World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:10, 2009

Impact of Fair Share and its Configurations on Parallel Job Scheduling Algorithms

Sangsuree Vasupongayya

***Abstract*—To provide a better understanding of fair share policies supported by current production schedulers and their impact on scheduling performance, A relative fair share policy supported in four well-known production job schedulers is evaluated in this study. The experimental results show that fair share indeed reduces heavy-demand users from dominating the system resources. However, the detailed per-user performance analysis show that some types of users may suffer unfairness under fair share, possibly due to priority mechanisms used by the current production schedulers. These users typically are not heavy-demands users but they have mixture of jobs that do not spread out.**

***Index Terms*—Fair share, Parallel job scheduler, Backfill, Measures.**

I. Introduction

FAIR share is one of the important goals of parallel job scheduling policies and is supported on many production job schedulers. However, the meaning of fair share and the impact on scheduling performance have been largely ignored in previous research. As a result, it is rather difficult for the system administrators to know how to configure fair share policies and for the users to know what to expect. The goal of this study is to provide a better understanding of fair share policies and their impact on other scheduling performances.

The Fair Share Scheduling [1] was originally proposed to extend Unix time-sharing systems. Fair Share Scheduling allocates resources fairly among competing users or groups, as opposed to processes. A similar idea of fair share is adopted in many production parallel job schedulers. The detailed implementations of fair share of different schedulers vary, but they all rely on some priority mechanisms to implement fair share similar to that of the original Fair Share Scheduling. However, these production parallel job schedulers are in general non-preemptive, it is not clear whether using a similar priority mechanism is effective in achieving fair share. In fact, the only study [2] on this subject concluded that fair share has no effect on performance of jobs of each user. The conclusion is reached by showing that there is no correlation between the share assigned to each group and the performance of jobs of each group. However, many factors could impact job performance, such as the load condition when jobs arrive, the job sizes, the particular fair share policies used, and the resource demand of each group. Thus, more study is required to further understand the impact of fair share policies on scheduling performance. The goal of this study is to provide a better understanding of fair share policies and their impact on scheduling performances.

The remaining of this paper is organized as follow. In Section II, fair share models supported in four well-known production job schedulers are reviewed. In Section III, the experimental setting is described. Sections IV-VI present the evaluation results. A conclusion is given in Section VII.

S. Vasupongayya is with the Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University, Hat Yai, Songkhla, 90112, Thailand. e-mail: sangsuree.v@psu.ac.th.

This work is supported by the Faculty of Engineering, Prince of Songkla University, under grant no. ENG-52-2-7-18-0031-S.

II. Fair Share in Production Schedulers

In this section, fair share and its configurable parameters supported in four well-known production job schedulers: PBS [3], [4], LSF [5], IBM LoadLeveler [6], Maui/Moab [7], [8] are reviewed. Since Moab is an extension of the Maui scheduler, Moab and Maui are viewed as one scheduler. Note that the terms used in this study may be different from those in the production schedulers.

A. Fair Share Models

A relative fair share model is supported in all four schedulers. To achieve fair share, a fair share priority is assigned to each user or group. The fair share priority of each user or group is a function of the *entitled share* and the *cumulated usage*. The entitled shares define the importance of each user or group relative to other users or groups. The entitled shares directly or indirectly specify the amount of resources each user or group is entitled to use. In other words, the entitled share is defined as the amount of resources that the user entitled to according to the current fair share policy. For example, an equal fair share policy can assign each user an equal amount of resources. While, the cumulated usage of each user is the amount of resources that the user currently used so far. Both the entitled share and the cumulated usage are dynamically calculated within a fair share window.

There are two implementations of the relative fair share model, which include the remaining share and the ratio of the entitled share and the cumulated usage. The remaining share is defined as the difference between the entitled share and the cumulated usage at the time when the priority is computed. These two implementations may look different however they are equivalent. That is, a user with larger remaining share relative to other users also has a higher ratio of the entitled share and the cumulated usage relative to other users.

When a fair share policy is in used, each user will have his/her fair share priority dynamically computed. All jobs belong to a user are given the same fair share priority. A

job priority will be adjusted up or down according to the fair share priority of its owner. For example, the remaining share is used to adjust the job priority up when the value is positive and down when the value is negative.

In addition, LoadLeveler and Maui/Moab also allow the entitled share to be defined as the share upper-bound or the share lower-bound. When the entitled share is defined as the share upper-bound, if the user or group has used more than the entitled share, their job priorities are adjusted down. When the entitled share is defined as the share lower-bound, on the other hand, if the user and group has used less than the entitled share, their job priorities are adjusted up.

B. Fair Share Window

Fair share policy has a fair share window–a configurable parameter. To compute the fair share priority, the usage of each user must be cumulated. Usually, the usages include the resources that have been actually used so far in the current fair share window. The future usages–resources expected to be used by the currently executing jobs or the jobs holding reservations–are not included in the usage calculation. However, LSF factors in some future usages committed so far into its usage calculation. LSF achieves this by defining its fair share policy as a weighted sum of the total actual CPU time and wall time used and the number of jobs that are currently running or having reservations. The weights are configurable. However, finding a good set of weights can be troublesome since each measure affects the scheduling performance differently. Even one good set of weights is found, it may not work during another period of time due to the changes in workload.

Normally, the default window size is one day (e.g., the default in the PBS and Maui/Moab schedulers) or seven days, except it is five hours in LSF. Similar to the original fair share, the usage from previous fair share windows can be decayed and carried over the current window. The decay factor is also configurable.

III. Experimental Setting

In this section, the policies are defined, the measures and the workload used in this study. The performance of all policies is evaluated by an event-driven simulation. The ten monthly workloads that ran on an Intel Itanium Linux cluster (IA-64) at the National Center for Supercomputing Applications at the University of Illinois at Urbana-Champaign during June 2003 to March 2004 are used as the input to the simulator. The performance measures of each month are computed for jobs submitted during the month. To be realistic, however, each simulation of a given month includes a one-week (from previous month) warm up, and a cool-down period in which jobs (from next month) continue to arrive until all jobs submitted during the month for which the performance measures are computed have started. The cool-down period is typically a few days only. To understand the full potential of fair share policies, the simulator uses the perfect runtime information of each job thus the impact of inaccurate runtime estimates is eliminated.

TABLE I
Information of NCSA IA-64 workload

Month	Proc. demand	#users	#jobs	#jobs per user		
				Avg.	Median	Max.
6/03	82%	73	2191	30.0	8.0	659
7/03	89%	68	1400	20.6	8.0	145
8/03	79%	73	3221	44.1	8.0	1873
9/03	72%	74	3057	41.3	15.0	703
10/03	71%	75	4149	55.3	15.0	1151
11/03	73%	81	3443	42.5	17.0	665
12/03	74%	61	3521	57.7	14.0	635
1/04	73%	53	3156	59.5	17.0	679
2/04	74%	73	3969	54.4	28.0	541
3/04	75%	70	3466	49.5	15.5	1234

Month	Job size (NT)			demand (NT) per user		
	Avg.	Median	Max.	Avg.	Median	Max.
6/03	34.5	0.8	960.0	1034.7	24.0	24071
7/03	60.6	1.3	1536.0	1247.4	145.7	16719
8/03	23.4	0.0	1536.0	1031.6	120.0	14346
9/03	21.7	0.1	912.0	895.5	72.5	18499
10/03	16.3	0.4	912.0	899.5	114.7	8060
11/03	19.5	0.7	1536.0	827.1	27.6	10183
12/03	20.1	1.1	1152.0	1159.3	23.2	17776
1/04	22.1	5.1	1920.0	1313.6	317.4	10340
2/04	16.6	0.3	1824.0	900.3	93.3	8931
3/04	20.6	0.0	1832.8	1018.0	46.1	12892

Month	Job size (N)			Job size (T)		
	Avg.	Median	Max.	Avg.	Median	Max.
6/03	12.1	4.0	128	1.4h	0.20h	12h
7/03	23.5	8.0	128	1.9h	0.18h	12h
8/03	7.3	1.0	128	1.1h	0.00h	12h
9/03	9.1	1.0	128	1.4h	0.03h	12h
10/03	5.0	1.0	128	2.0h	0.13h	12h
11/03	6.0	1.0	128	2.5h	0.15h	12h
12/03	5.6	1.0	128	3.6h	0.33h	24h
1/04	10.7	2.0	128	4.5h	1.10h	24h
2/04	5.0	2.0	128	3.1h	0.11h	24h
3/04	5.8	1.0	128	2.4h	0.00h	24h

A. Workloads

Information of the workload is shown in Table I. The information includes the processor demand (Proc. demand), the number of users (#users), the number of jobs (#jobs), the number of jobs per user (i.e., average, median, and maximum), and the job size information (i.e., average, median, and maximum processor (N), runtime (T), and processor-hour (NT)) of each month. The total available nodes of the system is 128 dual-processor nodes. The runtime limit of June 2003 to November 2003 was 12 hours and it was increased to 24 hours after that. The processor demand of all jobs is typically in the range of 70-80%, but July 2003 has a higher load (89%). For more details see [9].

B. Policies

Two fair share policies using the relative fair share model described previously in Section II-A was studied. Their scheduling performances are compared against FCFS-backfill and LXF-backfill policies.

Under backfilling scheme, jobs are considered for scheduling in the order of their priority; if a job cannot be started due to insufficient resources; the job is given a reservation (i.e., the earliest time sufficient resources become available for the job); lower-priority jobs can be backfilled on idle resources as long as they do not delay any reservation. In the simulation of backfill policies, only the highest-priority waiting job receives the

reservation because giving reservations to more jobs does not find to be beneficial for the workloads studied. FCFS-backfill considers jobs for scheduling according to their arriving order thus it usually achieves a good maximum wait performance. While, LXF-backfill considers jobs for scheduling according to their slowdown (largest slowdown first), thus it usually achieves good performances on average performance measures (both slowdown and wait).

The fair share policies, in this study, prioritize jobs according to the fair share priority (i.e., higher the share higher the priority). The tie is broken by the job arriving order. Similar to the production job schedulers, the fair share policies in this study also support backfilling. Only equal sharing among users is considered in this study. Fair share window is a configurable parameter for both fair share policies. In this study, the fair share window varies from 4 hours to 7 days (i.e., the typical value used in most production job schedulers). To eliminate the impact of the decay factor, the cumulated usage of each user is reset to zero at the beginning of each fair share window. In other words, the usage in a window does not carry over to the next window. The impact of decay factor will be left for future work. The two fair share policies are described below.

(1) *RelShare(fw)*: defines the fair share priority of each user to be $\frac{entitled\ share}{cumulated\ usage}$ where *fw* specifies the fair share window. For example, RelShare(1d) uses a one-day window that is the cumulated usage of a user includes the actual usage of the user from the beginning of the day until the current time and the expected usage of resources allocated to the currently running jobs and the job with reservation. However, only the expected usage of resources within the current day (for *fw*=1d) is included.

(2) *RelShare(fw) w/o expected usage*: is similar to *RelShare(fw)*. However, the cumulated usage of *RelShare(fw) w/o expected usage* only contains the actual usage so far, excluding any expected usage.

C. Measures

Measures required to define fair share policies and to evaluate to what extent the fair share is achieved are defined in this section. Each measure can be computed over any period of time.

(1) *entitled share*: To compute a fair share priority value, a cumulated entitled share must be defined. The cumulated entitled share of a user during a given period of time is simply the cumulated entitled nodes over that period of time. For example, if a user is entitled to 10 nodes in a 20-minute period, and 15 nodes in the next 40 minutes, then the user has an entitled usage of 800 node-minutes or about 13.3 node-hours, during that one-hour period of time.

At any point of time, the entitled nodes of each user is proportion to the shares assigned to the user, i.e., 128 * share assigned to the user / total shares assigned to active users. Note that an active user at the current time is a user who has any job waiting or running at the current time. In the equal-share policy, the initial entitled nodes of each user is simply the maximum number of nodes divided by the number of currently active users.

If a user's entitled node is larger than the user's total requested nodes, i.e., the total requested nodes of currently waiting or running jobs that belong to the user, then the entitled nodes of the user is reset to the user's current total requested nodes. The entitled nodes are recursively computed for the remaining active users and remaining nodes.

(2) *dev*: To evaluate to what extent the fair share is achieved, a per-user fair share measure namely dev (a.k.a. deviation in node hours) is defined. Deviation in node hours is defined to be the cumulated actual usage minus the cumulated entitled share of each user over a given period of time. Note that a positive value means an *over-share*, and a negative value means an *under-share*.

IV. Performance of fair share policies

In this section, the impact of using RelShare(1d) and RelShare(1d) without expected usages is studied, in comparison with FCFS-backfill and LXF-backfill. Both RelShare policies use a one-day fair share window.

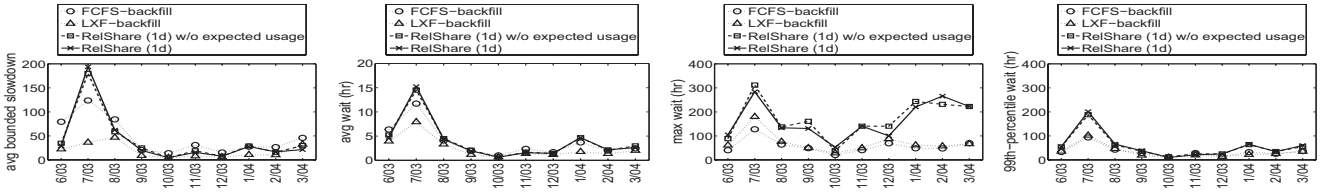
Figure 1(a)-(d) plot the average bounded slowdown, average wait, maximum wait, and 99th-percentile wait, respectively, of FCFS-backfill, LXF-backfill, RelShare(1d) w/o expected usages and RelShare(1d). To reduce the impact of a high slowdown of very short jobs, a bounded slowdown is used instead of a slowdown. That is, all jobs with runtime less than 1 minute are treated as 1-minute jobs when compute their slowdown (i.e., (wait time + runtime)/ max(1,runtime)). Figure 1(a)-(b) shows that all four policies have fairly comparable average bounded slowdown and average wait each month, except 6/03 and 7/03. RelShare(1d) has worse maximum wait time than those of both backfill policies in each month studied. Note that only a few percentage of jobs under RelShare(1d) in each month incurred a very poor wait time. This can be seen from Figure 1(d), which shows that RelShare(1d) has similar 99th-percentile wait time as that of FCFS-backfill for each month, except 7/03 and 1/04.

As shown in Figure 1(a)-(d), both fair share policies have fairly comparable scheduling performances each month. The similar performance of RelShare(1d) w/o expected usages and RelShare(1d) shows that the poor maximum wait performance of RelShare is not the result of using expected usages in the cumulated usage calculation.

The results in this section suggest that both fair share policies have little impact on both average performance measures. However, it is rather unexpected to find such a large gap in the maximum wait between RelShare(1d) and both backfill policies. Another point of these results is that if only the average performance measures were studied, an incorrect conclusion would have been drawn that fair share policies have little impact on the scheduling performance.

V. Detailed analysis of fair share

In this section, a detail analysis of per-user performance is studied to find whether the fair share policies are indeed more fair than both backfill policies. First, the users who suffer under RelShare policies are analyzed. Next, the analysis moves to the users who benefit under RelShare policies. Then, conclusions are drawn from both results.



(a) Avg. bounded slowdown (b) Avg. wait (c) Max. wait (d) 99th-percentile wait

Fig. 1. Performance of Share policies versus backfill policies

TABLE II
USERS WITH THE WORSE DEGRADATION IN WAIT TIME UNDER RELSHARE(1D)

(a) Performance of each user

User	month	FCFS-backfill dev	FCFS-backfill MaxW	FCFS-backfill AvgW	FCFS-backfill AvgX	LXF-backfill dev	LXF-backfill MaxW	LXF-backfill AvgW	LXF-backfill AvgX	RelShare(1d) dev	RelShare(1d) MaxW	RelShare(1d) AvgW	RelShare(1d) AvgX	Type
8	7/03	-2907.8	82.8h	21.1h	168.1	-1606.5	26.6h	4.6h	34.7	-7245.0	280.7h	80.7h	907.2	S1
21	7/03	-1706.0	65.6h	11.4h	126.6	-1611.1	63.5h	6.3h	28.9	-667.8	109.5h	11.9h	103.8h	S2
24	7/03	-1787.6	114.8h	26.1h	459.5	-894.3	30.7h	8.8h	122.4	-4592.9	283.8h	73.3h	2033.2	S1
49	7/03	1532.0	66.2h	11.5h	106.7	-1373.0	179.8h	16.0h	42.6	68.6	163.5h	23.7h	108.4	S2
71	7/03	6585.3	105.1h	29.1h	148.2	3803.6	143.5h	27.7h	45.8	4582.9	211.2h	31.8h	94.1	S2

(b) Job size of each user in July 2003

User	#jobs	Total NT (hr)	N Min	N Avg	N Med	N Max	T (hrs) Min	T (hrs) Avg	T (hrs) Med	T (hrs) Max	N×T (node-hrs) Min	N×T (node-hrs) Avg	N×T (node-hrs) Med	N×T (node-hrs) Max
8	53	1452	1	75.8	128.0	128	0.0	1.6	0.2	12.0	0.0	27.4	15.6	180.0
21	49	4030	1	45.7	32.0	128	0.0	1.1	0.4	6.0	0.0	82.3	6.4	768.0
24	44	827	1	57.4	32.0	128	0.0	0.5	0.3	3.0	0.0	18.8	8.1	95.5
71	58	11693	32	48.8	32.0	100	0.0	3.2	1.6	10.0	0.6	201.6	51.1	1000.0
49	83	6051	1	12.7	1.0	32	0.0	3.9	1.5	12.0	0.0	72.9	1.5	384.0

A. Who suffer under fair share

To study what users suffer poor performance under RelShare(1d), Table II provides the information of the top users who suffer the largest degradation in the wait time performance under RelShare(1d) than that under FCFS-backfill in July 2003. Note that the results of July 2003 are selected to demonstrate the problem due to the largest performance differences among policies studied. The results of other months are similar, however the differences are smaller.

For each user, Table II shows dev, the maximum wait (MaxW), average wait (AvgW), and average bounded slowdown (AvgX) of all jobs submitted by the user during the month under each policy. In the case of users who have a large over-share under FCFS-backfill (e.g., #49 and #71), it is reasonable to reduce its over-share under the fair share policy. Similarly, users who already suffer a large under-share under LXF-backfill (e.g., #21), it is reasonable to reduce its under-share under the fair share policy. However, users who already suffer a large under-share under FCFS-backfill (e.g., #8 and #24), now suffer an even larger under-share under RelShare(1d). The results not only are counter-intuitive, but also indicates that there is a problem in using the fair share policies implemented using a priority function.

To shed some light into the causes and problems of fair share policies, The characteristics of these users are studied in more detail including how they are treated differently under different policies.

Table II(b) shows the statistics of the job size (N, T, and N×T) of each user during July 2003. All four users have a significant number of jobs, compared with an average of 21 jobs and a median of 8 jobs per user during that month.

The users shown in Table II represent two types of users who incur a large performance degradation under RelShare(1d): Type-S1 users have a mixture of large-node jobs with small-node jobs; Type-S2 users has a heavy daily demand on most days or has a heavy daily demand on some days. The demand of Type-S2 users will be seen more clearly in Figure 3 later.

Note that Type-S1 users may not have a high demand but they do have some difficult jobs (i.e., large jobs or long jobs and large-long jobs) together with other jobs. While, Type-S2 users may not have heavy-demand on all days but simply have a heavy-daily demand on some days. A monthly heavy-demand users, on the other hand, may in fact have only a light to moderate demand each day, but the cumulated demand of the users over the month is high because the user submits the jobs on most days of the month.

B. Why they are suffered

For a Type-S1 user, the problem happens when a large job of the user queues behind a small job of the same user; once small jobs of the user are started, the fair share priority of the user is lowered, which may prevent the first large job of the user from receiving a reservation, even if the job is the oldest in the queue. Since it is difficult to find enough resource to start large jobs, the delay in reserving resources for such jobs can cause an extended delay of starting these jobs.

Further analysis finds that users who submit short jobs together with long jobs even with small number of nodes, may also suffer degrade in performance under RelShare(1d) due to the delay in receiving a reservation for the first long job of the user because the short jobs of the user gets the resources

and lowers the fair share priority of the user. However, the degree of suffering is not as serious as Type-S1 users because ,in general, large node jobs are more difficult to schedule than long jobs. For a Type-S2 user, the problem is simply that the user has too much load.

To further illustrate the problem incurred by Type-S1 users under RelShare(1d), Figure 2(a)-(c) shows how user #8 (in July 2003) is treated under RelShare(1d), FCFS-backfill and LXF-backfill, respectively. In each graph, the number of waiting jobs of the user at the beginning of each day ("previous wait"), the number of new jobs submitted by the user during each day ("new submitted"), and the number of started jobs of the user during each day ("started today") under the given policy are plotted. Figure 2(a) shows that under RelShare(1d), user #8 typically starts 1-2 jobs and no more than four jobs per day, except for one day (July 2, 2003). As a result, the user has a large backlog each day. In contrast, under FCFS-backfill (Figure 2(b)) and LXF-backfill (Figure 2(c)), the user can start 4-7 jobs per day in many days, and when the user has a backlog, it is cleared in a few days and no more than a week.

Figure 3(a)-(c) plot the demand, usage, and entitled share of user #8 for each day under the three policies. Similarly, Figure 3(d)-(i) plot the results for Type-S2 users (i.e., #71 and #49). The average demand of all users of each day is also shown for a contrast. The *demand* of the user in each day is the total remaining node hours that can be used today by jobs that were already running at the beginning of the day, plus the total node hours that can be possibly used during that day by all "previous wait" jobs and "new submitted" jobs of the user. The usage of the user in each day is simply the total node hours actually used by any job of the user in that day. The entitled share was defined in Section III-C.

Please note two important details here. First, the demand in these graphs include the demand of backlog each day as well resulting in the higher value than the total node-hours of the user for some users. Second, only the node hours that can be possibly served on a given day (assuming that the jobs can start right the way) are included in the demand. For example, if a job, requiring 1 node and 10 hours, was submitted 1 hour before the end of the day, the job could receive at most one node-hour today; thus, only one node-hour of the job is included in the demand of the user who submitted the job. The rest will be included in the next day demand.

As shown in Figure 3(a)-(c), user #8 has much lower demand per day, yet uses much less than its entitled share under both FCFS-backfill and RelShare(1d). While, there is only slight difference in the entitled share and the usage under LXF-backfill.

In contrast, user #71 has a much higher per-day demand (up to 20-30 times) than the average demand of all users for many days (Figure 3(d)-(f)). The user is able to use his/her entitled share or more in most days under RelShare(1d), but it can use a lot more than the entitled share under FCFS-backfill and LXF-backfill in most days. Figure 3(g)-(i) show an example of users who have small-long jobs (i.e., user #49) which can have much higher per-day demand than the average demand of all users for many days. Similar to user #71, user #49 use his/her entitled share or more in most days under FCFS-backfill while the user is not allowed to use as much under RelShare(1d).

TABLE III
INFORMATION OF BENEFIT AND SUFFER USERS EACH MONTH

	Fraction of all users			
	$>$ 10 hours		$>$ 0 hour	
Month	Benefit	Suffer	Benefit	Suffer
6/03	21.9%	8.2%	61.6%	20.5%
7/03	32.4%	23.5%	48.5%	41.2%
8/03	19.2%	23.3%	42.5%	47.9%
9/03	14.9%	10.8%	55.4%	32.4%
10/03	9.3%	8.0%	56.0%	34.7%
11/03	16.0%	6.2%	56.8%	28.4%
12/03	3.3%	14.8%	44.3%	37.7%
1/04	7.5%	11.3%	39.6%	41.5%
2/04	20.5%	17.8%	46.6%	46.6%
3/04	22.9%	14.3%	50.0%	34.3%

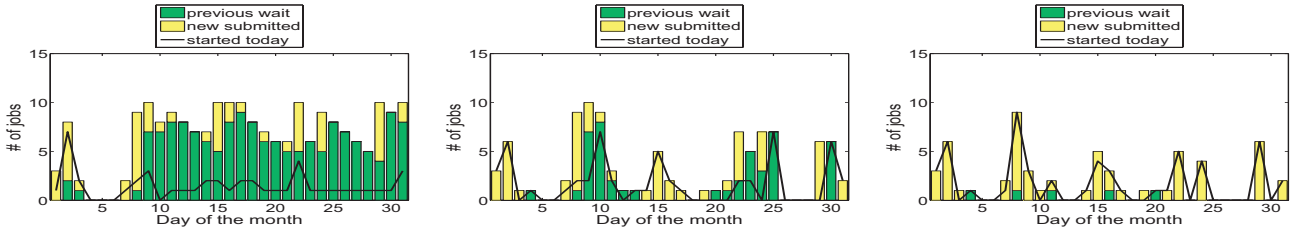
Key conclusions from the above results are (1) RelShare(1d) may reduce the chance of heavy-load users from dominating the system resources. (2) Some users may suffer a very poor performance and larger under-share under RelShare(1d). These users include users who submit large jobs with small jobs in between large jobs and users who submit long jobs with short jobs in between long jobs.

C. Who benefit under fair share

With some users suffering worse performance under RelShare(1d), some users are benefited.

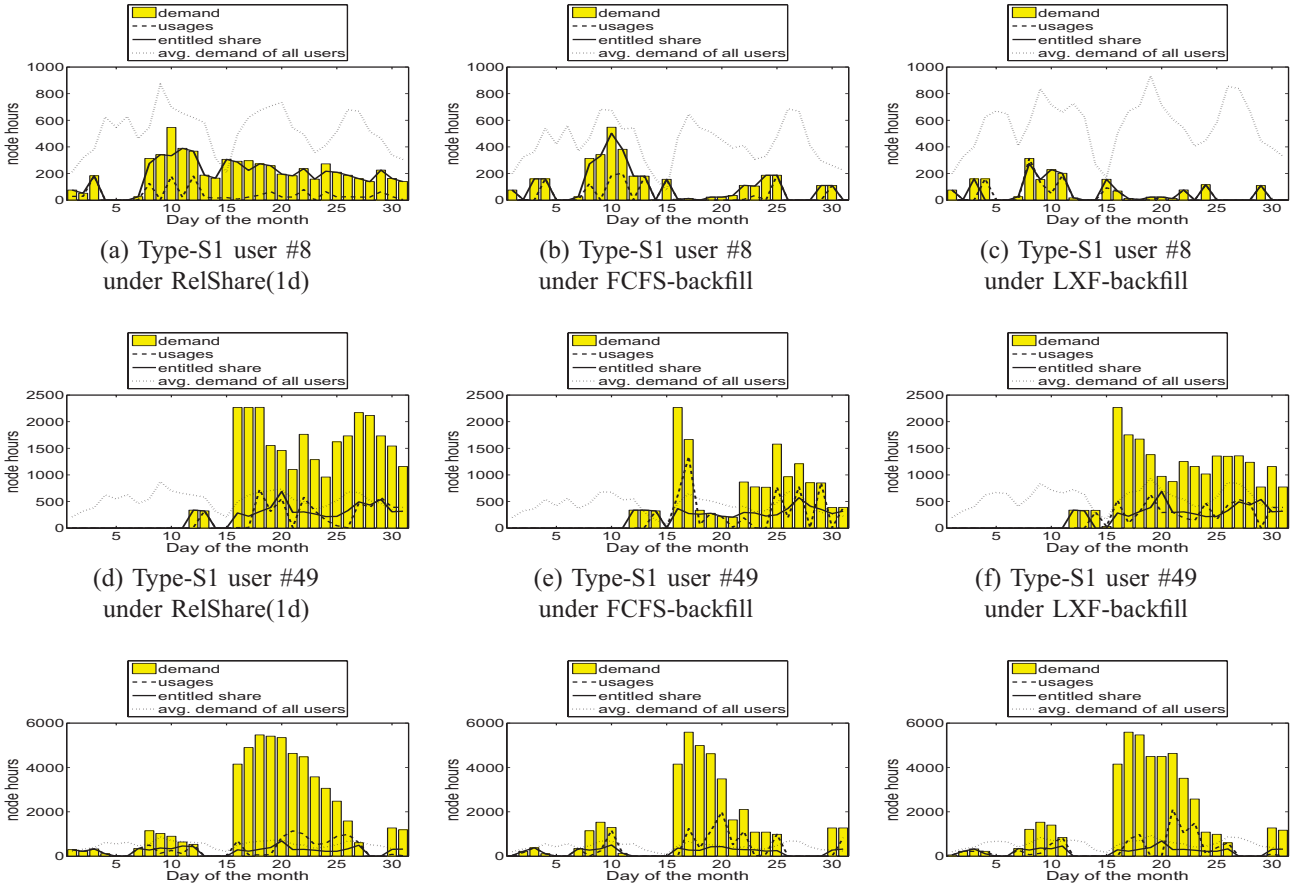
Table III provides the fraction of users who considerably benefit and suffer under RelShare(1d). i.e., if the maximum wait time of a user under RelShare(1d) is lower than that under FCFS-backfill policy, the user is considered benefit, otherwise, the user is consider suffer. In fact, the number of users who considerably benefit under RelShare (1d) (by a reduction of $>$ 10 hours in maximum wait) is typically similar or larger than the number of users who considerably suffer under RelShare(1d) (by an increase of $>$ 10 hours in the maximum wait time), except 12/03 and 1/04. The users who benefit under RelShare(1d) are usually those who (1) have a few very short jobs (even if they have a large number of nodes), (2) have only fairly small-node jobs ($\leq$ 32), or (3) have difficult jobs but spread their jobs out such as those shown in Table IV.

Table IV shows the users who have a mixture of large jobs together with small jobs (i.e., #70 in 7/03, #113 in 8/03 and #103 in 11/03) or have a mixture of long jobs together with short jobs (i.e., #116 in 11/03) but these users spread their jobs out. As expected, these users are benefited from RelShare(1d). By spreading their difficult jobs out, the problem of delaying reservation of difficult jobs (as that in the type-S1 users) is eliminated. For example, user #70 and #103 are under-share under FCFS-backfill, but they are slightly over-share under RelShare(1d). One could argue that these users do not have a job as large as the entire system, however their jobs are still difficult compared with an average of jobs submitted during the month. For example, user #70 has a average job size of 163.5 node-hours while the average job size of July 2003 is 60.6 node-hours. LXF-backfill policy has problem with long



(a) Type-S1 user #8 under RelShare(1d) (b) Type-S1 user #8 under FCFS-backfill (c) Type-S1 user #8 under LXF-backfill

Fig. 2. Per-day number of jobs of an example users who suffer under RelShare(1d) (July 2003)



(a) Type-S1 user #8 under RelShare(1d) (b) Type-S1 user #8 under FCFS-backfill (c) Type-S1 user #8 under LXF-backfill

(d) Type-S1 user #49 under RelShare(1d) (e) Type-S1 user #49 under FCFS-backfill (f) Type-S1 user #49 under LXF-backfill

(g) Type-S2 user #71 under RelShare(1d) (h) Type-S2 user #71 under FCFS-backfill (i) Type-S2 user #71 under LXF-backfill

Fig. 3. Per-day node-hours of example users who suffer under RelShare(1d) (July 2003)

jobs as shown in the increasing under-share of user #166 whose jobs are mostly long. Please note that user #166 is an example of users with long jobs and submit short jobs together with long jobs but the jobs are spreading out.

The results in this section suggest that users with mixture of jobs should spread their jobs out to reduce the effect of delaying reservation of difficult jobs. Users who have a few very short jobs and users who have fairly small jobs are treated more fair under RelShare(1d) because the heavy-demand users cannot overtake the system resources as they do under both traditional backfill policies.

D. Are fair share more or less fair?

In this section, the overall fair share of all users under RelShare(1d), FCFS-backfill, and LXF-backfill are evaluated.

First, Table V shows for each policy in each month, the number of users who are active in that month and the fraction of users with more than 50 node-hours of under-share (i.e., cumulated entitled share - cumulated usage $>$ 50), and those with more than 50 node-hours of over-share (i.e., cumulated

TABLE IV
USERS WITH MIXTURE OF LARGE AND SMALL JOBS BUT SPREAD THEIR JOBS

(a) Performance of each user

User	month	FCFS-backfill				LXF-backfill				RelShare(1d)			
		dev	MaxW	AvgW	AvgX	dev	MaxW	AvgW	AvgX	dev	MaxW	AvgW	AvgX
70	7/03	-1040.8	49.0h	25.7h	54.1	-581.3	60.0h	17.4h	25.8	423.5	20.1h	8.9h	28.8
113	8/03	-1178.6	35.6h	19.2h	59.9	-351.6	28.7h	11.4h	15.9	-411.5	30.6h	12.4h	61.4
103	11/03	-355.1	33.9h	8.9h	141.7	-483.0	32.4h	2.9h	27.8	424.8	20.5h	1.1h	2.8
166	11/03	-393.0	32.1h	4.6h	6.2	-273.6	37.8h	3.2h	3.1	-103.4	13.9h	1.2h	2.3

(b) Job size of each user

User	#jobs	Total NT (hr)	N				T (hrs)				N×T (node-hrs)			
			Min	Avg	Med	Max	Min	Avg	Med	Max	Min	Avg	Med	Max
70	7	1144.4	7	53.0	52.0	104	0.1	3.1	0.2	12.0	0.8	163.5	22.8	624.0
113	13	1258.0	16	30.8	32.0	32	0.0	3.1	2.5	5.7	0.2	96.8	80.0	183.9
103	49	6233.6	1	21.0	16.0	48	0.0	2.8	0.2	12.0	0.0	127.2	0.9	576.0
*166	85	774.7	1	1.0	1.0	1	0.0	9.1	11.8	12.0	0.0	9.1	11.8	12.0

TABLE V
INFORMATION OF UNDER-SHARED AND OVER-SHARED USERS EACH MONTH

Month	# Users	Under-shared > 50 node hours (% of all users)			Over-share > 50 node hours (% of all users)		
		FCFS backfill	RelShare	LXF backfill	FCFS backfill	RelShare	LXF backfill
6/03	73	46.6%	34.2%	36.9%	6.8%	6.8%	6.8%
7/03	68	54.4%	41.2%	52.9%	10.3%	13.2%	5.8%
8/03	74	56.8%	56.8%	45.9%	9.5%	9.5%	10.8%
9/03	74	56.8%	48.6%	56.8%	6.8%	5.4%	1.3%
10/03	77	53.2%	45.5%	46.7%	9.1%	5.2%	7.7%
11/03	81	44.4%	33.3%	32.0%	6.2%	7.4%	4.9%
12/03	61	37.7%	37.7%	42.6%	8.2%	6.6%	4.9%
1/04	53	50.9%	45.3%	50.9%	11.3%	9.4%	7.5%
2/04	73	58.9%	42.5%	49.3%	6.8%	9.6%	4.1%
3/04	70	48.6%	40.0%	42.8%	7.1%	10.0%	4.2%

usage - cumulated entitled share $>$ 50). Typically, there are 70-80 active users each month. RelShare(1d) consistently has a similar or lower fraction of users incurring an under-share or an under-share of over 50 node hours each month. Only under 10% of users each month have an over-share. Note that the entitled share used to compute over-share or under-share is an idealized share, which can only be achieved under time-sharing systems. Thus, most users will be under-shared according to the idealized share measure. All policies have fairly comparable number of over-shared users in that one has more over-shared users in some months, but fewer in other months, than that of other policies. The results in Table V shows that RelShare(1d) does reduce the number of under-shared users.

Based on the results of the previous section and this section, The conclusion is that RelShare(1d) is more fair except for users who have large jobs and submit small jobs in between large jobs or users who have long jobs and submit short jobs in between long jobs. This is due to the delay in reserving resource for large jobs or long jobs under RelShare(1d).

VI. PERFORMANCE IMPACT OF FAIR SHARE WINDOWS

Next, the question: "how does the fair share window size affect the performance of RelShare?" is examined For this study, fw is varied in the range of 4 hours and 7 days, corresponding to the range of fw typically used.

Intuitively, a larger fair share window (fw) could degrade the performance of heavy-demand users, since a longer history of their resource usage will be used to adjust down the current priority of their jobs. As a result, a larger fw may benefit users with a lighter demand, including Type-S1 users #8 and #24, shown in Table II.

Table VI shows how the performance of individual users are affected by RelShare using different fw values: 4 hours, 12 hours, 1 day, 2 days, and 7 days. Table VII shows how dev of individual users are affected by RelShare using different fw values. The results for users who suffer or benefit under RelShare(1d) are shown in Tables II-IV. For convenience, FCFS-backfill and LXF-backfill are repeated here. These results represent the trend observed for other users.

As expected, the heavy-demand user (i.e., #71), who incurs poorer performance under RelShare(1d) than under both backfill policies, suffers even more performance degradation if a larger fw (2 or 7 days) is used. On the other hand, for user #24, who also incurs poorer performance under RelShare(1d) than under both backfill policies but does not have a particularly high demand during that month, using a fw of 2 or 7 days improves the maximum and average wait of the user by over 80%. For user #70, who have difficult jobs and spread their jobs out but does not have a particularly high demand during that month, has similar performance when using a fw of 2 or 7 days. For users with a light demand or a few jobs, a larger fw also tend to improve their wait time performance. This can be seen from the results for user #107 in the table. Conversely, using a smaller fw (4 or 12 hours) tends to improve the performance of heavy-demand users but

TABLE VI
IMPACT OF FAIR SHARE WINDOW ON RELSHARE (JULY 2003)

	User #71			User #24			User #70			User #107		
Policy	Max. Wait	Avg. Wait	Avg. Bounded Slowdown	Max. Wait	Avg. Wait	Avg. Bounded Slowdown	Max. Wait	Avg. Wait	Avg. Bounded Slowdown	Max. Wait	Avg. Wait	Avg. Bounded Slowdown
FCFS-backfill	105.1h	28.7h	145.6	114.8h	26.1h	459.5	49.0h	25.7h	54.1	78.1h	16.1h	82.9
LXF-backfill	143.5h	27.7h	45.8	30.7h	8.8h	122.4	60.0h	17.4h	25.8	76.7h	15.2h	23.3
RelShare(4h)	127.2h	22.3h	128.2	151.1h	45.5h	1079.0	44.6h	18.9h	28.7	#92.2h	15.4h	137.2
RelShare(12h)	159.1h	29.2h	25.7	159.4h	39.5h	1137.3	59.6h	20.8h	18.5	42.6h	13.5h	148.9
RelShare(1d)	211.2h	30.2h	87.9	#283.8h	73.3h	2033.2	*20.1h	8.9h	28.8	*23.5h	8.0h	133.2
RelShare(2d)	#310.2h	35.2h	184.6	*53.8h	14.6h	288.5	*22.0h	11.7h	20.3	37.4h	6.4h	50.7
RelShare(7d)	#451.1h	49.2h	89.6	*30.8h	5.6h	106.9	*15.3h	6.6h	25.8	*20.2h	3.4h	39.2

(# marks fw values that have particularly poor max wait, compared with other fw values.)
(* marks fw values that have particularly good max wait, compared with other fw values.)

TABLE VII
IMPACT OF fw ON DEV OF EACH USER (JULY 2003)

	FCFS	LXF	RelShare				
User	backfill	backfill	fw=4h	fw=12h	fw=1d	fw=2d	fw=7d
71	6585.3	3803.6	5700.6	5884.6	4582.9	3016.5	-1095.3
24	-1787.6	-894.3	-2748.2	-2686.7	-4592.9	-1570.5	- 385.6
70	-1040.8	-581.3	-435.0	-669.4	423.5	-5.7	369.5
107	-1224.5	-1803.6	-1505.5	-783.1	-262.9	-183.3	340.4

degrade the performance of light-demand users, compared with fw = 1 day. However, note that a user with a light monthly demand may become a heavy demand in a shorter period of time (say a day or a few hours). Thus, the effect of fw on this user changes as the fw changes.

To continue the discussions of the impact of fw, Figure 4 summarizes the overall performance of each month under RelShare policies with different fw values. Figure 4(a) plots the average bounded slowdown of fw = 4h, 12h, and 1 day. Figure 4(b) plots the average bounded slowdown of fw = 1, 2, and 7 days. Figure 4(c) plots the maximum wait in each month under RelShare with fw = 4 hours, 12 hours and 1 day. Figure 4(d) plots the maximum wait in each month under RelShare with fw = 1, 2, and 7 days.

As shown in Figure 4(a), for the range of fw studied, the value of fw does not impact the average bounded slowdown by much, except July 2003. They also have minimal impact on the average wait each month (not shown). However, different fw values have a significant impact on the maximum wait each month. Specifically, using a smaller fw of 4 hours or 12 hours improves the maximum wait time of using fw = 1 day for most months, since heavy-load users are improved. On the other hand, using a fw of 7 days significantly degrades the maximum wait time for most months, because heavy-load users are significantly degraded. A fw of 2 days seems to be a good compromise in that it can significantly improve type-S1 users who have a large under-share under RelShare(1d), and has a much smaller impact on the maximum wait each month than that of fw = 7 days. However, this requires further study which will be include in the future work.

In this section, the average wait and bounded slowdown is rather insensitive to the value of fw, but the maximum wait is very sensitive to fw for the range of fw studied. To configure the fair share policies, the system administrators need to understand that fair share policies do have a significant impact on individual users. Too small a fw could penalize light-demand users; too large a fw could starve heavy-demand users. The choice of fw would depend on the goals of the computer systems and the workloads.

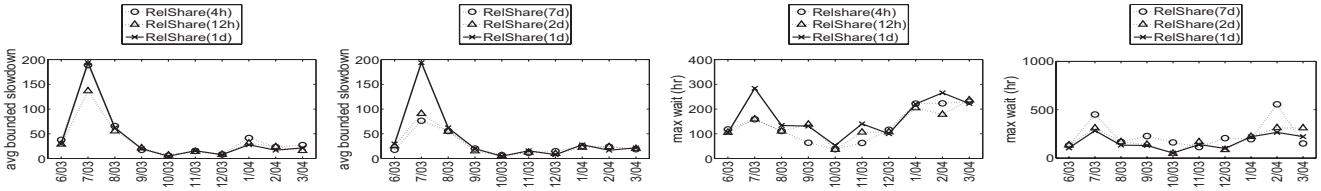
VII. SUMMARY

To better understand the impact and how to configure fair share used on production parallel job schedulers, the performance impact of fair share policies and its configurations are studied. Our main goal is to provide some insight into the performance implications of fair share policies, which may help system administrators configure their fair share policies.

First, similar to results in [2], our results show that fair share policies, studied in this research have little impact on the average performance measures. However, fair share policies have a significant impact on the maximum wait time and individual users, which were overlooked in the previous paper.

Second, the fair share window size, fw, has a significant impact on the performance, in that a large fw has the potential to starve heavy-demand users, but a small fw could penalize light-demand users.

Third, the analysis of fair share showed that fair share policies studied are more fair. That is, the heavy-demand users are not allowed to dominate the system resources, resulting in available resources for other users in the system. However, users who have large jobs but also submit small jobs in between large jobs or users who have long jobs but also submit short jobs in between long jobs may suffer poor performance. This is the main problem of fair share policies implemented by using the priority mechanism. That is, when a difficult job (i.e., a large or a long job) of the user queues behind an easy job (i.e., a small or a short job) of the same user; once the easy job of the user started, the fair share priority of the user is lowered, which may prevent the difficult job of the user from receiving a reservation, even if the job is the oldest one in the



(a) Avg. bounded X fw = 4h, 12h, 1d

(c) Avg. bounded X fw = 1d, 2d, 7d

(c) Max. wait fw = 4h, 12h, 1d

(d) Max. wait fw = 1d, 2d, 7d

Fig. 4. Impact of using different fair share window sizes (fw) for RelShare

queue. However, the users with mixture of jobs can reduce the impact of delaying a reservation by spreading their jobs.

REFERENCES

[1] J. KAY and P. LAUDER, "A fair share scheduler," *Communications of the ACM*, vol. 31, no. 3, pp. 44–55, 1988.
[2] S. Kleban and S. Clearwater, "Fair share on high performance computing system: What does fair really mean?" in *Int'l Symp. on Cluter Computing and the Grid (CCGRID)*, 2003.
[3] OpenPBS. [Online]. Available: http://www.nas.nasa.gov/Software/PBS/
[4] PBS pro. [Online]. Available: http://www.pbspro.com
[5] LSF fairshare documentation. [Online]. Available: http://accl.grc.nasa.gov/job_schedulers/lsf/Docs/lsf6.1/lsf6.1_admin/E_fairshare.html
[6] S. Kannan, M. Roberts, P. Mayes, D. Brelsford, and J. Skovira, "Workload management with loadleveler," IBM Redbook, Tech. Rep., 2001.
[7] Maui scheduler. [Online]. Available: http://www.supercluster.org/maui/
[8] MOAB scheduler. [Online]. Available: http://www.clusterresources.com/products/mwm/docs/MoabAdminGuide450.pdf
[9] S.-H. Chiang and S. Vasupongayya, "Design and potential performance of goal-oriented job scheduling policies for parallel computer workloads," *IEEE Trans. Parallel and Distributed Systems*, 2008.

Sangsuree Vasupongayya received a B.Eng degree from the Prince of Songkla University, Thailand, a M.S. degree from California State University, Chico, and a Ph.D. degree from Portland State University. Interested research areas include high-performance computer resource scheduling, cryptography and engineering education.