CHARACTERISTICS OF FAIRNESS METRICS AND THEIR EFFECT ON PERCEIVED SCHEDULER EFFECTIVENESS

J. Ngubiri
Department of Computer Science
Makerere University
P.O.Box 7062
Kampala, Uganda
ngubiri@cit.mak.ac.ug

M. van Vliet
Informatics and Information Science
Radboud University Nijmegen
Toernooiveld 1, 6525 ED, Nijmegen,
The Netherlands
mario@cs.ru.nl

ABSTRACT
Parallel job schedulers are mostly evaluated using performance metrics. Deductions however can be misleading due to selective job starvation (unfairness). To choose a better scheduler, therefore, there is a need to compare schedulers for fairness as well. Performance and fairness, however, have mostly been studied independently. We examine characteristics of three approaches to fairness evaluation in parallel job scheduling. We examine how they represent job starvation and other aspects of discrimination. We show that the implied unfairness is not always starvation/discrimination in practice. We use simultaneous consideration of performance and fairness and compare deductions with scheduler effectiveness derived from group-wise performance evaluation. We observe that due to possible misrepresentation of starvation by fairness metrics, schedulers shown as superior may not be so in practice.

KEY WORDS
Performance, Fairness, Effectiveness, Co-allocation.

1 Introduction
Parallel job scheduling has been an active field of research for over a decade [1]. The satisfaction users get from the scheduler can vary from scheduler to scheduler. Broadly, we consider a scheduler that most satisfies the users to be the most effective. The choice of an effective scheduler is hard to make. One of the reasons could be that the (performance) metric used is inappropriate [2]. The appropriateness of the metric depends on the scheduling set up. If the metric is not appropriate, users’ satisfaction/dissatisfaction cannot be deduced from the metric values. Even in cases where performance metric used is appropriate, the results may be misleading due to selective starvation of jobs. Some jobs may be starved at the expense of others and the starvation is not adequately implied by the metric. An implied good/poor performance may therefore be unrealistic in practice. An effective scheduler needs be both fair and with good performance.

While a lot of research has been done on performance of parallel job schedulers [3], less (like [4][5][6][7][8]) has been done on fairness. Performance and fairness have also been studied independently yet the trend of one cannot be deduced from another.

Making a choice of a good scheduler needs simultaneous consideration of performance and fairness. This is because in some cases, good ‘performance’ is got by starving some jobs at the expense of others (unfairness). Measures of dispersion [6][4], Fair start-time analysis [5] and the Resource Allocation Queuing Measure (RAQFM) [17][18] are common fairness metrics. These measures can, sometimes, give misleading deductions for parallel job schedulers. They can indicate unfairness in cases where there is no starvation (and vice versa) which may lead to wrong deductions. We use typical parallel job scheduling cases and examine metric characteristics to identify instances where a scheduler starving no jobs can be deemed unfair (and vice versa). We then compare selected instances of multi-cluster schedulers by simultaneous consideration of performance and fairness. We validate the results by using group-wise performance evaluation. Previous studies [10][11][12][13] show that group-wise evaluation reveals job favoritism/starvation and hence scheduler effectiveness.

We observe that there are circumstances where measures of fairness do not represent all (un)fair scenarios accurately. They can, therefore, lead to counter-intuitive deductions. They are not reliable in evaluating scheduler effectiveness in parallel job scheduling.

We discuss the concepts and evaluation of performance and fairness in Section 2. We then discuss scheduler effectiveness in Section 3. We study fairness evaluation by dispersion, fair start-time analysis and RAQFM in Sections 4, 5 and 6 respectively. We describe the experimental studies in Section 7 and present the results in Section 8. We discuss our findings in Section 9 and make conclusions together with suggestions for future research in Section 10.

2 Performance and Fairness Evaluation

We now discuss the concepts and evaluation of scheduler performance and fairness.
2.1 Performance

Performance can be envisaged as a measure of user satisfaction. For example, a user may need to have the results to his request at the earliest possible time. Average Response Time (ART) can be used as a user based performance metric. Other performance metrics include Average Waiting Time (AWT), Job Slow-down (JSD) and throughput. An appropriate metric depends on the scheduling environment [2]. In a real time scenario for example, AWT is not an appropriate metric. The proportion of jobs that are processed before the deadline can be a better metric. The system set up also determines the interpretation that can be derived from the metric. While the deductions from AWT and ART give an in-depth view of performance. This can be due to differences in job schedulability [14]. Previous studies show that group-wise performance analysis is beneficial. This is because starved and favored jobs can easily be identified. Ngubiri and van Vliet [11][14] show that schedulability is highly dependant on job characteristics. Different job groups (grouped by size, number of components and width of widest component) have different performances. The rates at which these performances change with scheduler parameters are also different. Srinivasan et al. [15] use job groups to compare robustness of schedulers for moldable jobs. They show that different (size based) job groups have different (in some cases opposite) performance changes as scheduler parameters change. Some jobs therefore have a trend opposite that of the entire job stream average. More work by Srinivasan et al. [12][13] show that the performance difference between conservative and aggressive backfilling is actually in (size based) job groups. The dominating group dictates the overall job stream relative performance. Evaluation using job groups therefore help in identifying starvation and favoritism among the jobs. Groups however need to be chosen carefully. The job characteristics that determine the favoritism/discrimination should be used to generate the groups.

2.2 Fairness

The interpretational benefit of group-wise evaluation is due to the fact that it reveals relative performance. A good scheduler needs to be fair to all the jobs. It should not starve some of the jobs at the expense of others. Though less studied in parallel job scheduling, fairness is a very important aspect in all queuing systems. Possibly, it is because of fairness that queues came into existence. Rafaeli et al. [16] observed that people in queues are more annoyed by perceived unfairness than by the time spent in queues.

There two main perceptions of fairness. The first (called FCFS style) uses the users’ expected order of service while the second uses equal resource distribution among the users. The second approach is only applicable in time-slicing cases. Since we consider space-slicing, we focus of FCFS style only. The FCFS fairness has been evaluated mostly using the fair start-time approach, measures of dispersion and RAQFM.

In fair start-time approach [5], a job is considered unfairly treated if its processing is delayed by job(s) that arrived after it. The idea behind is that in a fair setting, a job should not be delayed by another that arrived after it. Let us consider a job \( J_i \) in a job stream. The actual start time \( t_i^a \) is the time \( J_i \) starts processing when the entire job stream is processed. The fair start time \( t_i^f \) is the time \( J_i \) starts processing if no job arrived after it (i.e. it is the last job in the queue). If \( t_i^a > t_i^f \), then \( J_i \) is unfairly treated. The mean of \( t_i^a - t_i^f \) for all unfairly treated jobs is used to determine scheduler fairness.

In dispersion approach [4][9], measures of dispersion like standard deviation and coefficient of variation are used to deduce fairness. The idea behind is that if a scheduler favors some jobs at the expense of others, then there is an increase in dispersion among the jobs’ performance. The higher the dispersion, the more unfair the scheduler is. Jain et al. [4] proposed that a good fairness metric should be population size and scale independent, bounded and continuous. They use the dispersion approach and proposed the fairness coefficient \( \kappa = \frac{1}{1+C_V} \) which satisfies the characteristics.

In RAQFM [17][18], the underlying principle is that all jobs are entitled to an equal share of the system resources. If at a time \( t \) there are \( N(t) \) jobs, then each job is entitled to \( \frac{1}{N(t)} \) of the system resources. If job \( J_i \) is given a proportion \( S_i(t) \), then the temporal discrimination \( D_i(t) \) is given by \( D_i(t) = S_i(t) - \frac{1}{N(t)} \). If \( J_i \) enters the queue at time \( t_i \) and completes processing at time \( t_i^c \), then the total discrimination is given by:

\[
D_i = \int_{t_i^a}^{t_i^c} S_i(t) - \frac{1}{N(t)} \, dt
\]  

(1)

The overall discrimination of the system is got by \( Var[D_i] \). This is because for non idling systems, \( E[D_i] = 0 \).

Different approaches to measuring fairness can lead to counter-intuitive deductions. The metrics can imply unfairness where there is no discrimination. Likewise, there can be cases of discrimination and the metrics do not imply them. We study such cases deeper in Sections 4, 5 and 6. Impressive fairness implied by these measures does not always translate into scheduler effectiveness in practice.

3 The Concept of Scheduler Effectiveness

Broadly, effectiveness of a scheduler depends on how well it treats all the users. Users (jobs) should get the best possible performance and some of the jobs should not be discriminated at the expense of others. The general behavior of queuing systems is determined by seniority and service requirements [18] of jobs. Job seniority can be looked at
as the time the jobs have spent in the queue and service requirements can be looked at the processor hours required to process a job. We note that ‘service requirements’ in parallel job scheduling is a bit more complicated than that used in [17]. In [17], they only consider the job runtime and in parallel job scheduling, there is an element of job size as well. Since we consider rigid jobs, the job size aspect cannot be transformed into the approach in [17]. Ideally, jobs would be processed in the order of seniority. However, parallel jobs have a big variance in sizes. A strict First Come First Served (FCFS) scheduler leads to system fragmentation and hence capacity loss. Much as FCFS gives a straightforward fair treatment of jobs, jobs do not perform equally. This is due to

1. **Job service requirements**: Different jobs have different sizes and runtimes. A job following short jobs is bound to wait less compared to that following a long job. Likewise, the time a job waits, while at the head of the queue increases with its size. Big jobs have to wait as the system accumulates enough processors.

2. **System and queue states**: Typical supercomputing environment have peak and off peak hours. Jobs that arrive in off peak hours enjoy better performance since they find shorter queues and a less busy system. The reverse is true for peak hours.

Fairness, therefore, does not necessarily imply performance equity. However, each job needs to get the best possible performance. An effective scheduler can improve the performance of some jobs as long as it does not do it at the expense of others. In fact, since capacity loss may be reduced, doing it may be at the benefit of others. This is applicable in schedulers like Conservative backfilling [19]. Aggressive backfilling [20] and backfilling with look ahead [23].

To evaluate scheduler effectiveness, we need to look for evidence that the best performing jobs are not doing so at the expense of others. Therefore, we can use the worst performing jobs. Jobs can be grouped by the characteristic that constrain their schedulability. Previous studies [10][14][11][15] show that group wise studies give detailed performance differences within the job stream.

### 4 Fairness by Measures of Dispersion

In this approach, we compute the extent of dispersion in performance in the job stream. This can be done by parameters like standard deviation (σ), variance (σ²), coefficient of variation (Cv) and fairness index (κ). Low dispersion implies a fair scheduler. The measures do not give consistent deductions. Due to the inconsistence, Vassupngayya et al. [8] observed that they are not sufficient to measure fairness. A high value of mean performance for example, lowers the value of Cv. In metrics like AWT and ART, we get a case where deterioration in performance implies an increase in fairness which is counter intuitive.

More to that, there are cases where an increase in dispersion does not come from selective job starvation. To illustrate this, we consider a job stream scheduled by FCFS and Conservative backfilling (BF) as a case study. Let us consider N online jobs J₁, J₂, ⋯, Jₙ scheduled by FCFS and BF. When scheduled by FCFS, AWT = μ and variance = σ² and Jᵢ has waiting time tᵢ. When scheduled by BF, let AWT = μ', variance = σ² and the waiting time for Jᵢ be tᵢ'. We compare the individual job performance, the average performance and the extent of dispersion among the jobs’ performance (fairness).

**Theorem 4.1** \( tᵢ' ≥ tᵢ \ ∀ Jᵢ \ i = 1, 2 \cdots N \)

**Proof** If no job backfills, CB ⇒ FCFS. Therefore:

\( tᵢ' = tᵢ \ ∀ Jᵢ \ i = 1, 2 \cdots N \)

In BF, a job backfills delays no other job. It starts processing earlier than when scheduled by FCFS (direct benefit). Removing a job from the queue for backfilling shortens the queue. This may lower the reservation times of jobs behind it and those yet to arrive (indirect benefit). If set S consists of jobs directly or indirectly affected by backfilling, then \( tᵢ' > tᵢ \ ∀ Jᵢ \in S \).

Since a job either benefits from BF or retains its FCFS reservation time:

\( tᵢ' ≥ tᵢ \)

**Theorem 4.2** \( μ ≥ μ' \)

**Proof** By definition:

\[
μ = \frac{1}{N} \sum_{i=1}^{N} tᵢ \quad \text{and} \quad μ' = \frac{1}{N} \sum_{i=1}^{N} tᵢ'
\]

From the validity of Theorem 4.1, Theorem 4.2 is also valid.

Let us define two non-negative parameters \( δμ \) and \( δtᵢ \) as:

\[
δμ = μ - μ' \quad \text{and} \quad δtᵢ = tᵢ' - tᵢ
\]

**Lemma 4.1** If we break the job stream into two disjoint sets S₁ and S₂, where \( Jᵢ \in S₁ \) if \( tᵢ' ≤ μ \) and \( Jᵢ \in S₂ \) if \( tᵢ' > μ \), then:

\[
\sum_{Jᵢ \in S₁} (μ - tᵢ')δtᵢ ≥ \sum_{Jᵢ \in S₂} (μ - tᵢ')δtᵢ
\]

**Proof** Since \( μ \) is a measure of central tendency for \( t₁', t₂', \cdots tᵢ' \), then:

\[
\sum_{Jᵢ \in S₁} (μ - tᵢ') = \sum_{Jᵢ \in S₂} (μ - tᵢ')
\]

In FCFS, a job keeps in queue because: (i) it has not reached the head of the queue or (ii) it is at the head of the queue but cannot fit in the available free processors.
Factor (i) affects all jobs equally while factor (ii) affects large jobs more. Small jobs therefore perform better than large jobs. This implies that small jobs dominate $S_1$ and large jobs dominate $S_2$.

BF utilizes processors that would be idle and small jobs mostly benefit from it [12]. Small jobs therefore get bigger performance improvement compared to large jobs. This implies that (i) for jobs in $S_1, |\mu - t_{m}^w| > 0$ and $\delta t_m \gg 0$ and (ii) for jobs in $S_2, |\mu - t_{m}^w| > 0$ and $\delta t_m \approx 0$.

Multiplying $\mu - t_{m}^w$ with a relatively large $\delta t_m$ makes LHS grow (absolutely) faster than RHS where $\mu - t_{m}^w$ is multiplied with a relatively small $\delta t_n$.

**Theorem 4.3** $\sigma' \geq \sigma$

**Proof** From the definition of variance:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (t_i^w - \mu)^2 \quad \text{and} \quad \sigma'^2 = \frac{1}{N} \sum_{i=1}^{N} (t_i^w' - \mu')^2$$

From (4), $\sigma'^2 - \sigma^2$ can be written as:

$$\frac{1}{N} \sum_{i=1}^{N} \left[(t_i^w' - \mu')^2 - (t_i^w - \mu)^2\right]$$

$$= \frac{1}{N} \sum_{i=1}^{N} (t_i^w' - \mu' + t_i^w - \mu)(t_i^w' - \mu' - t_i^w + \mu)$$

eliminating $t_i^w'$ and $\mu'$ using (2), we get:

$$\frac{1}{N} \sum_{i=1}^{N} (2(t_i^w - \mu) + \delta \mu - \delta t_i)(\delta \mu - \delta t_i)$$

$$= \frac{1}{N} \sum_{i=1}^{N} (\delta \mu - \delta t_i)^2 + \frac{2}{N} \sum_{i=1}^{N} (t_i^w - \mu)(\delta \mu - \delta t_i)$$

$$= \frac{1}{N} \sum_{i=1}^{N} (\delta \mu - \delta t_i)^2 + \frac{2}{N} \sum_{i=1}^{N} (t_i^w \delta t_i - \mu \delta \mu + \mu \delta t_i)$$

$$= \frac{1}{N} \sum_{i=1}^{N} (\delta \mu - \delta t_i)^2 + \frac{2}{N} \left[N \mu \delta t_i - \sum_{i=1}^{N} t_i^w \delta t_i\right]$$

$$\Leftrightarrow \frac{1}{N} \sum_{i=1}^{N} (\delta \mu - \delta t_i)^2 + \frac{2}{N} \left[\sum_{i=1}^{N} \mu \delta t_i - \sum_{i=1}^{N} t_i^w \delta t_i\right]$$

This simplifies to:

$$\sigma'^2 - \sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (\delta \mu - \delta t_i)^2 + \frac{2}{N} \sum_{i=1}^{N} (\mu - t_i^w)\delta t_i$$

(5)

Using the two sets defined in Lemma 4.1, RHS of (5) can be rewritten as:

$$\frac{1}{N} \sum_{i=1}^{N} (\delta \mu - \delta t_i)^2 + \frac{2}{N} \left[\sum_{m \in S_1} (\mu - t_m^w)\delta t_m + \sum_{n \in S_2} (\mu - t_n^w)\delta t_n\right]$$

(6)

($\delta \mu - \delta t_i)^2 > 0$ and using Lemma 4.1, RHS of Equation (6) is non negative.

**Theorem 4.4** If $\kappa$ and $\kappa'$ are the fairness coefficients for FCFS and BF respectively, then $\kappa' \leq \kappa$

**Proof** By definition, $\kappa = (1 + C_1^2)^{-1}$, from Theorem 4.3 and 4.2:

$$\sigma'^2 \geq \sigma^2 \Rightarrow \sigma' \geq \sigma \text{ and } \mu \geq \mu' \Rightarrow \mu \sigma' \geq \sigma' \mu' \Rightarrow \frac{\sigma'}{\mu} \geq \frac{\sigma}{\mu'} \Rightarrow C_V' \geq C_V \text{ Since } C_V' \text{ and } C_V \text{ are positive, then } C_V'^2 \geq C_V^2 \Rightarrow \kappa' \leq \kappa$$

Theorems 4.1, 4.3 and 4.4 show that:

(i) No job in BF perform worse than in FCFS, therefore BF does not starve any job as compared to FCFS

(ii) Dispersion in BF is higher than that in FCFS.

Dispersion, therefore, measures relative rather than absolute starvation. Since however jobs have different service requirements and find different queue and system states, it should be expected in practice. If we are to use dispersion as a measure of fairness, we would conclude that FCFS is fairer than BF yet the higher dispersion in BF is not a result of starvation. Actually, the increase in dispersion is due to different rates at which different jobs get improvements in performance as they are scheduled by BF instead of FCFS. There is, therefore, a practical loophole in using dispersion to deduce unfairness/starvation in parallel job scheduling.

5 **Fairness by Fair Start Time Analysis**

In fair-time analysis, we consider a job to be unfairly treated if it is delayed by another that arrived after it. This approach has some short comings.

First, this approach cannot compare schedulers where delays are not explicitly caused by later arriving jobs. Sometimes, a job can be delayed due to poor packing of jobs that arrived before it. Such a case is not catered for.

Secondly, equivalence in fairness does not always imply equivalence in satisfaction. Let us consider a case where we compare FCFS and BF. Since FCFS processes in the strict order of arrival, there can be no instance where a job is delayed by another that arrived after it. This implies that there is no unfairly treated job in the job stream. Let us also consider a case where the job stream is scheduled by BF. In BF, the jobs are allowed to violate the FCFS order but only if the jobs jumped are not delayed. This implies that in BF as well, there is no job which is delayed by another that arrived after it. This gives us a conclusion that FCFS and BF are equally as fair. The benefits of backfilling [19], in such a case, show no effect on fairness. This, definitely, is counter-intuitive.

Thirdly, this approach uses a dynamic point of reference. The fair start time of a job is dependent on the scheduling algorithm. This implies that a single job has different fair start times for different schedulers. The deviations, computed from these fair start times have different implications in real life. Let us consider two schedulers A and B where a certain job $J_i$ has fair start times 10 and 50.
If the actual start times are say 20 and 30 respectively, fair start time approach will indicate scheduler A to be more unfair compared to scheduler B. This is despite the fact that the job owner actually gets more satisfaction from scheduler A than from B.

Lastly, the approach considers the disadvantage caused to a job by a later arriving job. This, in some cases, may be small and local that it does not give the global extent and shift in user satisfaction. To illustrate this, let us consider a job \( J_k \) in a queue. We look at some of the circumstances that can alter its reservation time.

(i) Job \( J_k \) may jump (in cases like Fit Processors First Served [25] or backfilling) and get processed earlier than it would. Let the "jump" benefit caused be \( b_{ki} \).

(ii) Some of the jobs initially ahead of \( J_k \) jump (like in cases of FPFS and Backfilling) utilize the existing idle processors. This effectively increases the rate at which the load is processed and shortens the queue which may improve the reservation of \( J_k \). If the benefit \( J_k \) gets from the jumping of \( J_i \) is \( b_{ki} \), and a set \( S_B \) of jobs creates such benefits to \( J_k \), then the net extra reservation benefit \( \Delta R^1_k \) of \( J_k \) is given by:

\[
\Delta R^1_k = \sum_{J_i \in S_B} b_{ki} \tag{7}
\]

(iii) Some of the jobs initially ahead of \( J_k \) jump but interferes with the packing scheme, increases fragmentation and leads to a worse reservation of \( J_k \). This is more likely in approaches like aggressive backfilling and FPFS. Let the disadvantage suffered by \( J_k \) due to \( J_i \) be \( d_{ki} \), and the set of such jobs be \( S_D \). The total disadvantage in reservation \( \Delta R^2_k \) is given by:

\[
\Delta R^2_k = \sum_{J_i \in S_D} d_{ki} \tag{8}
\]

(iv) Some of the jobs initially behind \( J_k \) jump in such a way that \( J_k \) gets a disadvantage in reservation. This can be due to increased fragmentation or long execution times. If such jobs have a set \( S_{D'} \) and \( J_i \) causes a disadvantage \( d'_{ki} \), then the total disadvantage in reservation \( \Delta R^3_k \) suffered by \( J_k \) is given by:

\[
\Delta R^3_k = \sum_{J_i \in S_{D'}} d'_{ki} \tag{9}
\]

The net change in reservation for \( J_k \) is got by summing up the benefits and the losses for \( J_k \). If the net change is given by \( \Delta R \), then:

\[
\Delta R = b_{ki} + \sum_{J_i \in S_B} b_{ki} - \sum_{J_i \in S_D} d_{ki} - \sum_{J_i \in S_{D'}} d'_{ki} \tag{10}
\]

The approach in [5] used only \( \Delta R^3_k \) to evaluate fairness. Like shown in (10), other factors are also involved. A disadvantage \( \Delta R^2_k \) is not necessarily a net disadvantage. In some schedulers like aggressive backfilling, it rarely go beyond a few job from the head of the queue [21]. Chiang et al. [21] propose that if we are to prevent a big effect of backfilled jobs delaying jobs other than that at the head, making about the first four reservations is good enough. Sometimes, however, the a decision made on one job can have an effect that goes far deep into the queue [22]. Studies in [11] actually show that even with no consideration of job run-times (like in FPFS), all jobs achieve a net benefit.

The contribution of \( \Delta R^3_k \) can be substantial, negligible or non-existent depending on the scheduling environment. In cases where its substantial, fair-time approach gives a big estimation of starvation otherwise it does not.

### 6 RAQFM

RAQFM was proposed in [17] and analyzed in [18]. It uses the difference between the resources allotted to a job and what it is fairly entitled to. The RAQFM in [17] was meant for a single server facility. It is able to intuitively explain the fair most decisions to be made in naturally challenging scenarios. For example, it adequately handles a case where in a supermarket a customer with very few groceries is behind one with a lot of groceries. If used in parallel job scheduling, it brings in a problem of what is perceived to be fair and what is actually fair. It can claim discrimination where users get ideal service. For example, if two jobs needing 2 and 6 processors with runtime 5 each arrive simultaneously into a 10 node cluster, they immediately get the service. However, RAQFM will consider the first job to have temporal discrimination \((\frac{2}{10} - \frac{1}{2}) \times 5 = -\frac{3}{4}\) and the second job to have \((\frac{6}{10} - \frac{1}{2}) \times 5 = \frac{1}{2}\). The first job is discriminated while the second is favored yet both get the ideal service. As part of future work in [17], the modification of RAQFM to fit multi-server cases is still in progress.

### 7 Experimental Evaluation

We now experimentally compare deductions from simultaneous comparison of group wise performance with a combination of job stream performance and fairness.

#### 7.1 The model

We consider a system of \( N \) homogeneous clusters \( C_1, C_2, \cdots, C_N \) served by a single queue and scheduler. The clusters are connected by a fast WAN with negligible communication overheads.

Jobs are online, rigid, and can be broken into multiple components and co-allocated. Job durations are unknown to the scheduler but finite. Job sizes are generated from a distribution \( D(q) \) defined on an interval \([n_1, n_2]\) \((0 < n_1 < n_2)\). In \( D(q) \), the probability \( p_i \), that a job has size \( i \),
is given by:

\[ p_i = \begin{cases} 
3q_i \quad \text{if } i \text{ is a power of 2} \\
\frac{q_i}{Q} \quad \text{if } i \text{ is not a power of 2}
\end{cases} \quad (11) \]

Jobs with size greater than a threshold \( \text{thres} \) are broken into components.

### 7.2 The Scheduling algorithms

We consider three scheduling algorithms.

The first is FCFS. Jobs are queued and allocated to clusters in their (strict) arrival order.

The second is Fit Processors First Served (FPFS) [25]. This is like FCFS but if the job at the head of the queue cannot fit in the system, the first job deeper in the queue, that fits is scheduled. The job at the head of the queue is protected from starvation by fixing the number of times it is jumped to \( \text{maxJumps} \). If it is jumped \( \text{maxJumps} \) times, no other job is allowed to jump it until enough space has been freed to start its processing. We use FPFS(\( x \)) to represent FPFS when \( \text{maxJumps} = x \).

The third is the Greedy multi-cluster scheduler [24]. Jobs are queued in the arrival order. Each job has a priority indicator which is the product of the time it has spent in the queue and its hardness value. The hardness value is an approximation of job schedulability (size in this case). The scheduler picks the highest priority job from the first \( \text{depth} \) jobs that fits in the available processor hole. Like in FPFS, the job at the head of the queue is protected from starvation by limiting the number of times it can be jumped to \( \text{maxJumps} \). We use Greedy(\( j, d \)) to represent the greedy scheduler when \( \text{maxJumps} = j \) and \( \text{depth} = d \).

### 7.3 Performance and Fairness Evaluation

We use ART performance metric. To measure fairness, we use \( \sigma \) and \( C_V \). We use size to estimate job schedulability and generate four size based groups \( Q_1, Q_2, Q_3, \) and \( Q_4 \). Groups are bounded by the (size) lower quartile, median and upper quartile.

### 7.4 Experimental set up

We simulate a 5 cluster system of 20 processors per cluster. Jobs are generated from \( D(0.85) \) over [1, 38]. Job inter-arrival and execution times are exponentially distributed with mean 0.64 and 10.0 respectively (load = 0.786). We take \( \text{thres} = 11 \) and jobs with \( \text{size} > \text{thres} \) are broken into up to 4 components. The probability that a job \( \text{size} > \text{thres} \) is broken into 2, 3, or 4 components is \( \frac{1}{2} \). To break a job of size \( x \) into \( n \) components, the first \( n - 1 \) components are made to have a width \( \lceil \frac{x}{n} \rceil \) and the \( n^{th} \) component to have \( x - (n - 1) \times \lceil \frac{x}{n} \rceil \). Results are recorded with a maximum absolute error of 0.05 at 95% confidence interval. We consider FCFS, FPFS(10), FPFS(50), Greedy(10,10), Greedy(10,50), Greedy(50,10) and Greedy(50,50).

### 8 Results

#### 8.1 Performance and Fairness

We summarize the performance and fairness (by \( \sigma \) and \( C_V \)) in a \( 2 \times 2 \) matrix (Figure 1). We observe that \( \sigma \) and \( C_V \) do not produce consistent deductions on fairness. The order of fairness among schedulers is not the same for \( C_V \) and \( \sigma \). This is in line with [8] that measures of dispersion are insufficient in evaluating fairness.

#### 8.2 Group wise evaluation

We now present job stream and group wise performance in Figure 2. We observe that FCFS and Greedy(50,50) give the worst and best performance respectively. The same
trend is observed in group wise evaluation. However, the entire job stream and group wise performance trends are not always the same. Greedy(10,50) and FPFS(10) are equal for the entire job stream but Greedy(10,50) is better for group wise evaluation. More to that, FPFS(10) is better than Greedy(10,10) for the entire job stream though the reverse is true for group wise evaluation. The differences in performance and fairness imply that using performance alone does not always give a good idea of scheduler effectiveness. The extent of job starvation (which is inadequately measured by measures of dispersion) needs to be put into consideration.

9 Discussion

Choosing an appropriate performance metric is a hard decision to make [2]. This is because different scheduling scenarios can have different ways quality of service can be envisaged. After choosing a performance metric, its outcome can be misleading because of relative job starvation. This is a general concept of scheduler fairness. Fairness has been measured in other queuing systems and several metrics have been proposed. Those metrics, however, are not necessarily appropriate in the parallel job scheduling case. They therefore lead to counter intuitive conclusions. The main problem is how to accurately represent starvation in an environment where jobs are having different requirements, varying arrival rates and need not to have the same performance for a scheduler to be fair. Since fairness is meant to measure the user’s feel of discrimination, there is a need to model what the user expects in real life. The deviation can be used to measure how close to the ideal the scheduler reaches. An approach like dispersion assumes equal performance which actually is not the case in practice. This makes the deductions being made on a non-realistic foundation.

While in this paper we do not propose an approach to measure fairness, we examine the existing approaches for their representation in the real life scheduling scenario of parallel jobs. We also investigate how one of the existing approaches relates with deductions from group wise evaluation. Deductions from fairness and performance evaluation do not rhyme with those of group-wise evaluation. Since the approaches may misrepresent users needs, their deductions do not necessarily give an accurate view of user satisfaction.

10 Conclusion and Future Work

We have shown that the commonly used fairness metrics do not always accurately represent the users’ needs as far as discrimination and favoritism is concerned. There are circumstances where a fair scenario is indicated as un fair and vice versa. This leads to a problem of the applicability of the metrics as well as the reliability of the deductions.

More research on this work can be on coming up with robust metrics for fairness in parallel job scheduling. There is however work in this direction (Raz et al [17] indicate it as ongoing work). There is also a need to study in more detail the possible pitfalls that may render a fairness metric unrepresentative of the real life environment.

References


