International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:6, No:5, 2012

Evaluating per-user Fairness of Goal-Oriented Parallel Computer Job Scheduling Policies

Sangsuree Vasupongayya

***Abstract*—**Fair share objective has been included into the goal-oriented parallel computer job scheduling policy recently. However, the previous work only presented the overall scheduling performance. Thus, the per-user performance of the policy is still lacking. In this work, the details of per-user fair share performance under the Tradeoff-fs(Tx:avgX) policy will be further evaluated. A basic fair share priority backfill policy namely RelShare(1d) is also studied. The performance of all policies is collected using an event-driven simulator with three real job traces as input. The experimental results show that the high demand users are usually benefited under most policies because their jobs are large or they have a lot of jobs. In the large job case, one job executed may result in over-share during that period. In the other case, the jobs may be backfilled for performances. However, the users with a mixture of jobs may suffer because if the smaller jobs are executing the priority of the remaining jobs from the same user will be lower. Further analysis does not show any significant impact of users with a lot of jobs or users with a large runtime approximation error.

***Keywords*—**deviation, fair share, discrepancy search, priority scheduling.

I. Introduction

FAIR share objective has been included in the goal-oriented parallel computer job scheduling policy called Tradeoff-fs(Tx:avgX), recently [1]. The impact of user request runtime which is known to be inaccurate [2] was presented in [3]. The policy was evaluated on several workloads with various characteristics in [4]. In all these works, the regular scheduling performances widely used in the field [5,6,7] such as average wait time, 99th-percentile wait time, maximum wait time and average slowdown were presented. In addition, the fair share measures namely dev (i.e., deviation) was proposed to measure the differences between the cumulated actual usage and the cumulated entitled share of each user over a given fair share window. However, the results presented in all previous works focused on overall performances. In this work, the details of per-user fair share performance under the Tradeoff-fs(Tx:avgX) policy will be further evaluated.

The remaining of this paper is organized as follows. Parallel computer job scheduling problems and the current solutions are described in Section 2. In Section 3, goal-oriented parallel computer jobs scheduling policies are reviewed. In Section IV, the experimental setting in this work including workloads, policies and performance measures are described. In Section V, the experimental results and discussions are presented. Finally, the conclusions are given in Section VI.

S. Vasupongayya is with the Department of Computer Engineering, Prince of Songkla University, Hat Yai, Songkhla, 90110, Thailand (e-mail: vsangsur@coe.psu.ac.th phone: 66-74-287360; fax: 66-74-212895).

II. Parallel Computer Job Scheduling

Parallel computer job scheduling problem is a problem of scheduling a job on a set of available computational nodes so that each job will eventually be executed or cancelled by its owner. Typically, the user must supply some job information such as the number of requested computational nodes and the estimated runtime of the jobs. The scheduler uses the job information to make a scheduling decision when any computational node becomes available. That is, when a job arrives at the system or when a job leaves the system. In this work, any of these two events will be called a 'scheduling decision point'.

To better explain the idea of the parallel computer job scheduling problem, Figure 1 simulates a scheduling decision point where the x-axis represented the time while the y-axis represented the number of computational nodes. At the current time (t_1), the system has two running jobs which are job no. 1 and job no. 2. And, there is one waiting job which is job no.3. If there is no arriving job during t_1 and t_2, the scheduler will be activated at t_2 because the job no.1 will be finished. Notice that, the job runtime information at the current time is not accurate because the job owner always gives an overestimated runtime. Thus, the job no.2 may be finished before job no.1 on the real system. Furthermore, the jobs can arrive at any time which is known as an on-line setting. That is, the scheduler will not know when the job will arrive making it difficult to make a working schedule offline. Therefore, this inaccurate information and on-line setting environment enhance the challenge of solving any parallel computer job scheduling problem.

Typically, the production parallel computer job scheduler is either a queue-based scheme [8,9,10] where each job is assigned to a waiting job queue by its characteristic or a job-based scheme [11,12,13], where all jobs are prioritized based on a weighted function of a set of pre-defined job measures. The difficulties of these schemes are the low-level parameter tuning process. For example for the job-based scheme, the system administrator must define what job measure to use in the weighted function in order to achieve a desired performance. The parameters sometime are not directly related to the scheduling performances. For example, to prevent a starvation problem the system administrator may use wait time of each job as the job measure to indicate the starvation problem. That is, the job should not wait too long otherwise the system may be in a starvation state. As a result, the system administrator must fine tune the parameters every time that something is changed such as a change in objectives, a change in workload characteristics, etc.

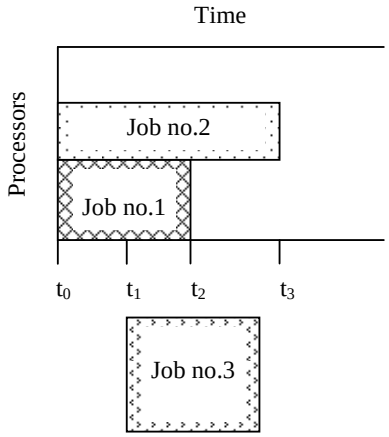

Fig. 1 Simplified parallel computer job scheduling problem

III. Goal-Oriented Parallel Computer Job Scheduling

To reduce the task of tuning the low-level scheduling parameters, the goal-oriented parallel computer job scheduling policy (Tradeoff(Tx:avgX) was proposed [14, 15]. The policy has been shown to achieve good scheduling performances under conflicting objectives. The policy also achieves good performances under both single partition systems and multiple partition systems [16]. Recently, the fair share objective was included in the goal-oriented parallel computer job scheduling policy [1]. The newly designed goal-oriented policy also achieves good scheduling performances and fair share performances under both accurate and inaccurate runtime and under various workload characteristics.

The main idea of the goal-oriented parallel computer job scheduling policy is to replace the task of tuning low-level scheduling parameters for performances using a search engine. By giving a set of objectives, the search engine traverses the space of solutions to find a 'good' solution within a certain time limit. Since the search space can be very large, the search engine must quickly find a 'good' solution and quickly discard a 'bad' one. To do so, a discrepancy based search technique, namely depth bounded discrepancy based search or DDS [17], is selected.

Under the goal-oriented parallel computer job scheduling policy, all waiting jobs at the current decision point are organized into a tree. Figure 2 shows a partial tree of five waiting jobs (i.e., job no. 1, job no. 2, job no. 3, job no. 4, and job no. 5) according to their arriving order. From depth 1 to the leaf node, only the left most branches are shown. At each level the jobs are ordered from left to right using a branching heuristic. In the figure, the jobs are ordered according to the first-come-first-serve or FCFS branching heuristic.

At each decision point, the tree of waiting jobs is organized. The search engine, then, walks on this tree to find a 'good' solution. The search engine starts from the first left node at the first left node at the first depth and find the best available time slot for the job to be scheduled. The first job is then assigned that time slot. The search engine follows the left most path until it reaches the left node.

That is, one complete solution is found. The score of this first solution is, then, calculated and saved as the best solution found so far. The performance impacts of various scoring modules are presented in [18]. The next path discovered will be according to the DDS algorithm which is the path with the discrepancy at the first depth. Once the next solution is found, its score will be calculated and compared with the best score so far. If the new solution is better, the new solution is kept as the best solution.

To illustrate the DDS algorithm, Figure 3 shows the order of paths discovered on a tree of three waiting jobs. The first path discovered is the heuristic path which is the left most path. This path follows the branching heuristic which is a reason to be called the 'heuristic path'. The next path (denoted no. 2 in the figure) is the left-most path of the first discrepancy node at the first depth. The next path (denoted no. 3 in the figure) is the next path of the next discrepancy node at the first depth. After the first three paths are discovered, the search moves to the discrepancy at the next level (i.e., depth 2). Path no. 4 is discovered next because it is the left-most path of the discrepancy paths at depth 2. Next, the search will discover path no.5 and path no.6.

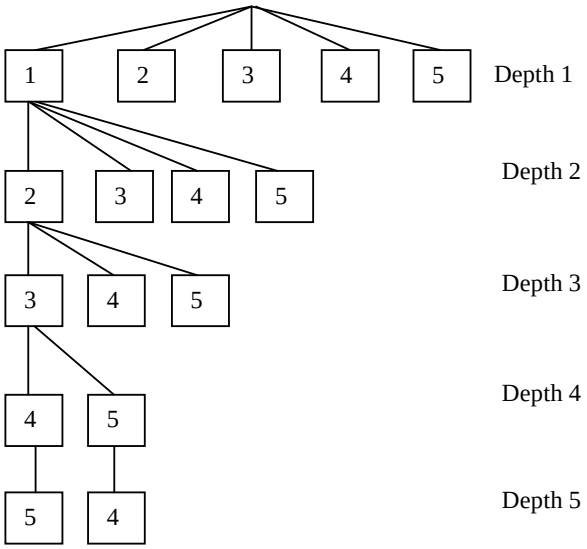

Fig. 2 A partial tree of five waiting jobs

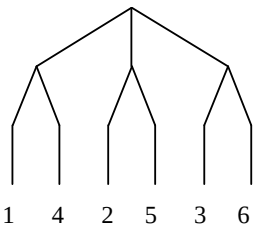

Fig. 3 the DDS order of paths discovered on a tree

As can be seen that the job on the left node will be discovered before the job on the right node, the order of jobs has a significant impact on the search engine.

Tradeoff-fs(Tx:avgX) policy includes fair share objective by ordering jobs according to the fair share measure. As a result, the policy can achieve both good scheduling performances and good fair share performances.

Furthermore, the very first job considered by the scheduler can have a significant impact on the remaining jobs because it will limit the available space on the system for the remaining jobs. Thus, discovering jobs according to DDS can help reduce this impact because the discrepancy at the first depth will be searched first before the discrepancy lower level of the tree. As a result, each job will have a chance to be the first job considered. If doing so can result in a better solution, that solution will be selected as the best solution found so far.

With the separation between the objectives and the search engine, the goal-oriented parallel computer job scheduling policy can reduce the administrator bundle of parameter tuning tasks for performances.

IV. Experimental setting and measures

The per-user fair share measures and scheduling performances of each user under the goal-oriented parallel computer job scheduling policy namely Tradeoff-fs(Tx:avgX) is evaluated in details. Another policy to be evaluated against the goal-oriented policy is the RelShare(1d) which will be described next. The results are collected from an event-driven simulator with real job traces from three production parallel computer centers as input. The workload information is presented in the next subsection.

To be realistic, a warm up and a cool down periods are included in the simulation. The simulation is done one month at a time with a one-week warm up (i.e., jobs from the previous month) and a cool down period (i.e., jobs from the next month continue to arrive). The experiments are conducted under both accurate and inaccurate runtime situations. The per-user fair share measures are evaluated against the accuracy of the user runtime approximation and the amount of resources requested by the user. These topics are still lacking in the previous work [1] because the previous work only focused on comparing the overall performances of the Tradeoff-fs(Tx:avgX) against other policies. Thus, the performance analysis aimed at demonstrating fairness of the scheduler on the users with high demand. These additional factors are evaluated, in this work, to further understand the strength and weakness of the fair share goal-oriented parallel computer job scheduling policies.

A. Workloads

There are three real job traces in this study. The first job trace is a ten-monthly workload 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. The second job trace is a ten-monthly KTH workload [19]. The last job trace is a twelve-monthly SDSC workload [19]. Table 1 shows the workload characteristic in each month of each job trace. The information presented includes total demand (Proc. demand), number of users (#users), number of jobs (#jobs), average job size (i.e., NT: nodes-hour, N: computational node, T: runtime in hours), and per-user information of average number of jobs and average demand in node-hours.

TABLE I
Information Of Each Job Trace

IA-64 workload						
				avg.	avg. per user	
Month	Proc. demand	#users	#jobs	job size (node-hours)	#jobs	demand
6/03	82%	73	2191	34.5	30.0	1034.7
7/03	89%	68	1400	60.6	20.6	1247.4
8/03	79%	73	3221	23.4	44.1	1031.6
9/03	72%	74	3057	21.7	41.3	895.5
10/03	71%	75	4149	16.3	55.3	899.5
11/03	73%	81	3443	19.5	42.5	827.1
12/03	74%	61	3521	20.1	57.7	1159.3
1/04	73%	53	3156	22.1	59.5	1313.6
2/04	74%	73	3969	16.6	54.4	900.3
3/04	75%	70	3466	20.6	49.5	1018.0
KTH workload						
				avg.	avg. per user	
Month	Proc. demand	#users	#jobs	job size (node-hours)	#jobs	demand
10/96	69%	68	2404	21.4	34.5	755.44
11/96	69%	66	1990	25.2	33.5	761.07
12/96	65%	69	2299	21.3	35.0	709.38
1/97	76%	65	2939	19.3	33.0	870.7
2/97	76%	75	2916	17.7	38.0	688.01
3/97	74%	69	2081	26.8	35.0	807.49
4/97	70%	83	2860	17.7	42.0	610.53
5/97	68%	80	4080	12.5	40.5	637.98
6/97	72%	58	2697	19.5	29.5	905.29
7/97	62%	59	2182	21.4	30.0	790.91
SDSC workload						
				avg.	avg. per user	
Month	Proc. demand	#users	#jobs	job size (node-hours)	#jobs	Demand
6/00	75%	120	7043	89.5	60.5	5251.6
7/00	72%	117	5607	110.9	59.0	5313.8
8/00	77%	141	5433	122.9	71.0	4736.1
9/00	60%	118	5172	97.8	59.5	4285.2
10/00	68%	119	4234	139.4	60.0	4959.4
11/00	70%	115	4132	141.5	58.0	5085.7
12/00	62%	107	3187	167.8	54.0	4998.9
1/01	62%	113	5963	90.5	57.0	4775.8
2/01	72%	128	6912	81.0	64.5	4372.2
3/01	70%	143	6206	97.5	72.0	4229.8
4/01	76%	132	7167	88.1	66.5	4783.4
5/01	83%	151	8428	81.8	76.0	4563.4

B. Policies

The Tradeoff-fs(Tx:avgX) will be evaluated against a basic priority backfill policy namely RelShare(1d) as proposed in [20]. RelShare(1d) policy considers jobs for scheduling according to the fair share measure of the job's owner. The backfill technique [21] is added to allow some out of priority order scheduling results. That is, some low priority jobs can be scheduled to execute on available nodes if its executions do not affect the executions or scheduled executions of higher priority jobs. The fair share priority is the ratio of the entitled share to the actual cumulated usage dynamically computed over a one-day window. One-day window is a typical fair share window value on several production schedulers [8,9,12,13]. Goal-oriented parallel computer job scheduling policies (Tradeoff-fs(Tw:avgX)) is described in Section III.

That is, Tradeoff-fs(Tw:avgX) is covered a fair share objective by using the fair share priority as the branching heuristic when organized the search tree. In addition to the fair share objective, the goal-oriented policies considered two objectives—preventing starvation and minimizing average measures.

C. Performance Measures

Both overall performances and per-user performances are considered in this study. The focused performance is the per-user fair share measure namely deviation or dev for short [20] which calculates the differences between the entitled share and the actual cumulated usage. Thus, the positive dev value means the over-share and the negative dev value means the under-share. The widely used scheduling performances such as the wait time and slowdown time will also be presented in some arguments. More overall scheduling performances and overall fair share performances were presented in previous works [1].

V. Results and Discussion

Table II shows the dev performance provided by each policy under both actual runtime information (T) and approximate runtime information (R) of the highest demand user of each month of the IA-64 workload. Table 3 shows the same information of the KTH workload while Table 4 shows that of the SDSC workload. Table 5-7 show the overall dev performances of the two policies on IA-64, KTH and SDSC workloads, respectively. Table 8-10 show the overall average wait time performance of the two policies on IA-64, KTH and SDSC workloads, respectively. Table 11-12 show the overall scheduling performances of the two policies on 06/03 IA-64 workload and 06/00 SDSC workload, respectively.

According to the results shown in Table 2, these users are mostly over-share under most cases. The very interesting points are the performances of the goal-oriented policy on 06/03, 09/03, 10/03, 01/04, 02/04 and 03/04 months. That is, the goal-oriented policy provides higher per-user dev performances than that produced by the RelShare policy. However, the overall absolute dev data shown in Table 5 shows that the goal-oriented policy provides lower total absolute dev performances in all months except 06/03 under inaccurate runtime information. According to the scheduling performance of all policies on 06/03 month, the goal-oriented policy achieves good scheduling performances. Thus, the goal-oriented policy trades dev performances for scheduling performances in this case. As a result, some jobs are receiving exceed resources because doing so will result in a better overall scheduling performance which is parts of the objective considered.

For the results of KTH workload, the high demand users do not always over-share under the goal-oriented policy. User no.3 on 10/96 month has an average job size at 244.4 node-hours and all 38 jobs of this user require 80 computational nodes which is the largest of this month. As a result, this user suffered under the basic priority backfill policy (i.e., RelShare(1d)).

However, the goal-oriented policy can reduce the under-share of this user. There are four months that the goal-oriented policy produces larger dev values than that produced by the RelShare(1d) policy. These months are 11/96, 02/97, 05/97 and 07/97. However, overall absolute dev performance of all policies of each month in Table 5 shows that the goal-oriented policy is fair because it reduces the total absolute deviation of all users.

TABLE II
Per-User Dev Performance Of The Highest Demand User Of Each Month Of IA-64 Workload

Month	User No.	RelShare(1d)		Tradeoff-fs(Tx:avgX)	
		T	R	T	R
06/03	49	1131	2951	5309	6605
07/03	3	6227	5920	2778	5447
08/03	3	4041	3795	3742	3464
09/03	3	1757	353	4762	3563
10/03	118	1101	543	1150	1198
11/03	3	2251	4390	1355	1924
12/03	103	3249	3349	967	1287
01/04	3	3459	-70	5007	1109
02/04	171	85	3427	-1150	-2441
03/04	42	1764	-187	2247	3185

TABLE III
Per-User Dev Performance Of The Highest Demand User Of Each Month Of KTH Workload

Month	User No.	RelShare(1d)		Tradeoff-fs(Tx:avgX)	
		T	R	T	R
10/96	3	-7730	-6899	-156	-93
11/96	6	-203	-554	-1607	-1281
12/96	6	145	-669	-11	394
01/97	15	1132	1614	230	444
02/97	84	4878	3996	4311	4858
03/97	84	4154	4125	354	329
04/97	14	-1062	-383	-411	-284
05/97	67	-274	-450	-515	-677
06/97	29	-1451	-1411	221	220
07/97	67	-8	-78	155	118

TABLE IV
Per-User Dev Performance Of The Highest Demand User Of Each Month Of SDSC Workload

Month	User No.	RelShare(1d)		Tradeoff-fs(Tx:avgX)	
		T	R	T	R
06/00	246	4737	-800	14515	-646
07/00	151	12241	7924	7989	3151
08/00	151	10309	10133	4510	7477
09/00	273	13851	-7017	15642	11092
10/00	95	-889	-7625	9937	10457
11/00	95	-1646	1364	24347	27580
12/00	95	-3748	1008	2666	8332
01/01	99	12975	12967	2630	-8372
02/01	174	-2040	20655	5554	16173
03/01	174	-11530	12333	3430	7035
04/01	101	-1227	-571	11828	11849
05/01	101	545	4983	-264	-1090

Another interesting point is user no.6 in 11/96 month which suffers under the goal-oriented policy while this user does not suffer as much under RelShare policy. This user has 75 medium size job (i.e., average job size is 114 node-hours). However, these jobs are a mixture of 2 to 64 nodes with an average runtime of 5.8 hours.

Thus, this user is an example of the goal-oriented policy weakness presented in the previous work [1]. That is, a user with a mixture of jobs may suffer because his/her small jobs can be backfilled resulting in a lower fair share priority. Thus, the larger jobs will be affected by this change because the jobs will be considered for scheduling after a lot of jobs (from other higher priority users).

Similar to the results of IA-64 workload, the high demand users of each month of the SDSC workload are mostly benefited under the goal-oriented policy, according to the data presented in Table 4. The overall absolute dev performances in Table 7 also shows that the goal-oriented policy produces lower total dev performances than that produced by RelShare(1d) on most months, except 06/00. According to the scheduling performances of each policy on 06/00 SDSC workload shown in Table 9, the goal-oriented policy trades the dev performances for the scheduling performances.

TABLE V
OVERALL ABSOLUTE DEV PERFORMANCE ON IA-64 WORKLOAD

	RelShare(1d)		Tradeoff-fs(Tx:avgX)	
Month	T	R	T	R
06/03	20813	28561	19832	29986
07/03	72789	85083	61553	72520
08/03	103526	123068	90127	106287
09/03	131350	152926	116515	136603
10/03	148315	174058	132585	153869
11/03	170053	201312	146209	170737
12/03	195094	223434	161359	190893
01/04	225696	250236	182672	210067
02/04	259492	294952	197859	231871
03/04	284678	331111	219018	261493

TABLE VI
OVERALL ABSOLUTE DEV PERFORMANCE ON KTH WORKLOAD

	RelShare(1d)		Tradeoff-fs(Tx:avgX)	
Month	T	R	T	R
10/96	23722	23876	11226	12456
11/96	45622	47507	26445	27733
12/96	63754	65695	39754	42806
01/97	82059	86580	52261	56912
02/97	102119	108646	68986	76068
03/97	125979	136087	85437	92557
04/97	139846	150409	94788	102476
05/97	148431	158745	103636	109483
06/97	155120	166703	109049	115794
07/97	160202	172427	112956	119368

TABLE VII
OVERALL DEV PERFORMANCE ON SDSC

	RelShare(1d)		Tradeoff-fs(Tx:avgX)	
Month	T	R	T	R
06/00	127867	122842	136207	139704
07/00	290361	289481	266049	289369
08/00	441725	479307	367728	343707
09/00	554176	581898	472440	478405
10/00	687519	717063	595267	606284
11/00	729824	773483	675065	713440
12/00	834837	883556	757964	819434
01/01	984392	1044091	845105	922062
02/01	1116066	1244936	959343	1061431
03/01	1229301	1416714	1051635	1174007
04/01	1436186	1648090	1198499	1315528
05/01	1674706	2026606	1273787	1466926

TABLE VIII
SCHEDULING PERFORMANCES ON 06/03 IA-64 WORKLOAD

	RelShare(1d)		Tradeoff-fs(Tx:avgX)	
Measure	T	R	T	R
Avg. Wait	4.7h	7.6h	4.1h	4.9h
Max. Wait	104.5h	163.4h	48.8h	61.5h
Avg. Slowdown	29.6	42.8	27.6	36.3

TABLE IX
SCHEDULEING PERFORMANCES ON 06/00 SDSC WORKLOAD

	RelShare(1d)		Tradeoff-fs(Tx:avgX)	
Measure	T	R	T	R
Avg. Wait	0.8h	0.9h	0.7h	0.8h
Max. Wait	94.2h	102.8h	38.9h	103.3h
Avg. Slowdown	7.9	13.1	6.9	12.9

In conclusion, the results in this section show that the per-user deviation of most high demand users is mostly over-share. Since these users are either have large jobs or have a lot of jobs, their jobs will eventually be scheduled or backfilled. However, the goal-oriented parallel computer job scheduling policies can provide the lowest total absolute deviation values on most months. For a few months with slightly higher absolute deviation values observed, the goal-oriented policy trades these fair share performances for scheduling performances.

Further analysis of the users with a lot of jobs and the users with a set of slightly large jobs found the followings. The number of jobs does not strongly affect the goal-oriented policy performances. The users with a set of slightly large jobs, however, can be suffered under the goal-oriented policy because their jobs may be difficult to be backfilled. In addition, if a few of their smaller jobs are scheduled then user's fair share priority will be reduced. Thus, the larger jobs will then be low priority jobs. As a result, these jobs may need to wait longer which may lead to an under-share performance. If the jobs are not too large to be backfilled, however, the job's owner may be benefited. This is because the jobs may be backfilled for performances under the goal-oriented policy. The study also shows that the accuracy of the users provided runtime information and the number of jobs the user has, do not produce any significant impact on the per-user performance of the goal-oriented policy.

VI. CONCLUSIONS

This study presents and analyzes the per-user performances of Tradeoff-fs(Tx:avgX) i.e., the extending goal-oriented parallel job scheduling policy to cover fair share objective by applying the fair share priority as a branching heuristic. Tradeoff-fs(Tx:avgX) and RelShare(1d) are evaluated using an event-driven simulator. Three real job traces are used as input to the simulator. The per-user performances of the high demand users on each month of each workload are studied in details. Furthermore, the most-submitted-job users and the users with a large error in their runtime information are also analyzed.

The experimental results show that these users are usually benefited (over-share) under most situations because their jobs are either large or they have a lot of jobs. In the large job case, one job executed may result in over-share during that period.

In the other case, the jobs may be backfilled for performances. Thus, only the users with a mixture of jobs may suffer because if the smaller jobs are executing the priority of the remaining jobs from the same user will be lower. Therefore, the users with mixture of jobs should separate their jobs over a period of time under the goal-oriented policy for a better performance according to their share. Further analysis does not show any significant difference or impact of users with a lot of jobs or users with a large runtime approximation error.

The results presented in this work are further confirmed that the Tradeoff-fs(Tw:avgX) policy does achieve good fair share performances and scheduling performances. Even though some high demand users without a mixture of large and small jobs may be benefited, it does not result in a worst overall scheduling or fair share performances.

REFERENCES

[1] S. Vasupongayya, "Achieving fair share objectives via goal-oriented parallel computer job scheduling policies", Proc. WASET ICCSE'09, Bangkok, Thailand, December 25-27, 2009.
[2] S.-H. Chiang, A. Arpaci-Dusseau and M. Vernon. "The impact of more accurate request runtimes on production job scheduling performance". In Lecture Notes in Computer Science (2537):103-127, 2002.
[3] S. Vasupongayya, "Impact of User Runtime Estimates on Achieving Fair Share Objectives", Proc. TISD, Nong Khai, Thailand, March 4-6, 2010.
[4] S. Vasupongayya, "Impact of Workloads on Fair Share Policies", Proc. ANSCSE14, Chiang Rai, Thailand, March 23-26, 2010.
[5] S.-H. Chiang and C. Fu. "Benefit of limited time-sharing in the presence of very large parallel jobs". In proceedings of the IEEE International Parallel and Distributed Processing Symposium, 2005.
[6] S.-H. Chiang and M. Vernon. "Production job scheduling for parallel shared memory systems". In proceeding of the IEEE International Parallel and Distributed Processing Symposium, 2001.
[7] D. Talby and D. Feitelson, "Supporting priorities and improving utilization of the IBM SP2 scheduler using slack-based backfilling". In proceeding of the International Parallel Processing Symposium, 1999.
[8] OpenPBS, http://www.nas.nasa.gov/Software/PBS/
[9] PBS pro, http://www.pbspro.com
[10] LSF, http://www.platform.com/product/ lsffamily.
[11] D. Jackson, Q. Snell & M. Clement. "Core algorithms of the MAUI scheduler". In proceeding of the Workshop on Job Scheduling Strategies for Parallel Processing, 2001.
[12] Maui scheduler, http://www.supercluster.org/maui
[13] Moab scheduler, http://www.clusterresources.com/products/mwm/docs/moabadminguide450.pdf
[14] S. Vasupongayya, S.-H Chiang and B. Massey, "Search-based job scheduling for parallel computer workloads", In proceeding of the IEEE Cluster, Boston, MA, 2005.
[15] S.-H. Chiang and S. Vasupongayya, "Design and potential performance of goal-oriented job scheduling policies for parallel computer workloads". In the IEEE Transaction on Parallel and Distributed Systems. 19(12):1642-1656, 2009.
[16] A. Prasitsupparote & S. Vasupongayya, "Impact of Multi-partition Systems on Goal-oriented Parallel Computer Job Scheduling Policies" JCSSE2010, Bangkok, Thailand, 2010.
[17] T. Walsh, "Depth-bounded discrepancy search" Proc. Of International joint conference in Artificial Intelligence, 1997.
[18] S. Vasupongayya and S.-H. Chiang. "Multi-objective models for scheduling jobs on parallel computer systems". In proceeding of IEEE Cluster, Barcelona, Spain, 2006.
[19] Parallel workload archieve, available at http://www.cs.huji.ac.il/labs/parallel/workload.
[20] S. Vasupongayya, "Impact of fair share and its configurations on parallel job scheduling algorithms". (to appear). In proceeding of the 2009 WASET International Conference on High Performance Computing, Venice, Italy, October 2009.
[21] D.Lifka, "The ANL/IBM SP Scheduling System", Proc. First Job Scheduling Strategies for Parallel Processing (JSSP'95), 1995.

Sangsuree Vasupongayya received a Bachelor of Engineering in Computer Engineering from Prince of Songkla University, a Master of Science in Computer Science from California State University Chico and a Ph.D. degree in Computer Science from Portland State University. Currently, Dr. Vasupongayya is an assistant professor and the associate department head for academic affairs at the computer engineering department, Faculty of Engineering, Prince of Songkla University. Interested research areas include resource scheduling, cryptography, E-Learning and engineering curriculum and education.