# Selective Reservation Strategies for Backfill Job Scheduling*

Srividya Srinivasan Rajkumar Kettimuthu Vijay Subramani P. Sadayappan

*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 - 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 super computer 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 [10]. However, when comparing different backfilling strategies, many studies have concluded that the relative effectiveness of different schemes depends on the job mix [12, 2]. The two main variants are conservative backfilling [4] and aggressive (EASY) [4, 11] backfilling. In 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 their 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 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 [9].

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 backfill-

*This research was supported in part by Sandia National Laboratories

ing.

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. Concluding remarks are provided in Section 6.

# 2 Background and Workload Characterization

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 [13, 14, 15, 7, 1] .The simplest way to schedule jobs is to use the First-Come-First-Served (FCFS) policy. This approach suffers from low system utilization. Backfilling [2, 8, 6] was proposed to improve system utilization and has been implemented in several production schedulers [5] . 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)[2, 11]. 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.

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 [4] 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.

## 2.1 Workload Characterization

The simulation studies were performed using a locally developed scheduler with workload logs from several supercomputer centers. From the collection of workload logs available from Feitelson's archive [9], 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 is from the 430 node Cornell Theory Center. The KTH trace is from the 100 node IBM SP2 system at the Swedish Royal Institute of Technology. The SDSC trace is from the 128 node IBM SP2 system at the San Diego Supercomputer Center. The LANL trace is from the 1024 node CM-5 system at the Los Alamos National Laboratory.

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 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 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 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 for the CTC, SDSC and KTH traces is 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 Tables 3,4,5 and 6.

**Job Categorization Criteria**

| | **$\leq$8 Processors** | **$>$8 Processors** |
|---|---|---|
| **$\leq$1Hr** | SN | SW |
| **$>$1Hr** | LN | LW |

Table 1: Categorization of jobs based on their Runtime and Width for CTC,KTH and SDSC traces

| | **$\leq$64 Processors** | **$>$64 Processors** |
|---|---|---|
| **$\leq$1Hr** | SN | SW |
| **$>$1Hr** | LN | LW |

Table 2: Categorization of jobs based on their Runtime and Width for the LANL trace

**Job Distribution for Various Traces**

The choice of the partition boundaries for the categories is somewhat arbitrary; however, we show in the next

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

Table 3: CTC Trace

| | N | W |
|---|---|---|
| **S** | 53.78% | 19.52% |
| **L** | 16.50% | 10.20% |

Table 4: KTH Trace

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

Table 5: SDSC Trace

| | N | W |
|---|---|---|
| **S** | 70.80% | 11.72% |
| **L** | 9.42% | 8.06% |

Table 6: LANL Trace

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 [12, 2] 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.

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 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.

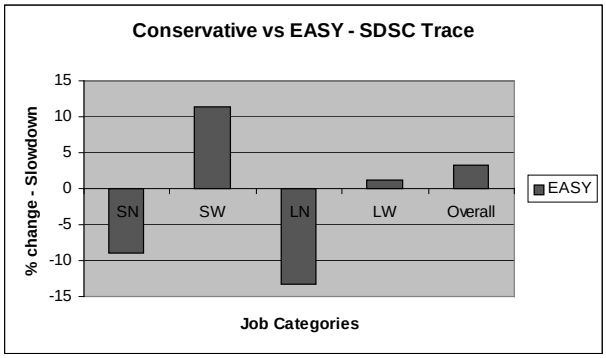
Wide jobs find it difficult to backfill because they cannot find enough free processors easily. Conservative backfill helps such wide jobs by guaranteeing them a start time when they enter the system. In EASY backfill, since these jobs are not given a reservation until they reach the head of the idle 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 four different traces. The average slowdown (or turnaround time) for EASY backfilling is 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 shrinking the job interarrival times). We find that the same trends are observed and that differences between the schemes are more pronounced under high load.

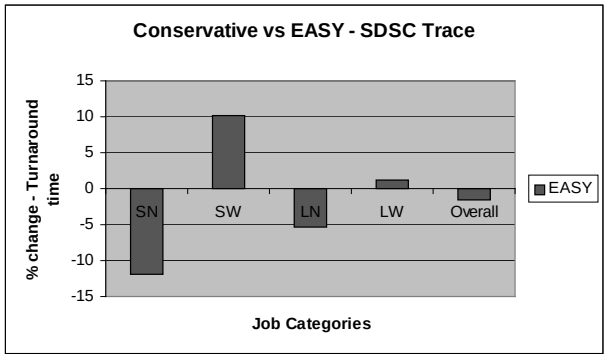
The data above highlights the strengths and weaknesses of the two backfilling schemes:

- 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.

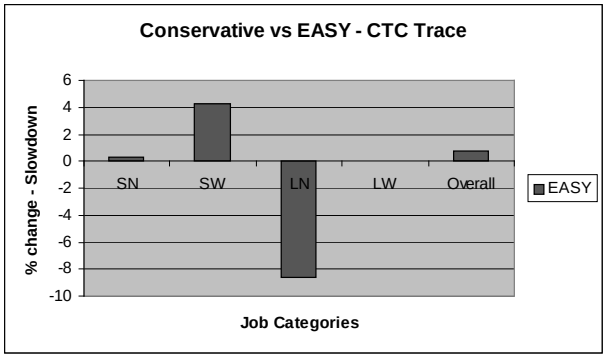
- EASY backfilling provides a reservation to only the



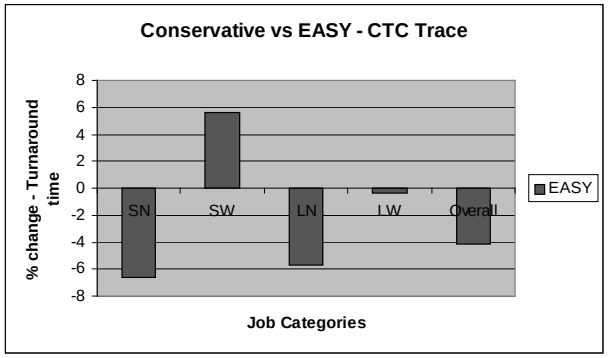

(a) SDSC Slowdown



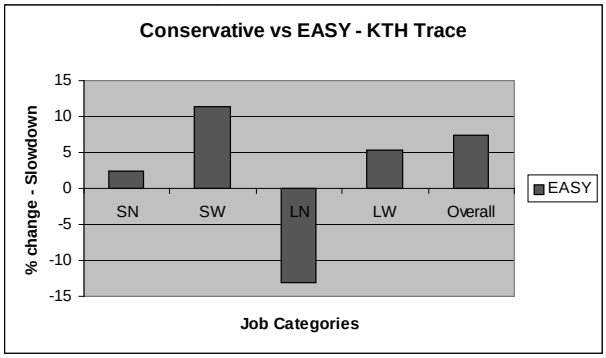

(b) SDSC Turnaround Time



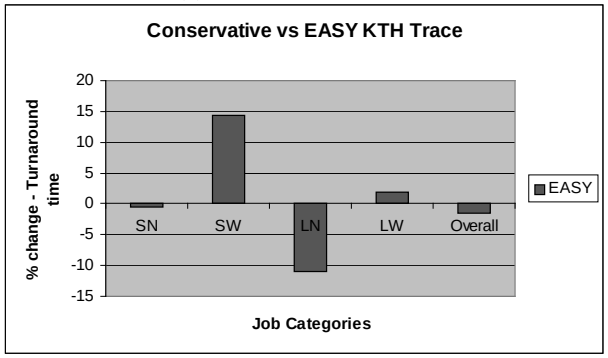

(c) CTC Slowdown



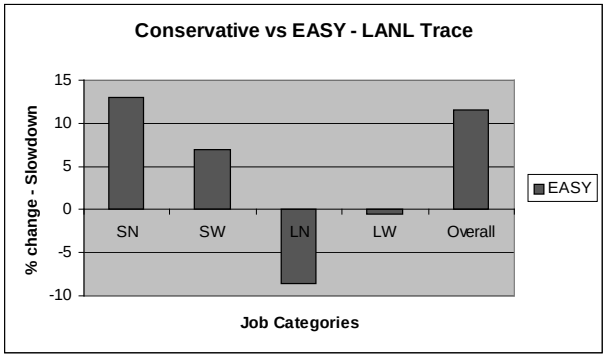

(d) CTC Turnaround Time



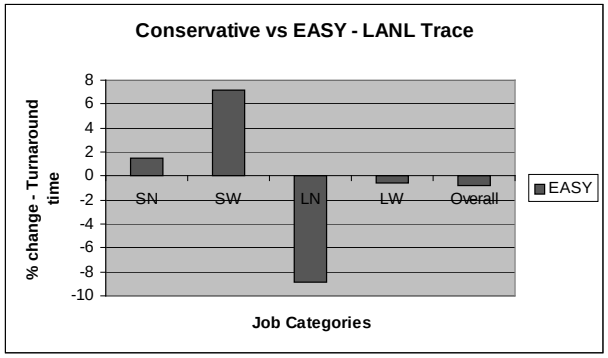

(e) KTH Slowdown




(f) KTH Turnaround Time

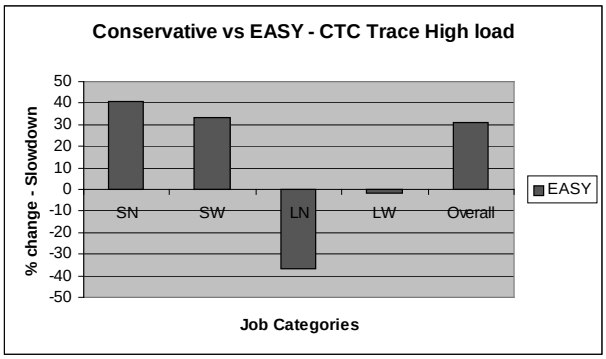



(g) LANL Slowdown

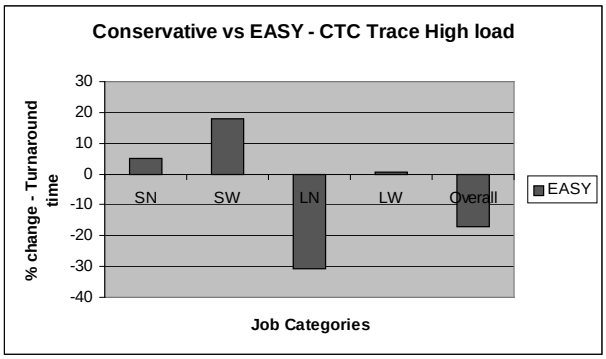



(h) LANL Turnaround Time

Figure 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.




(a) CTC Slowdown High Load




(b) CTC Turnaround Time High Load

Figure 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.

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 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. In order to assess the idea of using such a selective reservation strategy, we chose the starvation threshold to simply be the average slowdown of the jobs of the trace when conservative backfilling is used to schedule the jobs. This is referred to as the *Selective* or *Sel* scheme.

In the Selective 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 statistics gathered from a simulation of the trace with conservative backfilling. We call this the *Selective-Differential* or *Sel-D* scheme. Since different thresholds are used for different job categories, the Selective-D scheme can also be used to tune specific job categories by appropriately selecting the starvation thresholds. In the rest of the paper Selective Backfilling and Selective Reservation are used interchangeably.

## 4.2 Performance Evaluation

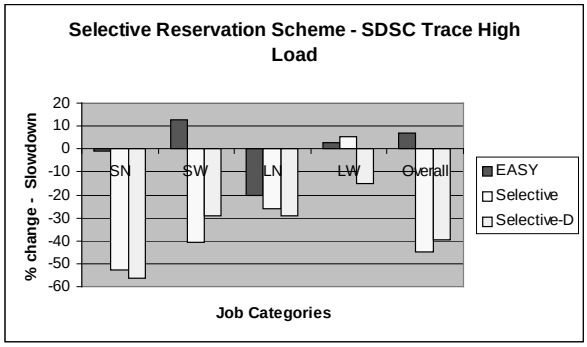
| | **N** | **W** |
|---|---|---|
| **S** | 22.30 | 72.40 |
| **L** | 1.72 | 2.59 |

Table 7: Category-wise thresholds used for the Sel-D scheme. A threshold of 26.69 was used for the Sel scheme

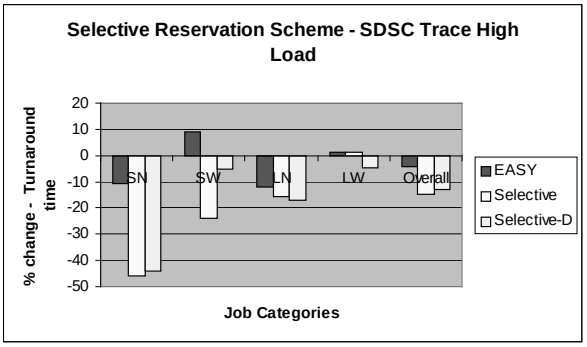
For the Selective scheme, the overall average slowdown obtained from a previous run of conservative backfill scheduling was used as the starvation threshold.

For the Selective-D scheme, the category based average slowdown from conservative backfill scheduling was used as the starvation threshold for each of the categories.

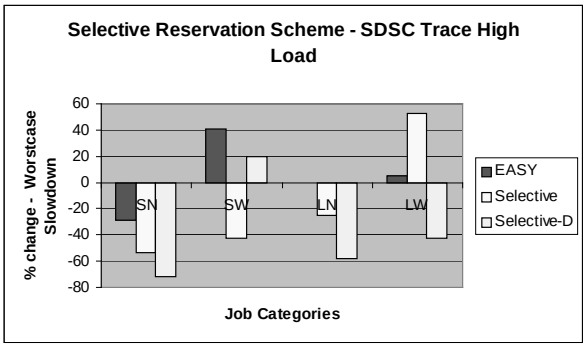
The threshold values used for the Selective schemes




(a) Slowdown

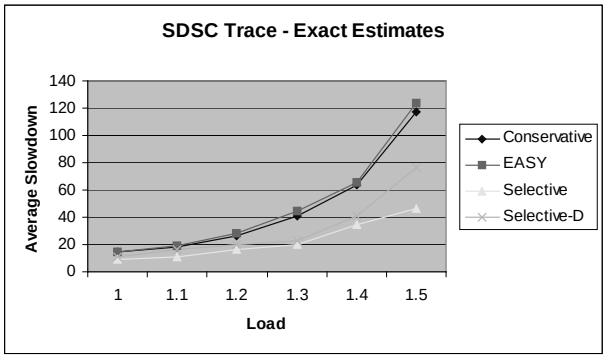



(b) Turnaround Time

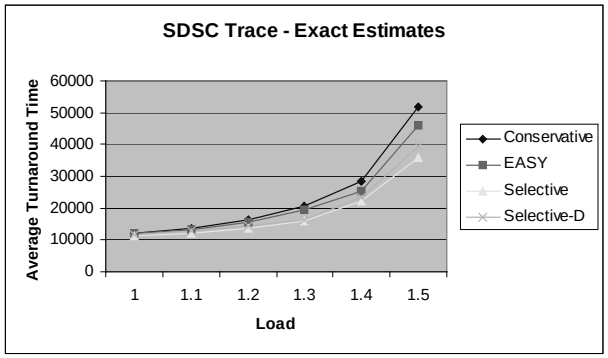



(c) Worst case Slowdown

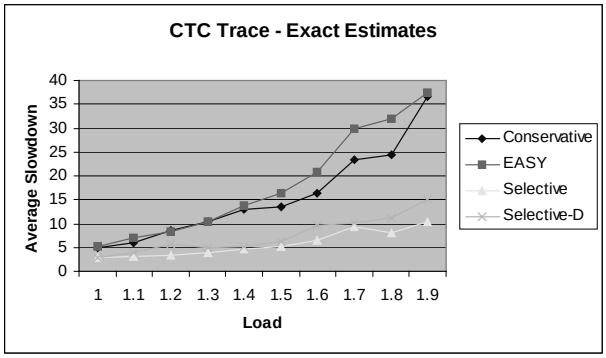
Figure 3: Performance of Selective Backfilling Schemes: Accurate User Estimates. The Selective backfilling schemes achieve a significant reduction in the overall slowdown and turnaround time. The Selective schemes also improve the average and worst case slowdowns of most categories.



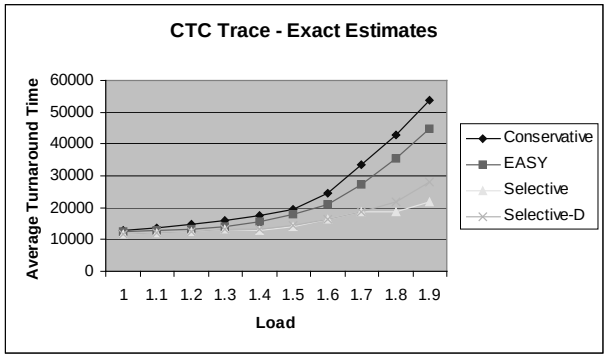

(a) SDSC Slowdown



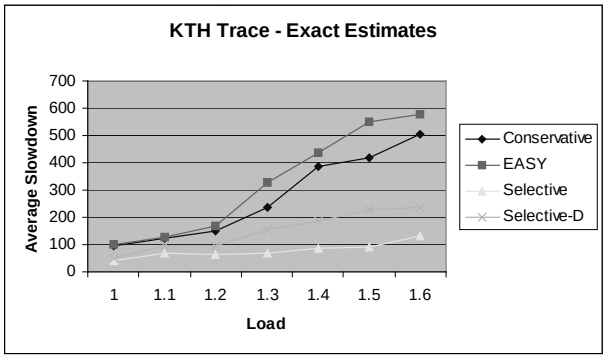

(b) SDSC Turnaround Time



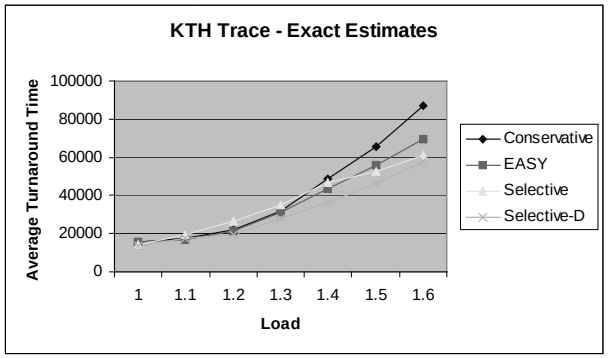

(c) CTC Slowdown



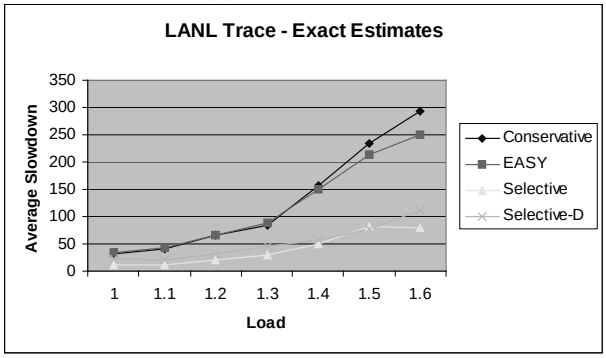

(d) CTC Turnaround Time



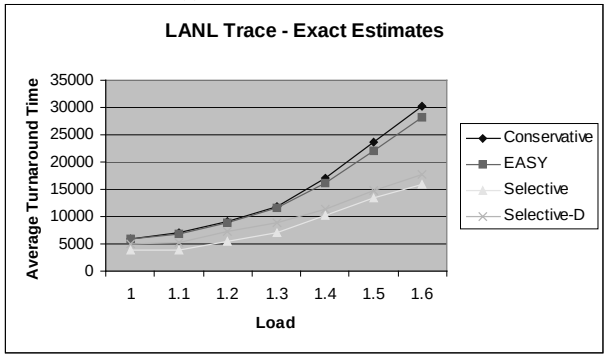

(e) KTH Slowdown




(f) KTH Turnaround Time




(g) LANL Slowdown




(h) LANL Turnaround Time

Figure 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 under high load.

are shown in Table 7.

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 EASY and 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 (derived from a previous run) 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 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 improvements in the other categories is slightly lower than the Selective scheme.

Similar trends are observed when comparing the turnaround times and worst case slowdowns as indicated in Fig.3b and 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 EASY backfilling. Further it has much lower worst case slowdowns when compared to conservative backfilling for all categories except the SW category.

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 dividing the arrival times of the jobs by suitable constants, keeping their run time the same as in the original trace. Thus higher values of the constant represent a higher inter job arrival rate and hence a higher load. 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 runtime 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 runtime information to make its scheduling decisions.

Table 8 shows the threshold values used for the Selective schemes. 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 well estimated jobs. This is because the large degree of over-estimation of their runtime 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 actual run times are within a factor of two of their estimated run times).

| | **N** | **W** |
|---|---|---|
| **S** | 3.55 | 21.53 |
| **L** | 1.35 | 2.27 |

Table 8: Category-wise thresholds used for the Sel-D scheme. A threshold of 3.92 was used for the Sel scheme

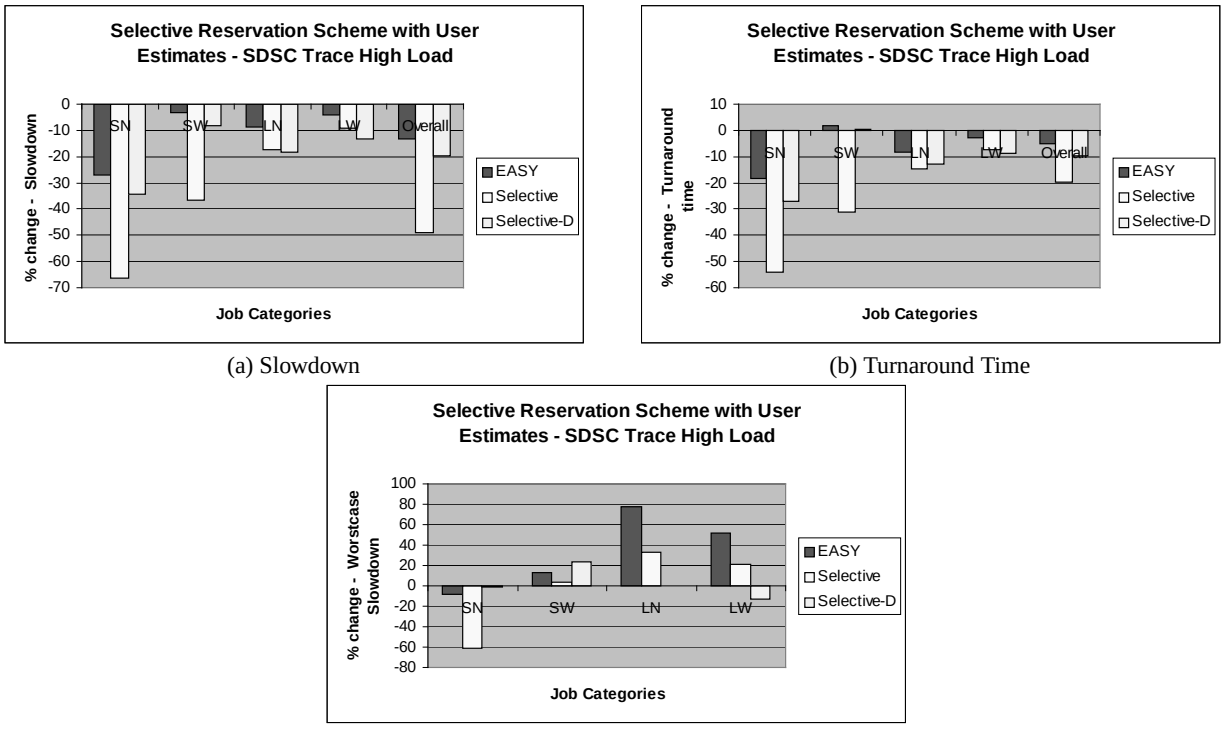
Fig.5a shows the percentage change in the average slowdown for EASY and the selective reservation schemes with respect to conservative backfill. It can be observed from the figure that the selective schemes clearly perform much better than conservative and EASY backfilling for all job categories. Under the Selective scheme the slowdowns of the short jobs reduces dramatically. Under the Selective-D scheme, the reduction in slowdown for the short jobs is not as high as under the Selective scheme for the same reasons as explained previously. But the Selective-D scheme improves the slowdowns of the long jobs to a greater extent than the Selective scheme. Similar trends can be observed with respect to the average turnaround time from Fig.5b.

As can be observed from Fig.5c, the Selective-D scheme achieves a significant improvement in the worst case slowdowns for all the categories compared to conservative and EASY backfilling, except the SW category which shows a slight deterioration.

Fig.6 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.

## 5 Fairness

Of great concern for production job scheduling are the issues of fairness and starvation-freedom. Freedom from starvation must be guaranteed by a scheduling scheme if it is to get serious adoption consideration. For example,




(a) Slowdown

(b) Turnaround Time

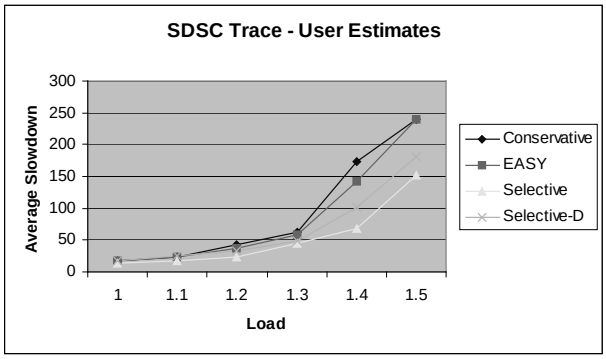
(c) Worst case Slowdown

Figure 5: Performance of Selective Backfill Schemes: Inaccurate User Runtime Estimates. The Selective schemes achieve a significant improvement in the average slowdown and turnaround time of all the categories.

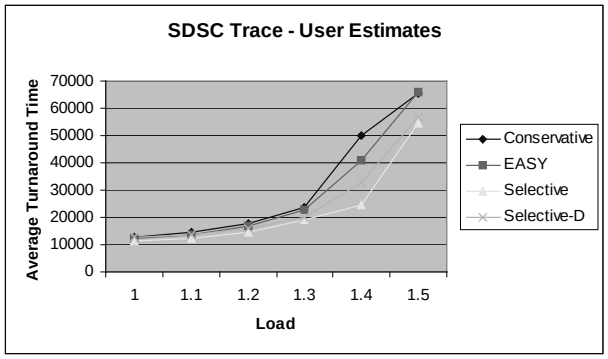
a Shortest-Job-First (SJF) scheduling priority scheme used in conjunction with EASY backfilling achieves lower average job slowdown than FCFS-EASY. However, SJF-EASY cannot guarantee freedom from starvation (as with the rest of this paper, we are only considering non-preemptive scheduling here). Hence supercomputer centers are unlikely to adopt SJF-EASY as their scheduling strategy even if it has demonstrably better performance with respect to the metrics of utilization, slowdown and turnaround time. The scheme that we propose in this paper is clearly starvation-free: after a job has waited till its XFactor exceeds the starvation threshold, it will get a reservation. Next we discuss the issue of fairness, which is much more complicated than the starvation issue.

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-NoBackfill schedule. In other words, although later arriving jobs may overtake queued jobs, it is not considered unfair because they do not delay queued jobs.

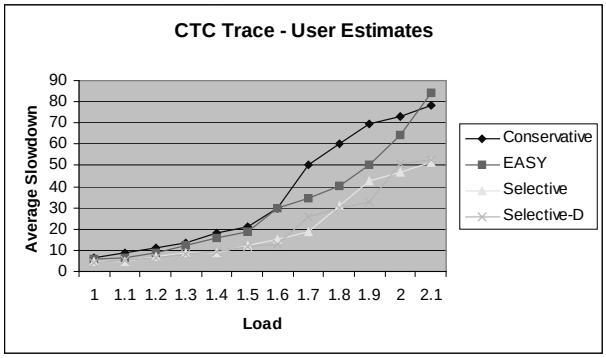
Still considering the scenario of accurate user estimates of runtime, how can we evaluate if an alternative scheduling scheme is fair under the above weak criterion? One possibility would be to compare the start time of each job with its start time under the strictly fair FCFS-NoBackfill schedule. However, this is unsatisfactory since the start times of most jobs under FCFS-NoBackfill will likely be worse than FCFS-Conservative, due the poorer utilization and higher loss-of-capacity with FCFS-NoBackfill. 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



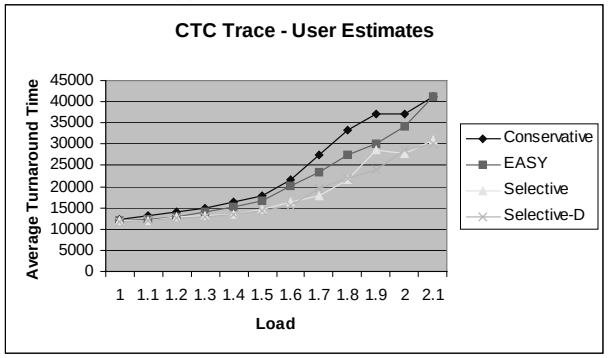

(a) SDSC Slowdown



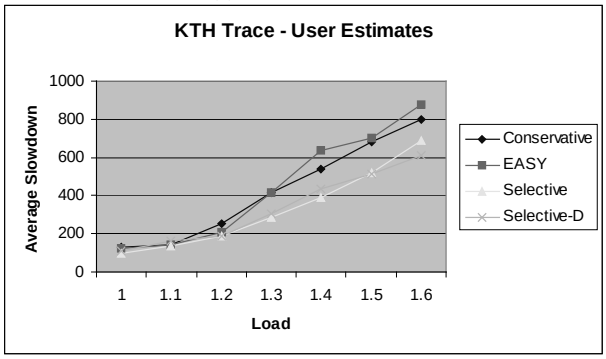

(b) SDSC Turnaround Time



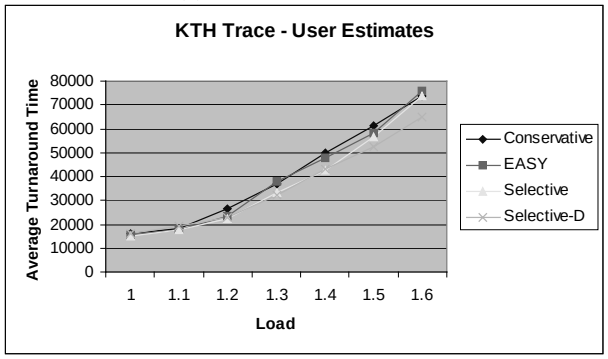

(c) CTC Slowdown




(d) CTC Turnaround Time




(e) KTH Slowdown




(f) KTH Turnaround Time

Figure 6: 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.

| | *≤1* | *1-1.5* | *1.5-2* | *2-4* | *>4* |
|---|---|---|---|---|---|
| *FCFS EASY* | 90.46 | 7.20 | 1.28 | 0.76 | 0.30 |
| *FCFS Selective* | 93.48 | 3.98 | 0.78 | 0.94 | 0.82 |
| *FCFS Selective-D* | 93.22 | 4.64 | 0.92 | 0.88 | 0.34 |
| *SJF EASY* | 91.08 | 5.24 | 1.16 | 1.18 | 1.34 |

Table 9: Fairness Comparison

suddenly changed to strict FCFS-NoBackfill 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 9 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 93% of the jobs received fair treatment under the Selective reservation schemes and the remaining 7% 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 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 runtime. We compare the number of jobs that received unfair treatment in each of the categories for the different schemes.

Fig.7 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 1Hr) 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 runtime 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 runtime.

# 6 Related Work

The relative performance of EASY and conservative backfilling is compared in [3] 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 [16], 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

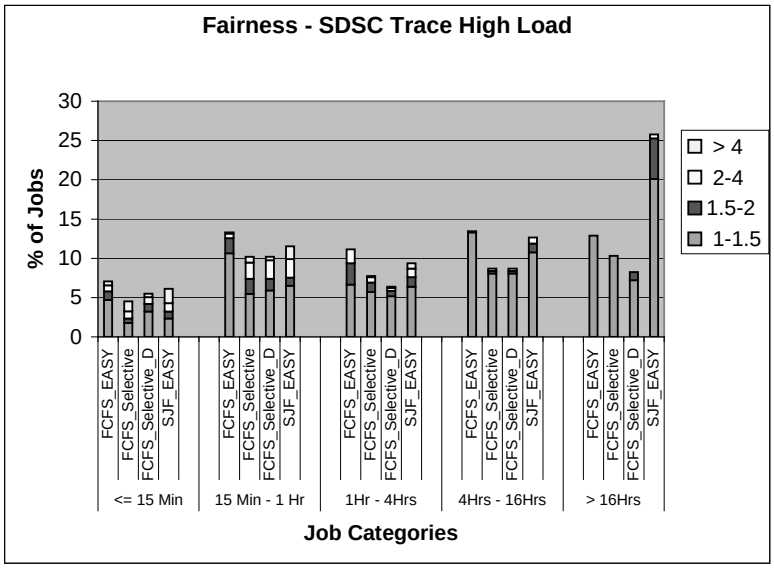



Figure 7: Fairness comparison of various schemes. The Selective backfilling schemes are better than or comparable to FCFS-EASY with respect to fairness.

on standard backfilling include [8, 15]. In [8], 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 [15], 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 [15] is to sort the waiting queue by length but provide no start-time guarantees. But this approach is unlikely to be adopted by production job schedulers because it can potentially lead to starvation.

# 7 Discussion and 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.

An issue of continuing investigation is that of determining the starvation threshold for selective reservation. The results shown here were based on thresholds derived from the average slowdown obtained by first running the same trace using conservative backfilling. Although this was valuable in demonstrating the effectiveness of the selective backfilling approach, it is clearly not a practically implementable strategy in a real on-line scheduler. An adaptive approach was evaluated, where the starvation threshold was obtained by dynamically maintaining a running average of the slowdowns of previously completed jobs. Compared to the use of the starvation threshold from a conservative backfill run, the results from the adaptive threshold run were worse (although still comparable or better than standard EASY or conservative backfilling). Further, the running average slowdown tended to oscillate over time, alternately increasing and decreasing.

The oscillatory behavior may be explained as follows. A lower average slowdown with selective backfilling (compared to conservative backfilling) lowers the starvation threshold, causing more jobs to get reservations. The larger number of reservations inhibits effective backfilling, and the schedule tends to get similar to one with conservative backfilling, causing the average slowdown to increase. This increases the starvation threshold, which reduces the number of jobs that get reservations, thereby improving backfilling and reducing average slowdown.

After observing this oscillatory behavior with the adaptive strategy for maintaining the starvation thresholds, we tried a "scaled-adaptive" approach, where the starvation threshold was set by scaling the average slowdown by

a scaling factor, say 1.5. This had the effect of reducing the range of oscillatory behavior and improved performance. The approach seems promising, but the effect of varying the scaling factor and the effect on different traces is still being evaluated.

## 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] A. Streit. On Job Scheduling for HPC-Clusters and the dynP Scheduler. In *HiPC*, pages 58–67, 2001.

[2] 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.

[3] D. G. Feitelson. Analyzing the Root Causes of Performance Evaluation Results. Technical report, Leibniz Center, Hebrew University, 2002.

[4] D. G. Feitelson and Rudolph and U. Schwiegelshohn and K. C. Sevcik and Wong. Theory and Practice in Parallel Job Scheduling. In *Feitelson & Rudolph (Eds.), Job Scheduling Strategies for Parallel Processing: IPPS '95 Workshop, Springer LNCS 949.* 1997.

[5] D. Jackson and Q. Snell and M. J. Clement. Core Algorithms of the Maui Scheduler. In *JSSPP*, pages 87–102, 2001.

[6] D. Lifka. The ANL/IBM SP Scheduling System. In *JSSPP*, pages 295–303, 1995.

[7] D. Perkovic and P. J. Keleher. Randomization, Speculation, and Adaptation in Batch Schedulers. In *Supercomputing*, 2000.

[8] D. Talby and D. 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.

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

[10] J. P. Jones and B. Nitzberg. Scheduling for Parallel Supercomputing: A Historical Perspective of Achievable Utilization. In *JSSPP*, pages 1–16, 1999.

[11] J. Skovira and W. Chan and H. Zhou and D. Lifka . The EASY - LoadLeveler API Project. In *JSSPP*, pages 41–47, 1996.

[12] Jochen Krallmann and Uwe Schwiegelshohn and Ramin Yahyapour. On the Design and Evaluation of Job Scheduling Algorithms. In *JSSPP*, pages 17–42, 1999.

[13] K. Aida. Effect of Job Size Characteristics on Job Scheduling Performance. In *JSSPP*, pages 1–17, 2000.

[14] O. Arndt and B. Freisleben and T. Kielmann and F. Thilo. A Comparative Study of Online Scheduling Algorithms for Networks of Workstations. *Cluster Computing*, 3(2):95–112, 2000.

[15] P. J. Keleher and D. Zotkin and D. Perkovic. Attacking the Bottlenecks of Backfilling Schedulers. *Cluster Computing*, 3(4):245–254, 2000.

[16] S. Srinivasan and R. Kettimuthu and V. Subramani and P. Sadayappan. Characterization of Backfilling Strategies for Parallel Job Scheduling. To appear in Proceedings of the ICPP Workshop on Scheduling and Resource Management for Cluster Computing, 2002.