reservation approach in terms of two queues with different scheduling policies - an entry "no-guarantee" queue where start time guarantees are not provided and another "all-guaranteed" queue in which all jobs are given a start time guarantee (similar to conservative backfilling). Jobs enter the system through the entry queue which schedules jobs based on FCFS priority without providing start time guarantees. If a job waits long enough in the entry queue, it is transferred to the guaranteed queue. This is done when the eXpansion Factor (XFactor) of the job exceeds some "starvation threshold".

The XFactor of a job is defined as: XFactor = (Wait time + Estimated Run time) / Estimated Run time

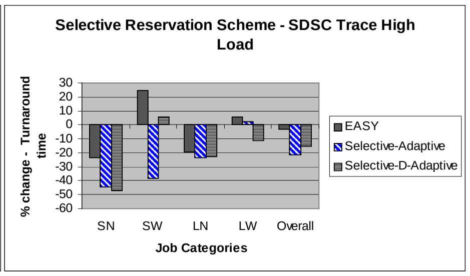
An important issue is that of determination of a suitable starvation threshold. We chose the starvation threshold to simply be the running average slowdown of the previously completed jobs. This is referred to as the *Selective-Adaptive* or *Sel-Adaptive* scheme.

In the Selective-Adaptive scheme, a single starvation threshold is used for all job categories. Since different job categories have very different slowdowns, another variant of selective reservations was evaluated, where different starvation thresholds were used for different job categories, based again on the running average slowdown of the previously completed jobs in each of these categories. We call this the *Selective-Differential-Adaptive* or *Sel-D-Adaptive* scheme. In both the schemes the thresholds are initialized to zero and as jobs complete the running average is updated appropriately. Since different thresholds are used for different job categories, the Selective-D-Adaptive scheme can also be used to tune specific job categories by appropriately scaling the corresponding starvation thresholds. In the rest of the paper Selective backfilling and Selective reservation are used interchangeably.

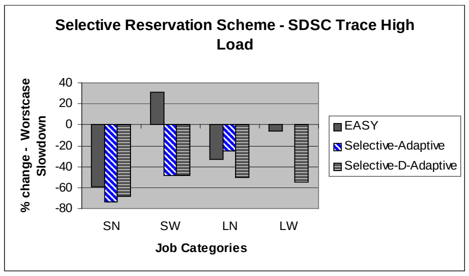
4.2 Performance Evaluation



(a) Average Slowdown



(b) Average Turnaround Time



(c) Worst case Slowdown

Fig. 3. Performance of selective backfilling schemes: accurate user estimates. The selective backfilling schemes achieve a significant reduction in the overall slowdown and turnaround time. The selective schemes also improve the average and worst case slowdowns of most categories

Fig. 3a compares the percentage change in the average slowdowns for the EASY and Selective schemes, with respect to conservative backfilling under high load. It can be observed that the Selective reservation scheme achieves at least 45% reduction in the overall slowdown compared to conservative and EASY backfilling. Further, it improves the slowdowns of all categories compared to conservative backfilling except the LW category, for which there is a slight degradation in slowdown. This degradation in the slowdown for the LW jobs is explained as follows. The LW jobs have difficulty backfilling and hence rely on reservations. Further, the average slowdown for the LW category tends to be much less than the overall average slowdown. Use of the overall average slowdown as the starvation threshold implies that LW jobs will not be moved to the guarantee queue and given a reservation until their XFactor is significantly

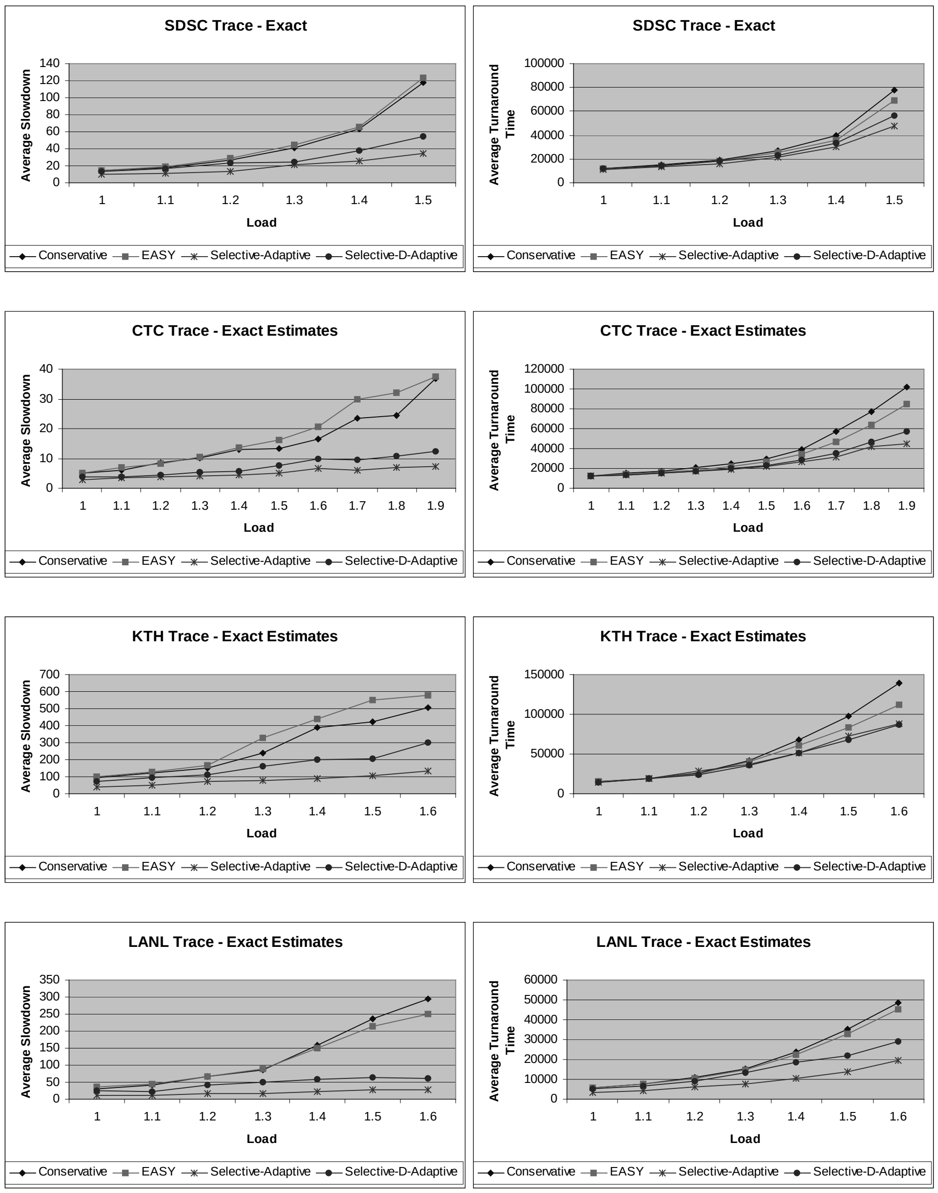


Fig. 4. Performance of the selective schemes for the various traces under different load conditions: exact estimates. The selective reservation schemes outperform conservative and EASY backfilling, especially at high load

higher than their group average. This causes a degradation in the slowdown for the LW category.

The Selective-D scheme improves the performance of all the categories including the LW category, although the magnitude of improvement for the SW category is slightly lower than the Selective scheme.

Similar trends are observed when comparing the turnaround times as indicated in Fig. 3b. From Fig. 3c it can be observed that the Selective-D scheme, achieves dramatic reductions in the worst case slowdowns for all the categories when compared to conservative and EASY backfilling.

Fig. 4 shows the performance of the Selective schemes compared to EASY and conservative backfilling for the various traces under different load conditions. The different loads correspond to modification of the traces by multiplying the run times of the jobs by suitable constants, keeping their arrival time the same as in the original trace. Higher values of the constant represent proportionately higher offered load to the system, in terms of processor-time product. We observe that the improvements obtained by the Selective reservation schemes are more pronounced under high load.

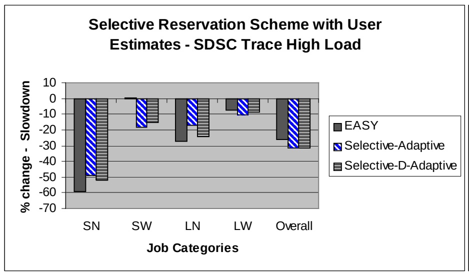
4.3 User Estimate Inaccuracies

We have so far assumed that the user estimates of run time are perfect. Now, we consider the effect of user estimate inaccuracy on the selective reservation schemes. This is desirable from the point of view of realistic modeling of an actual system workload, since a job scheduler only has user run time information to make its scheduling decisions.

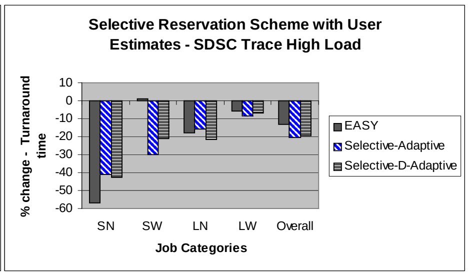
A clarification about these threshold values is in order. Real traces contain a number of aborted jobs and jobs with poorly estimated run times. The slowdowns of these jobs tend to be much larger than the slowdowns of similar well estimated jobs. This is because the large degree of over-estimation of their run time makes these jobs very hard to backfill. Instead of using the average slowdown of all jobs, which tends to be skewed high due to the aborted or poorly estimated jobs, the starvation threshold is computed from the average slowdown of only the well estimated jobs (whose estimated run times are within a factor of two of their actual run times).

Fig. 5a shows the percentage change in the average slowdown for EASY backfill and the selective reservation schemes with respect to conservative backfill. It can be observed from the figure that the selective schemes perform better than conservative backfilling for all job categories. Similar trends can be observed with respect to the average turnaround time from Fig. 5b. Fig. 5c shows the percentage change in the worst case slowdown of the various schemes as a percentage change with respect to that of conservative backfilling.

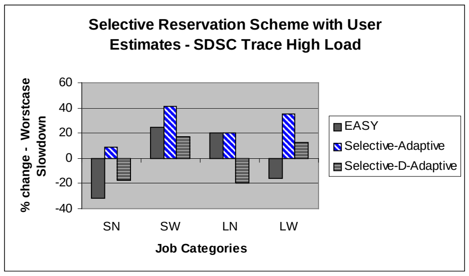
Comparing the Selective-Adaptive schemes with EASY backfill, the improvements are not as good as with exact run time estimates. The jobs with significantly over-estimated run times do not get reservations easily (since their XFactors increase at a slower rate compared to an accurately estimated job of



(a) Average Slowdown



(b) Average Turnaround Time

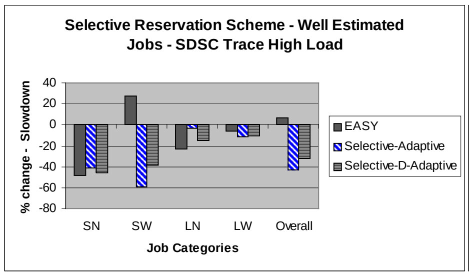


(c) Worst case Slowdown

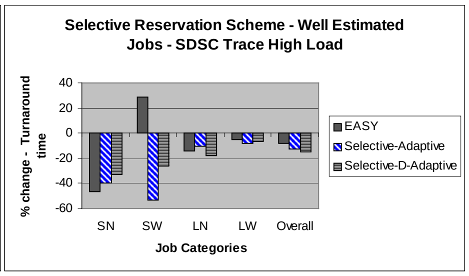
Fig. 5. Performance of selective backfill schemes: actual user estimates. The selective schemes achieve a significant improvement in the average slowdown and turnaround time of all the categories compared to conservative backfilling

the same length) and also cannot backfill easily owing to their seemingly large length. Therefore these jobs tend to incur higher slowdowns with the Selective-Adaptive schemes than under EASY backfill, which provides greater opportunities for these jobs to backfill (because there is no more than one impeding reservation).

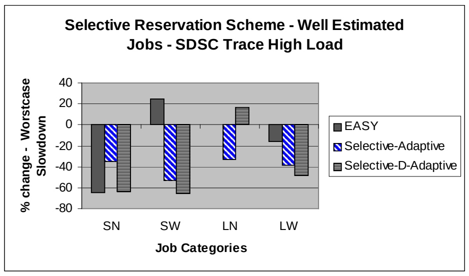
In Fig. 6, we show performance of well-estimated jobs (those with estimated run time within a factor of two of the actual run time). The percentage change in the average slowdown and turnaround time and the worst case slowdown are shown for EASY backfill and the selective reservation schemes, relative to conservative backfill. For well-estimated jobs, the performance trends for the various categories are quite similar to the case of exact run time estimates - the selective schemes are significantly better than conservative backfill, and also better than EASY backfill for most of the cases.



(a) Average Slowdown



(b) Average Turnaround Time



(c) Worst case Slowdown

Fig. 6. Performance of selective backfill schemes: well-estimated jobs

Fig. 7 shows the performance of the Selective schemes compared to EASY and conservative backfilling for the SDSC, CTC and KTH traces under different load conditions. The LANL trace did not contain user run time estimates. We again observe that the improvements obtained by the Selective reservation schemes are more pronounced under high load.

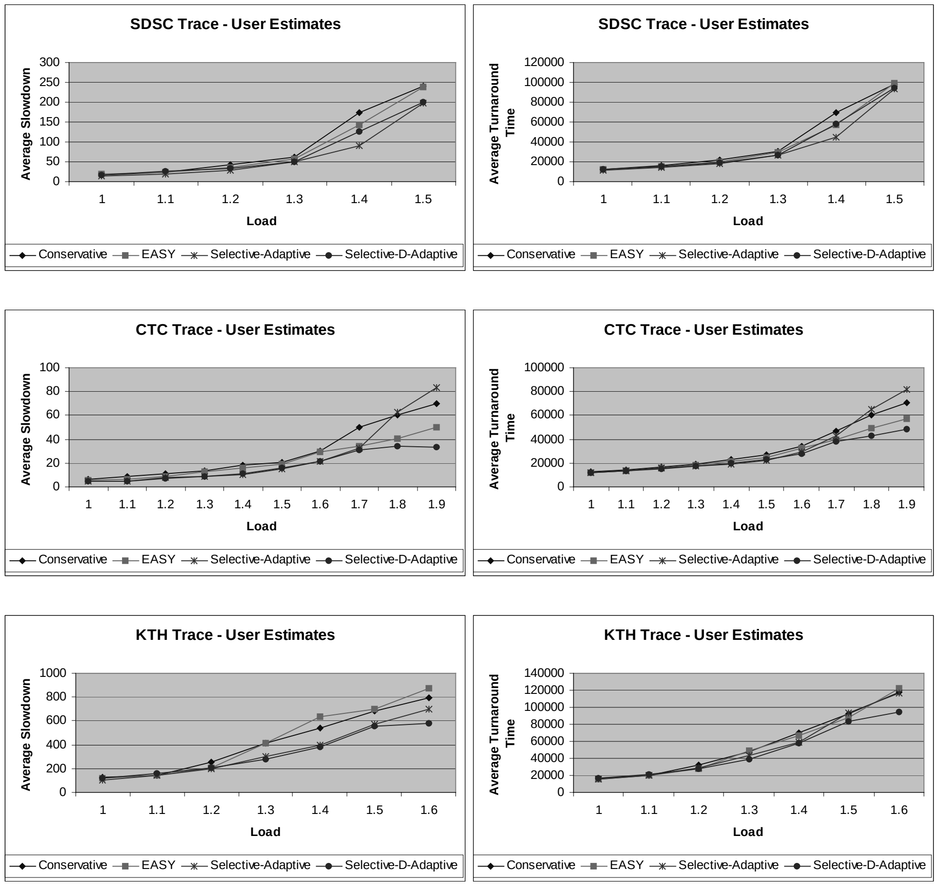


Fig. 7. Performance of the selective schemes for the various traces under different load conditions: actual user estimates. The selective reservation schemes outperform conservative and EASY backfilling, especially under high load

5 Fairness

Of great importance for production job scheduling is the issue of fairness. A strict definition of fairness for job scheduling could be that no later arriving job should be started before any earlier arriving job. Only an FCFS scheduling policy without backfilling would be fair under this strict definition of fairness. Once backfilling is allowed, clearly the strict definition of fairness will be violated. It is well established that backfilling significantly improves system utilization and average slowdown/turnaround-time; thus backfilling is virtually indispensable for non-preemptive scheduling. If we consider FCFS with conservative backfilling under a scenario of perfect estimation of job run times, a weaker definition of fairness is satisfied: No job is started any later than the earliest time it could have been started under the strictly fair FCFS-No-Backfill schedule. In other words, although later arriving jobs may overtake queued jobs, it is not considered unfair because they do not delay queued jobs.

Still considering the scenario of accurate user estimates of run time, how can we evaluate if an alternative scheduling scheme is fair under the above weak criterion? One possibility would be to compare the start time of each job with its start time under the strictly fair FCFS-No-Backfill schedule. However, this is unsatisfactory since the start times of most jobs under FCFS-No-Backfill will likely be worse than FCFS-Conservative, due to the poorer utilization and higher loss-of-capacity with FCFS-No-Backfill. What if we compared start times of each job under the new schedule with the corresponding start time under FCFS-Conservative? This has a problem too - those jobs that got backfilled and leaped ahead under FCFS-Conservative would have a much earlier reference start time than would be fair to compare against. To address this problem, we define a "fair-start" time with each job under a FCFS-Conservative schedule. It is defined as the earliest possible start time the job would have received under FCFS-Conservative if the scheduling strategy were suddenly changed to strict FCFS-No-Backfill at the instant the job arrived. We then define a fair-slowdown of a job as:

Fair-Slowdown = (Fair-Start time under FCFS-Conservative - Queue time + Run time)/(Run time)

We can now quantify the fairness of a scheduling scheme by looking at the percentage of jobs that have a higher slowdown than their fair slowdown. Table 4 shows the percentage of jobs in 5 different groups. The first column indicates the percentage of jobs that have slowdown less than or equal to their fair slowdown value. Column two, indicates the percentage of jobs that have slowdown between 1-1.5 times their fair slowdown value. Column three shows the percentage of jobs that have slowdown between 1.5-2 times their fair slowdown value. Column four indicates the percentage of jobs that have slowdown between 2-4 times their fair slowdown value. Column five shows the percentage of jobs that have slowdown greater than 4 times their fair slowdown value.

From the table, it can be observed that 92% of the jobs received fair treatment under the Selective reservation schemes and the remaining 8% of the jobs had worse slowdown than their fair slowdown and can be considered to have been treated unfairly, relative to FCFS-Conservative. However, it may be observed that the percentage of jobs that got unfair treatment under aggressive backfilling schemes is higher. Compared to SJF-EASY backfilling, the Selective reservation schemes are clearly more fair. But, the percentage of jobs that had slowdown

Table 4. Fairness comparison

	≤*1*	*1-1.5*	*1.5-2*	*2-4*	>*4*
FCFS EASY	90.46	7.20	1.28	0.76	0.30
FCFS Sel-Adaptive	92.64	4.98	0.7	1.04	0.54
FCFS Sel-D-Adaptive	92.18	5.18	1.02	0.88	0.6
SJF EASY	91.08	5.24	1.16	1.18	1.34

greater than twice their fair slowdown value is slightly greater under the selective reservation scheme when compared to FCFS-EASY backfilling.

A scheme that worsens the slowdowns of many jobs in the long categories is not likely to be acceptable even if it improves the slowdowns of most of the other categories. For example, a delay of 1 hour for a 10 minute job (slowdown = 7) is much more tolerable than a slowdown of 7 (i.e. a one-week wait) for a 24 hour job. In order to get insights into how different categories of jobs are treated by the different schemes, we categorized the jobs based on their run time. We compare the number of jobs that received unfair treatment in each of the categories for the different schemes.

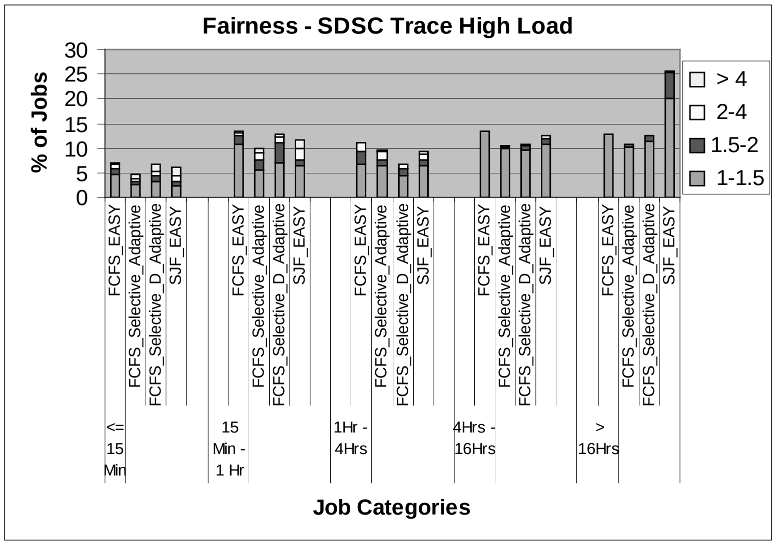


Fig. 8. Fairness comparison of various schemes. The selective backfilling schemes are better than or comparable to FCFS-EASY with respect to fairness

Fig. 8 shows a comparison of the fairness of the selective reservation schemes with FCFS-EASY and SJF-EASY schemes. From the figure we observe that under the Selective reservation schemes, all the jobs that have slowdowns greater than four times their fair slowdown value are short jobs (run time less than or equal to 4Hrs) and none of the very long jobs suffer a degradation greater than two times their fair slowdown value. For most length categories, the number of unfairly treated jobs is less with the selective reservation schemes than the aggressive backfilling schemes. Overall, we can conclude that the new schemes are better than or comparable to FCFS-EASY with respect to fairness. FCFS-EASY is a widely used scheduling strategy in practice - thus the new selective scheduling schemes would appear to be very attractive, since they have better performance and comparable/better fairness properties.

The above model for fairness was based on the observation that FCFS-Conservative satisfies a weak fairness property and therefore the fair-start time of jobs under FCFS-Conservative can be used as a reference to compare the start-times with other schedules. Of course, in practice user estimates of run time are not accurate, and in this scenario, even the weak definition of fairness is not satisfied by FCFS-Conservative schedules. Nevertheless, FCFS-Conservative is considered completely acceptable as a scheduling scheme from the viewpoint of fairness. Hence we believe it is appropriate to use it as a reference standard in evaluating the fairness of other schedules in the practical scenario of inaccurate user estimates of run time.

6 Related Work

The relative performance of EASY and conservative backfilling is compared in [5] using different workload traces and metrics. A conclusion of the study is that the relative performance of conservative and EASY backfilling depends on the percentage of long serial jobs in the workload and the accuracy of user estimates. It is observed that if user estimates are very accurate and the trace contains many long serial jobs, then conservative backfilling degrades the performance of the long serial jobs and enhances the performance of the larger short jobs. This is consistent with our observations in this paper.

In [14], the effect of backfill policy and priority policy on different job categories was evaluated. A conclusion of the study is that when actual user estimates are used, the average slowdown of the well estimated jobs decreases compared to their average slowdown when all user estimates are accurate. Poorly estimated jobs on the other hand, have worse slowdowns compared to when all user estimates are accurate. This effect is more pronounced under conservative backfilling compared to EASY.

Other studies that have sought approaches to improve on standard backfilling include [9], [16]. In [16], an approach is developed where each job is associated with a deadline (based on its priority) and a job is allowed to backfill provided it does not delay any job in the queue by more than that job's slack. Such an approach provides greater flexibility to the scheduler compared to conservative backfilling while still providing an upper bound on each job's actual start time. In [9], it is shown that systematically lengthening the estimated execution times of all jobs results in improved performance of backfilling schedulers. Another scheme evaluated via simulation in [9] is to sort the waiting queue by length and provide no start-time guarantees. But this approach can result in very high worst case delays and potentially lead to starvation of jobs.

7 Conclusions

In this paper we used trace-based simulation to characterize the relative performance of conservative and aggressive backfilling. We showed that by examining the performance within different job categories, some very consistent trends can be observed across different job traces. We used the insights gleaned from the characterization of conservative and aggressive backfilling to develop a new selective backfilling approach. The new approach promises to be superior to both aggressive and conservative backfilling. We also developed a new model for characterizing the fairness of a scheduling scheme, and showed that the new schemes perform comparably or better than aggressive backfilling schemes.

Acknowledgments

We thank the anonymous referees for the numerous suggestions for improving the paper, especially "Referee 4" for his/her extensive comments and suggestions.

References

1. K. Aida. Effect of Job Size Characteristics on Job Scheduling Performance. In *Workshop on Job Scheduling Strategies for Parallel Processing*, 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. P. J. Keleher D. Perkovic. Randomization, Speculation, and Adaptation in Batch Schedulers. In *Supercomputing*, 2000.
4. D. G. Feitelson. Logs of real parallel workloads from production systems. http://www.cs.huji.ac.il/labs/parallel/workload/logs.html.
5. D. G. Feitelson. Analyzing the Root Causes of Performance Evaluation Results. Technical report, Leibniz Center, Hebrew University, 2002.
6. D. G. Feitelson, L. Rudolph, U. Schwiegelshohn, K. C. Sevcik, and P. Wong. Theory and Practice in Parallel Job Scheduling. In *Workshop on Job Scheduling Strategies for Parallel Processing* , pages 1–34. 1997.
7. D. Jackson, Q. Snell, and M. J. Clement. Core Algorithms of the Maui Scheduler. In *Wkshp. on Job Sched. Strategies for Parallel Processing*, pages 87–102, 2001.
8. J. P. Jones and B. Nitzberg. Scheduling for Parallel Supercomputing: A Historical Perspective of Achievable Utilization. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 1–16, 1999.
9. P. J. Keleher, D. Zotkin, and D. Perkovic. Attacking the Bottlenecks of Backfilling Schedulers. *Cluster Computing*, 3(4):245–254, 2000.
10. J. Krallmann, U. Schwiegelshohn, and R. Yahyapour. On the Design and Evaluation of Job Scheduling Algorithms. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 17–42, 1999.

11. D. Lifka. The ANL/IBM SP Scheduling System. In *Workshop on Job Scheduling Strategies for Parallel Processing*, pages 295–303, 1995.
12. A. W. Mu'alem and D. G. Feitelson. Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling. In *IEEE Trans. Par. Distr. Systems*, volume 12, pages 529–543, 2001.
13. J. Skovira, W. Chan, H. Zhou, and D. Lifka. The EASY - LoadLeveler API Project. In *Wkshp. on Job Sched. Strategies for Parallel Processing*, pages 41–47, 1996.
14. S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan. Characterization of Backfilling Strategies for Parallel Job Scheduling. In *Proceedings of the ICPP-2002 Workshops*, pages 514–519, 2002.
15. A. Streit. On Job Scheduling for HPC-Clusters and the dynP Scheduler. In *Proc. Intl. Conf. High Perf. Comp.*, pages 58–67, 2001.
16. D. Talby and D. G. Feitelson. Supporting Priorities and Improving Utilization of the IBM SP Scheduler Using Slack-Based Backfilling. In *Proceedings of the 13th International Parallel Processing Symposium*, 1999.