# Unfairness Metrics for Space-Sharing Parallel Job Schedulers

Gerald Sabin and P. Sadayappan

The Ohio State University, Columbus OH 43201, USA
{sabin, saday}@cse.ohio-state.edu

**Abstract.** Sociology, computer networking and operations research provide evidence of the importance of fairness in queuing disciplines. Currently, there is no accepted model for characterizing fairness in parallel job scheduling. We introduce two fairness metrics intended for parallel job schedulers, both of which are based on models from sociology, networking, and operations research. The first metric is motivated by social justice and attempts to measure deviation from arrival order, which is perceived as fair by the end user. The second metric is based on resource equality and compares the resources consumed by a job with the resources deserved by the job. Both of these metrics are orthogonal to traditional metrics, such as turnaround time and utilization.

The proposed fairness metrics are used to measure the unfairness for some typical scheduling policies via simulation studies. We analyze the fairness of these scheduling policies using both metrics, identifying similarities and differences.

## 1 Introduction

There has been considerable research [1, 2, 3, 4] focused on various aspects of job schedulers for parallel machines, much of it being reported at the annual Workshop on Job Scheduling Strategies for Parallel Processing [5]. These studies have generally focused on user performance metrics (such as turnaround time and slowdown) and system metrics (such as utilization and loss of capacity).

Scheduling algorithms have been developed with the intent of providing users better functionality and enhanced performance, while improving system utilization. But the issue of fairness has not been much addressed in the context of job scheduling. We believe that this is in large part due to the lack of accepted metrics for characterizing fairness. Whereas performance metrics such as wait time, response time, slowdown and utilization are widely accepted and used by the job scheduling research community, fairness metrics are not well established. Researchers have analyzed worst case and 95th percentile performance data for the more standard user metrics [6, 7] in an attempt to indirectly characterize scheduling schemes with respect to the issues of starvation and fairness. However, there has been very little work which proposes direct fairness metrics for job schedulers.

© Springer-Verlag Berlin Heidelberg 2005

In contrast to the work in the job scheduling community, there is a body of existing work on the topic of fairness in fields such as sociology, computer network scheduling and queuing theory. The Resource Allocation Queuing Fairness Metric (RAQFM) [8, 9, 10] is a model for assessing fairness in queuing systems. RAQFM evenly divides available resources amongst all jobs to determine the "fair" amount of resources which should have been assigned to each job.

Studies [11, 12] show that fairness in queues is very important to humans. For instance, a long wait at a supermarket is made more tolerable if all customers are served in order. Further, it has been observed that customers get upset if they are treated unfairly, even if the absolute wait time is short. Research by Larson [12] has shown that fast food restaurant customers prefer a longer, single queue rather than a multiple queue system with a shorter wait time. This can be attributed to a perceived increase in fairness due to the knowledge that the users are served in arrival order. Mann [11] studied queues for long over night waits at multiple events. He and his associates note the importance of preserving the FCFS queuing order. They analyze the impact of queue position on "queue jumping" and report that the response to "queue jumping" can range from "jeering and verbal threats" to physical altercations. While these queuing situations have significant differences from a non-preemptive space sharing parallel job scheduler, there also exist important similarities. For instance, it is postulated that the human responses to unfairness result from contention for "valued" resources (such as the event tickets in the previous example and the compute cycles in job scheduling) and the perception that a user who has waited longer somehow deserves the resources more.

Thus it is important that fairness be considered in addition to common user metrics (such as turn around time and wait time). By characterizing a center's scheduling policy through suitable fairness metrics, system administrators could assure their users that they are being treated fairly. In this paper, we present two rather distinctly different approaches to quantifying fairness in job scheduling, and assess several standard job scheduling strategies by use of these fairness metrics.

The remainder of this paper is organized as follows: Sect. 2 provides a motivation for fairness by introducing past fairness work in other disciplines. Section 3 introduces the proposed fairness metrics for parallel job schedulers. Sections 4 and 5 provide a simulation study showing the relative fairness of prevalent scheduling schemes. Section 6 provides possible directions for future work, and Sect. 7 concludes the paper.

## 2 Motivation and Related Work

Networking, sociology and operational research provide two broad categories of fairness. The first is based on social justice and is centered on the user's perception of fairness in terms of expected order of service (arrival order). This type of fairness model is common in sociology and operations research studies. The second fairness category is based on equally dividing the resources (servers) amongst all active users. This concept is based on the fundamental belief that all users equally deserve the resources whenever they are present. Resource equality based metrics are prevalent in both computer networking and queuing systems.

Social justice is the foundation for the "slips and skips" view of fairness [12, 13, 14] in queuing systems. They introduce "slips and skips" as a means to measure "social justice" in queuing systems. A slip occurs when the queuing position is worsened due to being overtaken by a later arriving entity. A skip is the result of an improved queue position due to the act of overtaking an earlier arriving entity. Rafaeli et. al. [15] performed studies which concluded that perceived fairness in queues is essential and can actually be more important than wait time. Therefore, it seems that it would be helpful to show users how fair a system is and more specifically how fairly an individual's jobs were treated.

The RAQFM [8, 9, 10] is an example of the second class of fairness categories. It calculates fairness in queuing systems by taking the difference between the "deserved" service attributed to a job and subtracting the actual service provided to the job. The deserved service is computed by equally dividing all resources amongst all active users in the system at each time quantum. The deserved resources are defined as $\delta_i = \int_{a_i}^{d_i} 1/N(t)dt$, where $N(t)$ is the number of jobs in the system, $a_i$ is the arrival time and $d_i$ is the departure or completion time. The discrimination is defined as $D_i = \delta_i - s_i$, where $s_i$ is the service time of job $i$. Three important observations made by Raz et. al. [10] are: 1) if service times are equal then FCFS is the most fair non-preemptive policy, 2) if arrivals times are the same then SJF is the most fair queuing priority, and, 3) a processor sharing scheme is the only ideally fair queuing policy.

There is also queuing fairness work in the context of streaming networks [16, 17, 18]; much of this work is based on max-min fairness, which is similar in spirit to RAQFM. This approach attempts to "evenly" divide bandwidth amongst active flows or packets such that a job can not receive more resources if it would result in reducing the resources of the least serviced job.

RAQFM is a metric which incorporates job seniority and a job's service time. In contrast, other queuing fairness [19, 20, 21] metrics based on a job's slowdown do not explicitly take both job seniority and the job's service time into account. Weirman and Balter [19] use the max mean fairness of all jobs, and consider a policy fair if all job categories have a mean slowdown of no more than $1/(1 - \rho)$, where $\rho$ is the load. They show that using this method no non-preemptive schedule is fair, processor sharing is fair, and SJF is more fair the FCFS. Schwiegelshohm and Yahyapour[22] introduce a parallel job scheduling algorithm that can bound a job's slowdown.

Sabin et. al.[23] measure the fairness of individual sites in a multi-site job scheduling environment. The focus of the multi-site fairness work is to ensure that lightly loaded local sites are not too adversely affected by joining a job sharing consortium. That work does not address the relative ordering (and fairness) of individual jobs, as we do in this paper. The metric suggested in [23] compares the performance of each job with job sharing to the performance of each job without job sharing. The authors conclude that the lack of focus on fairness may have lead to unfair decisions in current meta-schedulers, and suggest techniques to improve fairness.

## 3 Unfairness Metrics

We introduce two fairness metrics. The first metric is related to “slips” and “skips” discussed earlier, and the second metric is based on an equal distribution of system resources amongst all active jobs in the system, similar to RAQFM [8, 9, 10] and network flow scheduling. The metrics are very different in nature and quantify significantly different notions of fairness in the schedule. However, both metrics appear very appropriate to consider when performing a fairness analysis for a schedule. We compare different scheduling schemes against both fairness metrics, to identify commonalities and differences.

These metrics are intended to be orthogonal to the traditional performance metrics. There may be cases where a more fair scheme also exhibits better user metrics such as turnaround time. However, there may be scenarios where a more fair scheme has worse performance metrics. For instance, a non-backfilling FCFS policy will be entirely socially just, but has a very poor average turnaround time. To fully evaluate a scheduling strategy, the fairness should be evaluated in addition to more traditional user and system metrics. There is a large body of research which evaluates the other user and system metrics for the scheduling policies studied in this paper; therefore this paper focuses on fairness and not on the variations in the other metrics. This balance is not unlike balancing wait time and fairness in physical queues as done in [12] and more formally in [15]. Rafaeli’s studies [15] showed that anger due to queuing is more closely related to the perception of unfairness rather than the absolute wait time.

### 3.1 Fair Start Time Analysis

Fair start time (FST) [24] based metrics are proposed as a means to measure social justice for parallel job schedulers. In order to motivate our approach, we use an analogy. Consider a deli line, having many workers, each performing only one task. Would it be fair for a person who arrived later to have access to the cashier earlier than the first person in the line? In general, this would be considered unfair and socially unjust. But if the later person is only ordering a soda, and arrives at the cashier in time to have the sale completed before the earlier person’s sandwich order is ready, there isn’t really any social injustice. The later person obtained a resource “out of order”, but did not delay the earlier arriving person. This is analogous to a later arriving job that backfills but has no negative effect on the start time of earlier arriving jobs. We will refer to this situation as a “benign backfill”. Therefore, this metric is similar to measuring the “slips”, as we are trying to measure the effect of jobs being run out of the preferred order (FCFS). However, as demonstrated in the analogy, there are significant differences which affect how we define this metric.

Parallel job schedulers must inherently deal with multi-resource jobs and multiple “servers” (processors). In this context, jobs use multiple resources

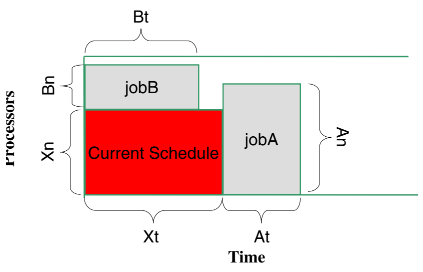


**Fig. 1.** An example of a benign backfill due to multiple resources where the apparent skip by jobB does not affect jobA

(processors); this means that all "skips" are not necessarily harmful. It is possible for a job to "skip" ahead and not affect any other job in the schedule. In a single resource queuing system, any job receiving service "out of order" inherently delays all other earlier arriving jobs. In a multi-resource job scenario, this is not always the case. It is possible that a "skip" is a benign backfill. For instance, assume that we have $Sn$ total resources (nodes) and $Sn - Xn$ free nodes for $Xt$ seconds, $jobA$ needs $An$ nodes for $At$ time, and $jobB$ needs $Bn$ nodes for $Bt$ time. Assume $Bt < Xt$ and $Bn < (Sn - Xn) < An$ and $jobA$ arrived before $jobB$. The start time of $jobA$ is not affected by $jobB$ running before $jobA$, even though $jobB$ arrived later (see Fig. 1). Therefore, the early start of $jobB$ represents a benign backfill and is not unfair.

Informally, how fairly a job is treated depends on whether the job could have run at an earlier time had no later arriving job been serviced. If it is delayed because of a later arriving job, it is considered to be treated unfairly. This is analogous to the deli line, and answers the question "did any later arriving job affect my start time". Based on the previous discussion, benign backfilling is obviously possible, and is in fact the desired result of backfilling. Therefore, simply counting how many jobs leapfrogged ahead of a particular job is not an adequate metric to characterize unfairness. Further, it is difficult to identify the benign backfills, as a backfill may only adversely affect some of the active jobs, due to inexact users estimates and the dynamic nature of online job scheduling.

**Strict Fair Start Time.** Since we can not directly measure the number of "skips" and/or "slips", we decide to measure the effect of later arriving jobs on the start time of each job. To measure this effect we run a "what if" simulation after all jobs have completed. Therefore the "what if" simulation has all the necessary data (actual runtimes, resource constraints, etc.) to determine exactly what would have happened had no later arriving jobs been in the system. This allows us to determine if, and by how much, the job was affected by later arriving jobs. This simulation takes into account the state of the schedule (running jobs and idle resources) when the job arrived. We are able to start the simulation from this state and determine when the job would have started if no later arriving jobs were ever in the system. We call the start time in this "what if" schedule the Fair Start Time (FST) for this job. If the actual start time of the job is not

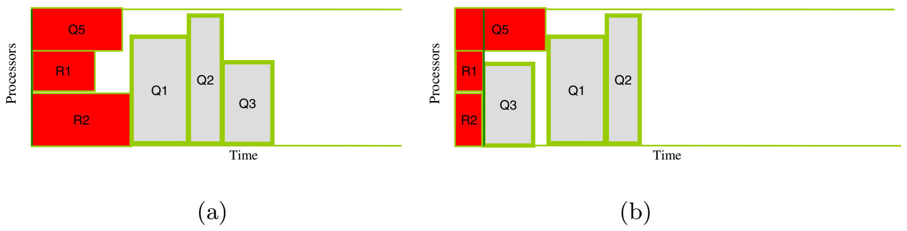


**Fig. 2.** An example of a job starting before it's fair start time. Job Q5 backfills in the top figure. In the bottom figure, job R1 and R2 complete early, allowing Q3 to start earlier than it would have if Q5 were not allowed to backfill.

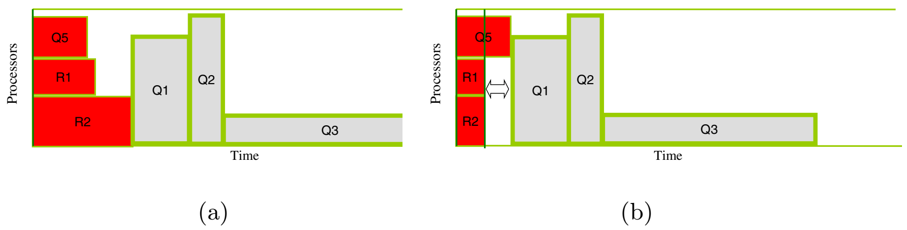


**Fig. 3.** A latter arriving job (Q5) delaying earlier arriving jobs (Q1). The delay is caused by the inaccurate estimates of R1 and R2. The top figure shows job Q5 backfilling into the scheduling window and the bottom figure shows the schedule after R1 and R2 complete early.

greater than this FST, the job was not delayed by any later arriving jobs, and any backfills (out of order executions) must have been benign with respect to this job.

It is possible that due to the dynamics of later arriving jobs, that the job actually had an earlier start time in the actual trace. This is because later arriving jobs could have created "holes" in the schedule for this job to backfill into (see Fig. 2). Therefore, if the job's FST is less than is actual start time is has been treated unfairly (see Fig. 3), however if the jobs FST is greater than the actual start time, the job has been given preferential treatment.

**Relaxed Fair Start Time.** We next define a less strict notion of the FST, motivated by a couple of issues pertaining to the strict FST metric defined above:

- In the process of generating a job's strict FST, it may have skipped ahead via backfilling. If in actual execution, the job was not able to leapfrog ahead of earlier jobs via backfilling, but was not otherwise delayed, should it really be considered to be unfairly treated?
- If the job backfills when computing its strict FST, and again during actual execution, this backfilling might force an earlier arrived job to violate its strict FST. Thus it may be the case that the collection of strict FST's for the jobs of a trace implies an inherently infeasible schedule.

Therefore, we define a “relaxed” FST metric, very similar to the strict FST in most ways. The only difference is that the job we are determining the FST for is not allowed to backfill (it can not start until all other jobs submitted before it have started). This gives an FST that is strictly greater than or equal to the Strict FST, thus we call this the Relaxed FST. It should be noted that while the job in question will not directly force another job to be treated unfairly, the set of relaxed FST’s is not necessarily a feasible schedule.

The pseudo-code below shows a possible implementation to compute the strict and relaxed fair start time. This would need to be added to the scheduler. The code assumes that a snapshot of the schedule (with information sufficient to start a simulation) is taken upon job arrival. This snapshot is retrieved with the getScheduleSnapshotAtArrival() method. As can be seen, the only difference between the relaxed and strict metric is when the newly arriving job is added to the “what if” simulation. It should be noted that this must be performed after all jobs which arrived before $j$ have completed, so that the scheduler can perform an accurate simulation using the actual runtime of the jobs.

A consequence of both of the proposed approaches (Strict and Relaxed) to characterizing fairness is that the FST of a particular job in a trace will generally be different under different scheduling strategies. Consider a job X that arrives at t=60 minutes. It is possible for it to have a fair-start time of 100 minutes and an actual start time of 120 minutes with scheduling strategy A, and a fair-start time of 200 minutes and an actual start time of 180 minutes with scheduling scheme B. Under our model of fairness, the job is unfairly treated by scheduling scheme A, but is fairly treated by scheduling scheme B, although the actual start time for scheme B is much worse. Is this reasonable? We believe that the conclusion regarding fairness is not inappropriate, since fairness and performance are orthogonal considerations. Although the user might prefer scheduling scheme A because it provides a better response time, it does not mean that it is a fairer scheme. Moreover, there has been sociological research that suggests users might actually prefer the fair scheme, similar to the longer wait times in the fast food restaurant [12].

Is it possible to use a common reference fair-start time for each job when comparing the fairness of different scheduling schemes? We believe not, since the relative performance of the base scheme would confound the evaluation. A strict FCFS policy (i.e. with no backfilling) is a completely fair scheduling scheme, in terms of social justice. So this might seem to be a logical choice for a reference scheme. However, FCFS no-backfill generates an inherently poor schedule with very high average response time due to low utilization. Therefore, backfilling scheduling schemes would likely result in most jobs having better response times than that with FCFS no-backfill. This would be true even with schedules that are blatantly unfair. For example, consider an FCFS-aggressive schedule on a trace with many pairs of identically shaped jobs that arrive in close proximity. With most implementations of aggressive backfilling, these pairs of identical, close proximity jobs would start in arrival order, i.e. be treated fairly. Now consider a modified schedule where we swap the positions of the job pairs

```
  # Paramaters:
  # j - the job we are computing the FST for
  # isRelaxed - should the relaxed or strict metric be computed
  #
  # Return:
  # The FST for job j

public long calcFST(Job j, boolean isRelaxed){
  #Get the state of the scheduler when job j arrived
  Schedule schedule = getScheduleSnapshotAtArrival(j)

  # If we are calucating the strict FST, insert job j into
  # the scheduler so that it is allowed to backfill.  If we are
  # calulating the relaxed FST, we need to allow all jobs in
  # the schedule to backfill and start before job j is added
  if !isRelaxed
    schedule.addJob(j)

  # Run the simulation to completetion
  schedule.simulateUntilAllJobsHaveStarted()

  if(not isRelaxed)
    # Simply return when the job started
    return j.getStartTime()
  else #isRelaxed
      # All jobs which arrived before j have already been forced
      # to start, add j to the schedule to see when it will start
    schedule.add(j);
    # Run the simulation to completetion
    schedule.simulateUntilAllJobsHaveStarted()
    return j.getStartTime()
}
```

**Fig. 4.** This method calculates the fair start time (relaxed or strict) for a given job

in the aggressive backfill schedule - i.e. have the later job run in the earlier job's slot and vice versa. This modified schedule is unquestionably unfair with respect to these job pairs. But in the corresponding FCFS-no-backfill schedule, it is very likely that all jobs would have later start times than the FCFS-aggressive schedule. So if we just compare a job's start-time with its reference start-time under the FCFS-no-backfill schedule, all jobs would be considered to be fairly treated for this contrived schedule that is obviously unfair.

Thus it is problematic to use a single reference schedule in comparing fairness of different scheduling schemes. The metric for fairness should be independent of the performance (from the classical user/system metrics) of the scheduling policies. The goal of characterizing fairness is not to determine whether one policy is more effective than another with respect to user/system performance metrics.

Fairness and performance are both important in determining the attractiveness of a scheduling strategy, but they are independent factors.

**Cumulative Metric.** Above we have defined ways to measure the FST for each job. We can then determine the unfairness for each job by subtracting the FST from the actual start time. If the $ActualStartTime - FST$ is less than zero, the job has been given preferential treatment; if it equals zero the job has been treated fairly; and if it is greater than zero it has been treated unfairly.

We define the overall average unfairness as the sum of the unfairness divided by the number of jobs.

$$OverallUnfairness = \frac{\sum_{i \in jobs} max(ActualStartTime_i - FST_i, 0)}{\sum_{j \in jobs} 1} \ . \quad (1)$$

Therefore, jobs which are given preferential treatment can not bring down the metric, as we only sum over unfairly treated jobs. Also, we divide the sum by the total number of jobs. So if only one job is treated unfairly by X, the overall unfairness is X/N, while a scheme where all jobs are treated unfairly by X will have an overall unfairness of X.

## 3.2 Resource Equality

The FST metrics explained above measure effects similar to "skips" and "slips" in queues. A very different model for fairness is that over any period of time, ideally an equal fractional time slice should be provided for each job in the system. The idea behind the Resource Equality (RE) based metric is to evenly divide the resources amongst all jobs which are active in the system. Active jobs are defined as jobs which have arrived but have not exited the system (completed), i.e. running and queued jobs. In a single resource context (i.e. all sequential jobs), each job simply deserves $1/N$ (where $N$ is the number of active jobs in the system) of the resources for each time quantum (the length of which is defined by job arrival and departures). However, in a multi-resource context (i.e. with parallel jobs), some jobs are able to use more resources in a time quantum than other jobs. For instance, if there are a 5-processor and a 15-processor job active in a 20 processor system, it does not make sense to insist that each job deserves 1/2 of the system for the time quantum. The 5 processor job could not use more than 1/4th of the system, and is not being treated unfairly if it only receives 1/4th of the system.

Conceptually, the resource equality based metric can be viewed as an ideal virtual time slicing scheme in a round robin fashion (processor sharing). Here each job receives an infinitesimally narrow time slice, and a width that equals the number of actual requested nodes. It is an idealized time slicing scheme, because we ignore packing issues by allowing jobs to fill fragmented slices and complete in the next time slice. We add the condition that no job can deserve more cycles than it is able to consume (a 8 process job, in a $t$ length time quantum, deserves a maximum of $8t$ cycles). Further, we follow the model that only the total used resources should be divided amongst active jobs, rather than the total system resources, as proposed in RAQFM in a multi-server context. Therefore, it is feasible for a scheme to be fair even if resources are wasted and the resource equality unfairness metric is independent of utilization and loss of capacity.

The deserved time for job i is more formally defined as below:

$$d_i = \int_{a_i}^{c_i} min(\frac{nodes_i}{\sum_{j \in ActiveJobs} nodes_j} \times used\_Nodes, nodes_i)dt \ . \tag{2}$$

The unfairness for each individual job is defined as the deserved resources ($d_i$) minus the consumed resources ($s_i = runtime_i \times processors_i$):

$$D_i = d_i - s_i \ . \tag{3}$$

We then define the overall unfairness in a manner similar to the FST based metrics:

$$OverallUnfairness = \frac{\sum_{i \in jobs}(max(D_i, 0))}{\sum_{j \in jobs} 1} \ . \tag{4}$$

This metric is partially independent of the actual scheduling policy used. The metric is based solely on when a job arrives and departs and how many other jobs are active in the system. Therefore, this metric can easily be computed using post processing, by using the scheduler output trace in the Standard Workload Format [25]. This is different from the FST based algorithm that explicitly uses the current scheduling policy and needs the state of the schedule in order to generate the FST when each jobs arrives.

## 4 Simulation Environment

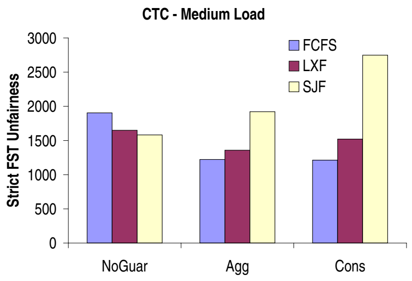
We used the proposed fairness metrics to analyze several standard space sharing, parallel job scheduling schemes. We analyzed three depths of reservations: no guarantees (no reservations), EASY/aggressive backfilling (1 reservation), and conservative backfilling (all jobs have a reservation). For each of the reservation depths, we simulated three queuing orders, First Come First Served (FCFS), Shortest Job First (SJF), and Largest eXpansion Factor (LXF)[26]. For each queuing order, the queue was dynamically sorted by the specified criterion.

We performed simulations on 5000 job subsets from each of the following traces: 128 node SDSC SP2, the 512 node CTC SP2 (with 430 batch nodes), the 1152 SDSC Blue, and the 178 node OSC machines available from the Feitelson workload archives [25]. We modified the traces to simulate offered loads of 70%, 80%, 90%, 95%, and 98%. We show representative data using the 80% (medium load) and 95% (high load) simulations using the CTC SP2 and SDSC Blue input traces.

To modify the workload of an input trace, we multiplied both the user supplied runtime estimate and the actual runtime by a suitable factor to achieve a desired offered load. For instance, let us assume that the original trace had a utilization of 65%. To achieve an offered utilization of 90%, the actual runtime and estimated wallclock limit was multiplied by 0.9/0.65. Thus we used runtime expansion to increase the mean load. Using runtime expansion in lieu of shrinking the inter-arrival keeps the duration of a trace consistent. When shrinking the inter-arrival times, the duration of the input trace is proportional to the change in the offered load. When using runtime expansion, the duration of a long simulation remains roughly same. We note that the schedules generated with either approach is equivalent. Let $i$ be the basic time unit when shrinking inter-arrival time, $r$ be the basic time unit when using runtime expansion, and $x$ be the factor by which the run times were expanded and the inter-arrival times were shrunk. Simply letting $x * i == r$ gives identical simulation states.

## 5 Evaluation

Figures 5 and 6 show the overall average unfairness using the strict FST based metric. These figures show that no-guarantee backfilling is worse than aggressive or conservative backfilling, especially for the SDSC Blue trace. There are cases where SJF no-guarantee is better than SJF conservative and SJF aggressive backfilling; however SJF is generally more unfair than FCFS and LXF. There are no consistent trends showing LXF or FCFS to be more unfair than the other. Figures 7 and 8 show the strict FST based unfairness metric categorized by job width (number of nodes). Only figures for FCFS data are shown, as the relative results for comparing no guarantee, aggressive and conservative backfilling remain similar for LXF or SJF queuing priority. Table 1 shows the number of jobs in each category. These figures show that wide jobs (i.e. jobs which need many nodes) are treated extremely unfairly using a no-guarantee backfilling scheme. Therefore, even though the overall unfairness for no-guarantee backfilling may look acceptable using the strict FST metric for the CTC trace, wide jobs are treated very unfairly. Therefore, we can conclude that the strict FST metric shows that SJF is an unfair queuing order and no-guarantee backfilling is unfair. However, no consistent conclusions can be drawn regarding the relative unfairness of an LXF and an FCFS queuing priority, or between aggressive and conservative backfilling.



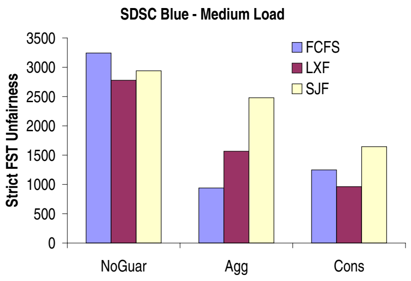


**Fig. 5.** Average strict fair start miss time under medium load

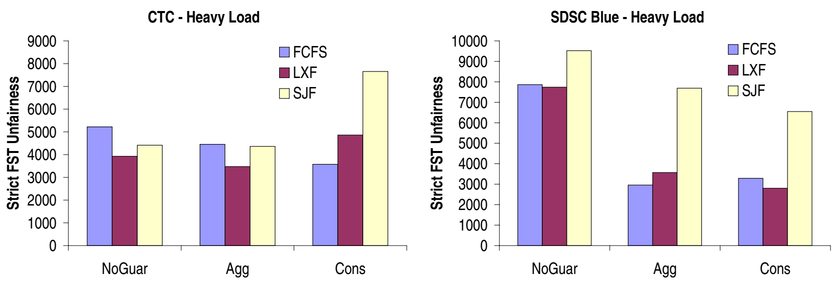


**Fig. 6.** Average strict fair start miss time under high load

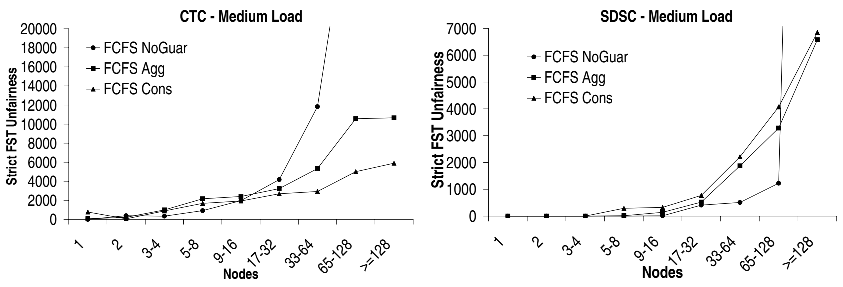


**Fig. 7.** Average strict fair start miss time for different width jobs under medium load

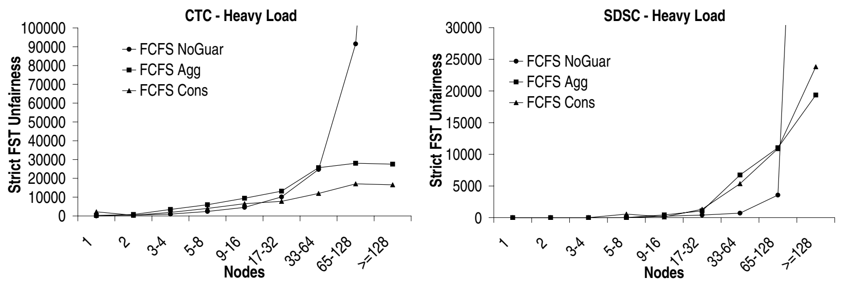


**Fig. 8.** Average strict fair start miss time for different width jobs under high load

**Table 1.** The number of jobs for width (processor) categories presented

| | 1 | 2 | 3-4 | 5-8 | 9-16 | 17-32 | 33-64 | 65-128 | > 128 |
|---|---|---|---|---|---|---|---|---|---|
| **CTC SP2** | 2209 | 435 | 722 | 487 | 609 | 309 | 157 | 45 | 30 |
| **SDSC Blue** | 3 | 0 | 0 | 2489 | 653 | 569 | 616 | 396 | 277 |

Figures 9 and 10 show the overall average unfairness using the relaxed FST based metric. The relaxed metric shows that no-guarantee backfilling or a SJF queuing priority are unfair, similar to the conclusions with the strict metric. However, for all simulations, the relaxed metric more clearly shows no-guarantee backfilling to be unfair than does the strict metric. Also, the relaxed metric

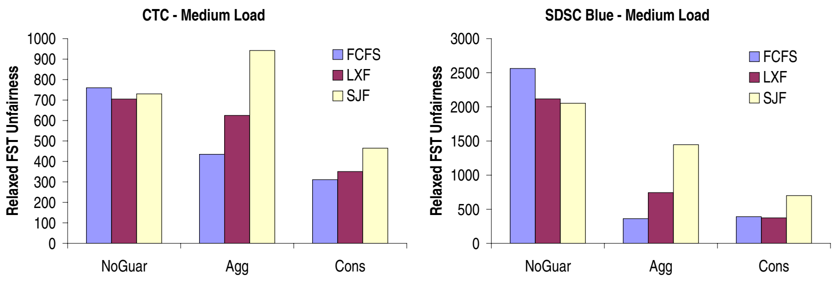


**Fig. 9.** Average relaxed fair start miss time under medium load

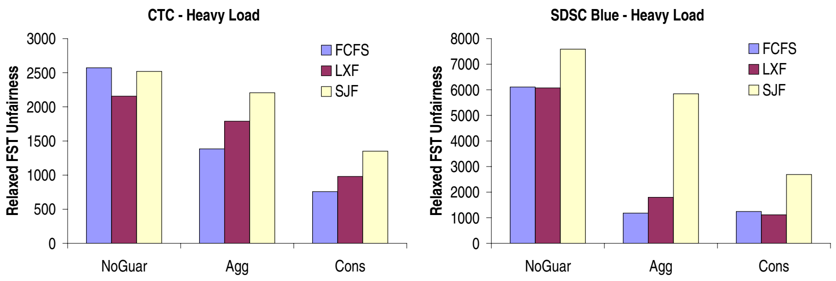


**Fig. 10.** Average relaxed fair start miss time under high load

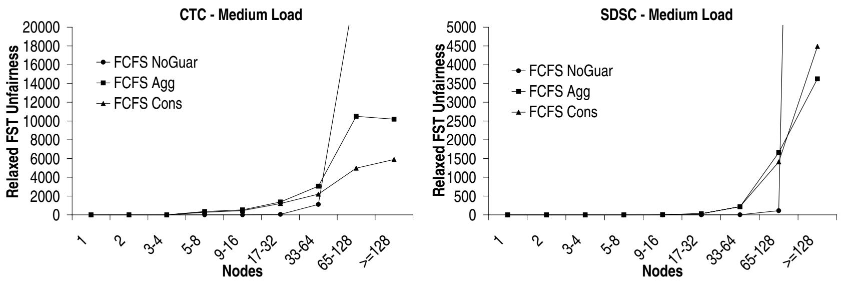


**Fig. 11.** Average relaxed fair start miss time for different width jobs under medium load

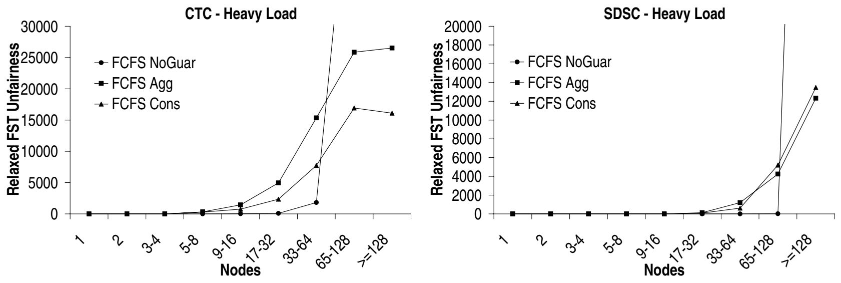


**Fig. 12.** Average relaxed fair start miss time for different width jobs under high load

shows a clearer separation between an LXF and FCFS queuing priority, and between an aggressive and conservative backfilling policy. Using this metric, FCFS is either less unfair or extremely close to LXF and conservative backfilling is less unfair than aggressive backfilling; however they are both close in some situations. Figures 11 and 12 show the relaxed FST based unfairness metric for width categories. Again only FCFS data is shown, as the trends for different depths of reservation are consistent for LXF and SJF. These figures further show the unfairness of no-guarantee backfilling, as the wide jobs are treated extremely unfairly. The conclusions that can be drawn from this metric are that an SJF queuing policy or no-guarantee backfilling are unfair scheduling policies. Further, LXF appears to be slightly more unfair than FCFS and aggressive backfilling appears to be slightly more unfair than conservative backfilling.

The relaxed and strict metric show similar results; however the relaxed metric gives a clearer picture with respect to the differences between LXF and FCFS, and between conservative and aggressive backfilling.

Figures 13 and 14 show data for the resource equality based metric, for the CTC and SDSC traces, for medium and high loads. This metric shows that a no-guarantee backfilling scheduling policy is very unfair with respect to the RE based metric. However, the RE based metric does not show any significant distinction between aggressive and conservative backfilling. Further, this metric does not show significant differences between different queuing priority orders. Thus, similar to the FST based metrics, there is little difference between ag-

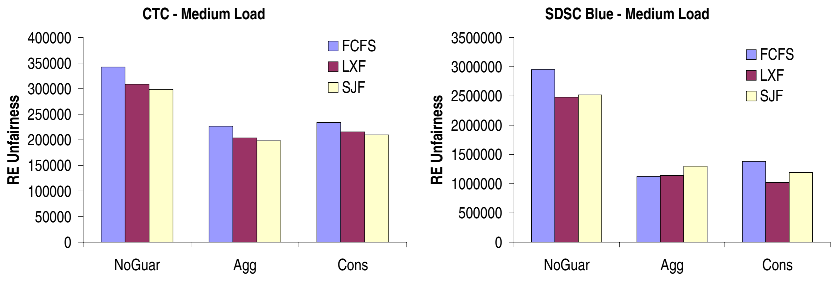


**Fig. 13.** Resource equality based unfairness under medium load

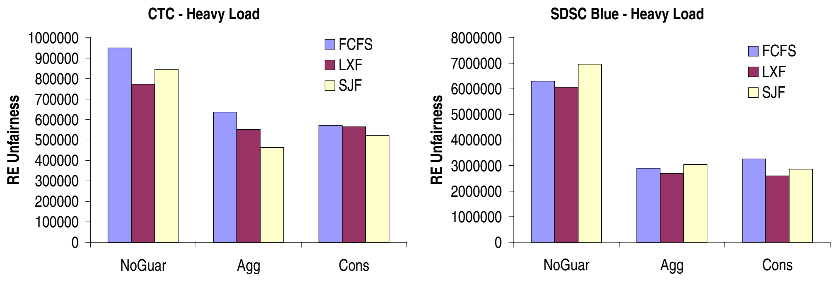


**Fig. 14.** Resource equality based unfairness under high load

gressive and conservative backfilling and no-guarantee backfilling appears to be unfair. However, SJF does not appear to be an unfair scheme using the resources equality based metric, which is very different from the conclusions drawn from the FST based fairness metrics.

Figures 15 and 16 show the overall resource equality based metric for the CTC and SDSC traces at the medium and high loads for varying width jobs - again only FCFS figures are shown as LXF and SJF shows similar trends. Again, it can be seen that no-guarantee backfilling treats wide job very unfairly.

All three metrics show that no-guarantee backfilling is an unfair scheduling policy, especially for wide jobs. This is not a surprising result, as no-guarantee backfilling does not provide a mechanism for wide jobs to start in a timely manner in the presence of narrower jobs. There is no consistent difference between aggressive and conservative backfilling when using two of the metrics; however the relaxed FST metric shows that conservative backfilling may be slightly less unfair. An SJF queuing order is much more unfair in the FST based metrics, while it is comparable to FCFS and LXF with the RE based metric. There is also little difference between LXF and FCFS: two of the three fairness metrics do not show any consistent trends, while the third metric (relaxed FST) shows that FCFS may be slightly less unfair than LXF. The apparent closeness of LXF and FCFS is a surprising result, as FCFS attempts to run jobs in arrival order; it appears that the inaccurate user estimates leads to a significant amount of unfairness even with an FCFS priority queue. This is due to the scheduler allowing a job to backfill. The apparently

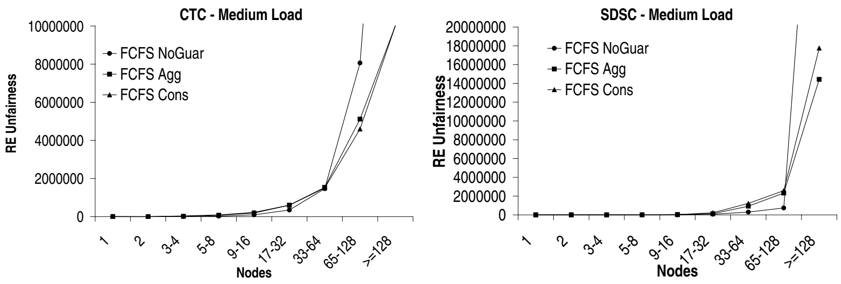


**Fig. 15.** Resource equality based unfairness for different width jobs under medium load

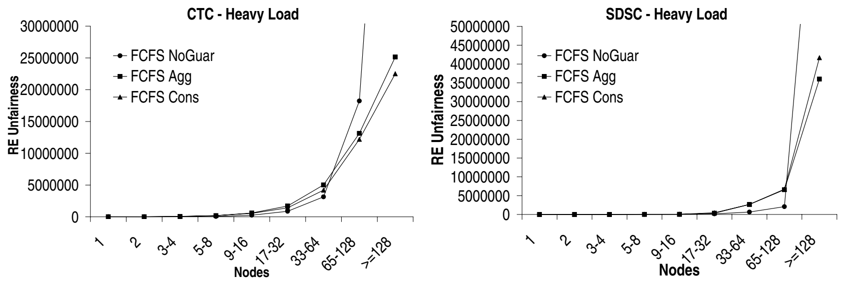


**Fig. 16.** Resource equality based unfairness for different width jobs under high load

benign backfill can delay earlier jobs when other running jobs have not accurately estimated their runtimes [24]. The unfairness in a SJF schedule is possibly caused by temporarily starving some of the longer jobs while giving preferential treatment to shorter jobs. In conclusion, SJF and no-guarantee backfilling generally exhibit greater unfairness, while there is little difference between FCFS vs. LXF and between conservative vs. aggressive backfilling.

### 5.1 Rank Correlation

Here we assess how sensitive these metrics are to the choice of jobs in a trace. Each trace was divided into 10 "random" subsets, based solely on the arrival order of the jobs. Table 2 shows the Spearman [27] correlation between each subset and the overall results, using all 45 simulations for each trace (combinations of load, backfilling policy, and queue priority policy). The rank correlation was performed between each of the subsets and the overall average. The data points being ranked are the 45 simulations on a trace by trace basis. The 45 simulations were obtained from 5 offered loads, the three queuing priorities, and the three backfilling policies. The simulations were ranked (individually) between 1 and 45 for each of the 10 subsets and for the overall average. The Spearman rank order correlation was than computed between each $subset_i$ and the overall average, to show the correlation between job subsets and the overall average.

The high correlation indicates that the order of the metrics would be similar even if we chose to only rank schemes by the unfairness of a subset of the jobs. Therefore, unfairness is similar for different subsets and does not vary significantly.

**Table 2.** Spearman correlation data between subsets of data and the overall unfairness

| | CTC | | | SDSC | | |
|---|---|---|---|---|---|---|
| | **FST Strict** | **FST Relaxed** | **Resource Equality** | **FST Strict** | **FST Relaxed** | **Resource Equality** |
| **Subset 0** | 0.96 | 0.96 | 0.96 | 0.95 | 0.91 | 0.93 |
| **Subset 1** | 0.97 | 0.95 | 0.97 | 0.98 | 0.96 | 0.98 |
| **Subset 2** | 0.93 | 0.87 | 0.89 | 0.97 | 0.96 | 0.96 |
| **Subset 3** | 0.93 | 0.76 | 0.88 | 0.97 | 0.98 | 0.98 |
| **Subset 4** | 0.95 | 0.85 | 0.88 | 0.97 | 0.97 | 0.97 |
| **Subset 5** | 0.94 | 0.83 | 0.91 | 0.96 | 0.95 | 0.98 |
| **Subset 6** | 0.97 | 0.96 | 0.98 | 0.97 | 0.96 | 0.95 |
| **Subset 7** | 0.96 | 0.97 | 0.97 | 0.99 | 0.98 | 0.97 |
| **Subset 8** | 0.98 | 0.91 | 0.95 | 0.94 | 0.88 | 0.93 |
| **Subset 9** | 0.94 | 0.95 | 0.96 | 0.85 | 0.74 | 0.76 |

## 6 Future Work

We hope to extend the FST based unfairness work to be based on other fairness priorities (other than FCFS). The current metric assumes that the fair order is

FCFS. The queue priority can be changed, but the FST based schemes run "what if" simulations assuming that no later arriving jobs exist. We would like to be able to use other fairness priorities, so that system administrators can select a fair order (possibly based on time in the queue or administrative priorities) and measure the unfairness based on these altered fairness priority assumptions. One issue that is difficult when addressing this issue is handling dynamic priorities and arriving jobs.

We would also like to expand these metrics and ideas to other related areas beyond "simple" non-preemptive space shared schedulers. For instance, we plan on measuring unfairness and identifying other fairness concerns for parameter sweep applications and moldable job schedulers.

It would also be interesting to perform a comparison of slowdown based metrics to the two metrics defined in this paper. The slowdown based metrics do not entirely fit into either of the two presented categories (social justice or resource equality).

This work has focused on the fairness of individual jobs. The metrics are based on related fields where each job is treated individually and there is no distinction between users and jobs. However, in parallel job scheduling, there often are heavy users who may submit many jobs. While job level fairness is reasonable, and provides a firm foundation in other fields, a user based fairness metric would also seem reasonable. We believe that our work on job fairness can serve as a good foundation for continued work on user based fairness metrics.

## 7 Conclusion

In this paper, we introduced two fairness metrics for job scheduling, each motivated by past fairness research in sociology, computer networking, and/or queuing systems. The first metric (Fair Start Time based) is based on a deli-line notion of fairness or social justice, where jobs should be serviced in order. We attempt to measure the effects of out-of-order execution caused by backfilling and inaccurate user estimates. This is similar to measuring "skips" and "slips". The second metric is based on equally sharing the resources amongst active jobs. This metric compares the resources a job deserves during its lifetime to the resources it actually consumes. It exhibits similarities to the RAQFM metrics used in queuing systems and fairness amongst network flows.

These two types of metrics are based on different principles and measure different features. Both metrics are orthogonal to traditional user metrics (turnaround time or slowdown) or system metrics (utilization). The extent of similarities in the observations using the two metrics is somewhat surprising. Both metrics indicate that LXF and FCFS have comparable unfairness, while results for an SJF queue depend on the unfairness metric being used. Further, there is not a consistent difference in unfairness between aggressive and conservative backfilling, while no-guarantee backfilling is consistently more unfair (especially in regards to the unfairness towards wide jobs).

Thus we have introduced unfairness metrics for parallel job scheduling, and were able to draw some consistent results across multiple traces and different loads. The data shows that there may not be significant unfairness concerns for some previously reported schemes (LXF) which are often not implemented due to unfounded unfairness concerns. We hope these unfairness metrics are able to improve the acceptance of other scheduling schemes, where unfairness may be a concern. Further, these metrics can be used to help alleviate user concerns about unfairness of existing scheduling schemes, hopefully increasing user satisfaction.

**Acknowledgments.** We would like to thank Dror Feitelson for the workload archive, Travis Earheart and Nancy Wilkins-Diehr for the SDSC Blue workload logs, Victor Hazlewood from HP Labs for the SDSC SP2 logs, and Dan Dwyer and Steve Hotovy for the CTC workload logs. We thank the referees for their suggestions for improvement.

## References

1. Talby, D., Feitelson, D.: Supporting priorities and improving utilization of the IBM SP scheduler using slack-based backfilling. In: Proceedings of the 13th International Parallel Processing Symposium. (1999)
2. Mu'alem, A., Feitelson, D.: Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. In: IEEE Transactions on Parallel and Distributed Systems. Volume 12. (2001) 529–543
3. Sabin, G., Kettimuthu, R., Rajan, A., Sadayappan, P.: Scheduling of parallel jobs in a heterogeneous multi-site environement. In Feitelson, D., Rudolph, L., Schwiegelshohn, U., eds.: Job Scheduling Strategies for Parallel Processing, 9th International Workshop. Volume 2862., Seattle, WA, USA, Springer-Verlag Heidelberg (2003) 87 – 104
4. Islam, M., Balaji, P., Sadayappan, P., Panda, D.K.: QoPS: A QoS based scheme for parallel job scheduling. In Feitelson, D., Rudolph, L., Schwiegelshohn, U., eds.: Job Scheduling Strategies for Parallel Processing, 9th International Workshop. Volume 2862., Seattle, Washington (2003)
5. Feitelson, D.: Workshops on job scheduling strategies for parallel processing. (www.cs.huji.ac.il/ feit/parsched/)
6. Shmueli, E., Feitelson, D.: Backfilling with lookahead to optimize the performance of parallel job scheduling. In Feitelson, D., Rudolph, L., Schwiegelshohn, U., eds.: Job Scheduling Strategies for Parallel Processing. Springer-Verlag (2003) 228–251 Lect. Notes Comput. Sci. vol. 2862.
7. Srinivasan, S., Kettimuthu, R., Subramani, V., Sadayappan, P.: Characterization of backfilling strategies for job scheduling. In: 2002 Intl. Workshops on Parallel Processing. (2002) held in conjunction with the 2002 Intl. Conf. on Parallel Processing, ICPP 2002.
8. Raz, D., Levy, H., Avi-Itzhak, B.: A resource-allocation queueing fairness measure. In: Proceedings of Sigmetrics 2004/Performance 2004 Joint Conference on Measurement and Modeling of Computer Systems, New York, NY (2004) 130–141 Also appears as *Performance Evaluation Review Special Issue* 32(1):130-141.
9. Avi-Itzhak, B., Levy, H., Raz, D.: Quantifying fairness in queueing systems: Principles and applications. Technical Report RRR-26-2004, RUTCOR, Rutgers University (2004)

10. Raz, D., , Levy, H., Avi-Itzhak, B.: RAQFM: a resource allocation queueing fairness measure. Technical Report RRR-32-2004, RUTCOR, Rutgers University (2004)
11. Mann, L.: Queue culture: The waiting line as a social system. The American Journal of Sociology **75** (1969) 340–354
12. Larson, R.: Perspectives on queues: Social justice and the psychology of queueing. Operations Research **35** (1987) 895–905
13. Gordon, E.S.: Slips and Skips in Queues. PhD thesis, Massachusetts Institute of Technology (1989)
14. Whitt, W.: The amount of overtaking in a network of queues. Networks **14** (1984) 411–426
15. Rafaeli, A., Kedmi, E., Vashdi, D., Barron, G.: Queues and fairness: A multiple study experimental investigation. (http://queues-fairness.rafaeli.net/)
16. Greenberg, A.G., Madras, N.: How fair is fair queueing? Association for Computing Machinery **39** (1992) 568–598
17. Demers, A., Keshav, S., Shenker, S.: Analysis and simulation of a fair queueing algorithm. Internetworking Research and Experience **1** (1990) 3–26
18. Nandagopal, T., Lu, S., Bharghavan, V.: A unified architecture for the design and evaluation of wireless fair queueing algorithms. Wireless Networks **8** (2002) 231–247
19. Wierman, A., Harchol-Balter, M.: Classifying scheduling policies with respect to unfairness in an M/GI/1. In: Proceedings of the 2003 ACM SIGMETRICS international conference on Measurement and modeling of computer systems. (2003) 238 – 249
20. Bansal, N., Harcol-Balter, M.: Analysis of SRPT scheduling: Investigating unfairness. In: SIGMETRICS. (2001)
21. Harchol-Balter, M., Sigman, K., Wierman, A.: Asymptotic convergence of scheduling policies with respect to slowdown. In: IFIP WG 7.3 International Symposium on Computer Modeling, Measurement and Evaluation. (2002)
22. Schwiegelshohn, U., Yahyapour, R.: Fairness in parallel job scheduling. Journal of Scheduling **5** (2000) 297–320
23. Sabin, G., Sahasrabudhe, V., Sadayappan, P.: On fairness in distributed job scheduling across multiple sites. In: Cluster. (2004)
24. Sabin, G., Kochhar, G., Sadayappan, P.: Job fairness in non-preemptive job scheduling. In: International Conference on Parallel Processesing. (2004)
25. Feitelson, D.G.: Logs of real parallel workloads from production systems. (URL: http://www.cs.huji.ac.il/labs/parallel/workload/)
26. Hansen, B.: An analysis of response ratio. In: IFIP Congress. (1971)
27. Weisstein, E.W.: Spearman rank correlation coefficient. http://mathworld.wolfram.com/SpearmanRankCorrelat ionCoefficient.html) From MathWorld–A Wolfram Web Resource.