Impact of Multi-partition Systems on Goal-oriented Parallel Computer Job Scheduling Policies

Amonrat Prasitsupparote Sangsuree Vasupongayya*
Department of Computer Engineering
Prince of Songkla University
Hat Yai, Songkhla, 90110, Thailand
*e-mail: vsangsur@coe.psu.ac.th

Abstract

This work is focusing on applying goal-oriented parallel computer job scheduling policies on multi-partition parallel computer systems since several existing systems partition their system resources. Goal-oriented parallel computer job scheduling policies utilizes a combinatorial search technique to find the best schedule within a time limit. Two implementation ideas of multi-partition goal-oriented parallel computer job scheduling policies are proposed and evaluated using an event-driven simulator. The first implementation idea includes the partition selection question into the search space while the second idea selects a partition according to some heuristics resulting in a smaller search space. The experimental results show that the second idea with a best-fit partition selection heuristic performs well under both equal-size and non-equal-size two-partition systems. Furthermore, the proposed policy also outperforms the two basic priority backfill policies.

Keywords: Multi-partition, job scheduling, goal-oriented, backfill, best-fit

1. Introduction

Parallel computer systems have recently increased their impacts on many applications such as drug designs, weather predictions, disaster predictions, scientific research simulations and modeling. A typical production parallel computer system can simultaneously service many users. To run a parallel computer job, a user usually submits his/her job to the job scheduler along with his/her job requirements such as the required amount of memory, the required amount of processors and the job estimated runtime. Most production parallel computer systems are typically non-preemptive. Under such schemes, a job will not be interrupted once it is executed. The scheduler is activated upon each job arrival and departure to select a set of waiting jobs for executions according to a pre-defined scheduling policy. All jobs in the system will eventually be executed unless they are canceled by their owner.

Typical non-preemptive parallel computer job scheduling policies are priority based backfilling policies. This set of policies considers jobs for executions according to some priorities. The priority is usually a linear combination of some job measures defined by the system administrator. For example, the job wait time may be used to allow the older jobs to have a higher priority than the younger jobs. The job slowdown may be used to allow the shorter jobs to have a higher priority than the longer jobs. A linear combination is usually a weighted function.

In many situations, however, a set of required objectives can be conflicting with each other. To prevent starvation, for example, difficult jobs (i.e., large jobs, long jobs and large-and-long jobs) must have a high priority because these jobs are likely to be delayed. To minimize average wait time, on the other hand, small and short jobs must have a high priority because majority of jobs are in this group. As can be seen, these two objectives are conflicting with each other. Most priority based scheduling policies can achieve one or the other but not both.

Goal-oriented parallel computer job scheduling policies [1] were proposed recently to reduce the system administrator tasks of adjusting and tuning low-level scheduling parameters (e.g., a set of weighted values) for performances. The goal-oriented policy employs a discrepancy-based complete search technique [2] to find a 'good' solution in a limited time according to a given set of objectives. It has been shown that the goal-oriented parallel job scheduling policies are good at finding a schedule that compromises a set of conflicting objectives [3, 4].

In this work, the goal-oriented parallel computer job scheduling policies are further evaluated under multi-partition parallel computer systems which is still lacking in previous works. Under a single partition system, all processors in the system are available

to the user. Thus, a job can be as large as the entire system. And, the scheduler only question is whether this job should be scheduled now. Under a multi-partition system, however, the processors are partitioned and a job must fit into one partition. Thus, the job can only be as large as the capacity of one partition. Therefore, a multi-partition job scheduler must answer an additional question that is on which partition this job should be scheduled.

Since several parallel computer systems partition their system resources and the original goal-oriented parallel computer job scheduling polices have not been tested on such systems, this work is focusing on modifying the original goal-oriented parallel computer job scheduling policies to handle multi-partition parallel computer systems. Both an equal-size and a non-equal size partition system are evaluated.

The remaining of this paper is organized as follow. Section 2 describes both the original and the proposed goal-oriented policies. Section 3 explains the methodologies for evaluating the proposed policies. Section 4 gives results and discussions while conclusions are given in Section 5.

2. Goal-oriented parallel job scheduling

The original goal-oriented policies are described in Section 2.1 while the proposed multi-partition goal-oriented policies are described in Section 2.2.

2.1 Single-partition goal-oriented policies

The idea of a goal-oriented parallel computer job scheduling is to reduce the difficulty of operating a parallel computer job scheduler. Instead of tuning low-level scheduling parameters for performances, the goal-oriented parallel computer job scheduler allows the system administrator to define a set of high-level scheduling objectives such as preventing starvations and improving average wait times. Then, the scheduler automatically searches the space of all possible schedules to find a 'good' schedule according to the objectives within a time limit.

To achieve this, all candidate schedules at each scheduling decision point are organized into a search tree according to some branching heuristic (i.e., the order of the jobs from left to right). Figure 1 shows an example search tree of four waiting jobs (numbered 1 to 4 in their arriving order) using the FCFS (i.e., first-come-first-served) branching heuristic. In the tree, each node represents a job while each path from the root to a leaf (e.g., 1-2-3-4) represents an order. The scheduler evaluates jobs for scheduling in the given order of each path. At each node, the earliest time that enough resources will be available to execute the current job is computed (according to the resources allocated to currently executing jobs and the waiting jobs considered prior to the current job). Note that the order of jobs considered for scheduling is not necessarily the same as the order the jobs started.

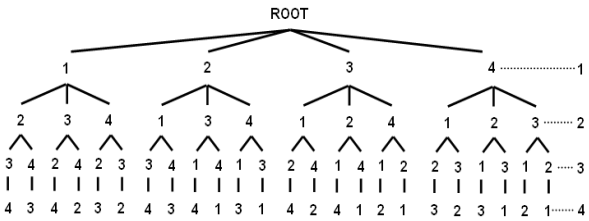


Figure 1. A search tree of four jobs

Once reaching a leaf node (i.e., search the whole path) the objective score is calculated and compared with the score of the best solution found so far. If the current solution is 'better' than the best solution found so far, the current solution is kept as the best solution. Otherwise, the current solution is discarded. The searching process repeats until the time limit is reached. The time limit can be described as the number of nodes visited or the actual search time. To find a good schedule, the search technique [2] employed by the goal-oriented policy is designed to search various different portions of the space of all possible schedules. Furthermore, allowing more search time will guarantee to always find an equally good or a better solution.

2.2 Multi-partition goal-oriented policies

Under a single partition system, the scheduler only needs to find out whether this job should be scheduled now. Under a multi-partition system, however, the scheduler must also answer on which partition this job should be scheduled. In this work, two implementation ideas of the multi-partition goal-oriented scheduling policy are proposed.

The first idea is to build a large search tree. Under this scheme, each node in the tree will have two pieces of information that is (1) which partition should this job be assigned and (2) when this job should start on the assigned partition. Figure 2 shows a partial search tree of four waiting jobs on a two-partition parallel computer system. The number of nodes in the tree increases linearly with the number of partitions. That is, the number of nodes on a two-partition system is twice as much as those on a single-partition system. For example, at the 1st-level there are 8 nodes (i.e., 1:1, 1:2, 2:1, 2:2, 3:1, 3:2, 4:1, 4:2) instead of 4 nodes for a 4-job search tree. At each node, the first number indicates the job identification number while the second number indicates the partition number that the job will be scheduled on. The doted lines on the right-hand side indicate the depth of the tree. At depth 1, all nodes (i.e., 8 nodes)

are shown. At depth 2-4, only the first subtree (i.e., the left-most subtree at each depth) is shown. As seen in the figure, the FCFS branching heuristic is used to order the jobs from left to right. And, the 1st partition always comes before the 2nd partition.

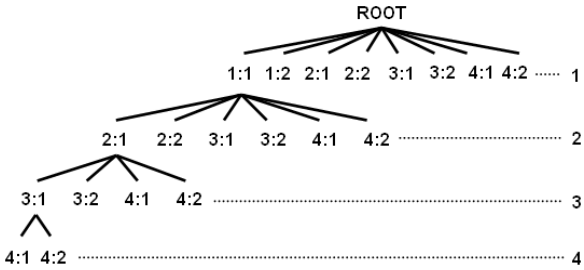


Figure 2. A partial tree on a two partition system

The second implementation idea aims to keep the search space small by first selecting the partition for each job using some heuristic. Thus, the search space is the same size as that of the original goal-oriented policies on a single partition system.

3. Experimental design

Two implementation ideas of the multi-partition goal-oriented parallel computer job scheduling policies discussed in Section 2.2 are evaluated using an event-driven simulator. The workload used in the previous works [1,3,4] is also used as input to the simulator. However, the workload is collected from a single partition system. Therefore, the workload is modified to be a two-partition system. The modification details are given in Section 3.1. The multi-partition goal-oriented policies are evaluated against two backfill policies whose details are given in Section 3.2. Section 3.3 provides performance measures and simulator settings.

3.1 Workload

Two multi-partition systems are tested in this work. One is an equal size partition each of which contains 64 computational nodes. Another one is a non-equal size partition i.e., the 1st partition contains 32 computational nodes while the 2nd partition contains 64 computation nodes. However, the original workload (i.e., IA-64) was running on a single partition of 128 dual-processor node system. Since each job has to be executed completely on one partition, it cannot request more than 64 computational nodes. Therefore, the original number of requested nodes of each job is modified to be the minimum between 64 and the original number of requested nodes.

Table 1 shows the characteristic of the modified workload including demand (Proc. demand), number of users (#users), number of jobs (#jobs) and job size (i.e., NT: node-hour) which is given in three measures (i.e., average (Avg.), median (Med.), and maximum (Max.)).

Table 1. Information of the modified IA-64 workload

Month	Proc. demand	#users	#jobs	Job size (NT)		
				Avg.	Med.	Max.
6/03	80%	73	2191	33.78	0.8	768.0
7/03	74%	68	1400	50.12	1.1	768.0
8/03	70%	73	3221	20.61	0.0	768.0
9/03	67%	74	3057	20.08	0.1	768.0
10/03	70%	75	4149	15.96	0.3	768.0
11/03	68%	81	3443	18.13	0.7	768.0
12/03	73%	61	3521	19.73	1.0	1151.0
1/04	70%	53	3156	21.11	5.1	1536.0
2/04	71%	73	3969	15.97	0.3	1536.0
3/04	72%	70	3466	19.77	0.0	1536.0

3.2 Policies

The most widely used non-preemptive parallel computer job scheduling policies are priority scheduling policies with backfilling techniques. Backfilling techniques allows lower priority jobs to start on idle resources as long as their executions do not delay the scheduled start time of the oldest waiting job [5]. In this work, the proposed multi-partition goal-oriented parallel computer job scheduling policies are evaluated against Best-fit-First-Come-First-Sever-backfill (i.e., FCFS-backfill) and Best-fit-Largest-Slowdown-First-backfill (i.e., LXF-backfill) policies.

The idea of the two backfill policies is similar to their single partition counterparts. The only difference is that each job will be assigned a partition using a best fit heuristic. That is, the waiting jobs are ordered using an appropriated priority value (either submit time for FCFS or slowdown for LXF). The best fit heuristic selects the partition which is best fit the considered job. That is, when the job can start on both partitions, the job will be assigned the partition with the least number of available nodes that is large enough to execute the job. For example, a two-partition system with 10 available computational nodes on the 1st partition and 8 available computational nodes on the 2nd partition. When a 5-node job arrives, it will be assigned to the 2nd partition because the job is best fitted in the 2nd partition.

Two versions of the multi-partition goal-oriented job scheduling policies (i.e., Tradeoff(Tw:avgX)) described in Section 2 are evaluated. The partition selection heuristic of the second implementation idea is best-fit similar to that of both backfill policies. To limit the search time during each scheduling decision, a number of node limit is used as a stopping criteria. In this study, the number of node limit is 4000 nodes which take only a few ten milliseconds.

3.3 Performance measures and simulator settings

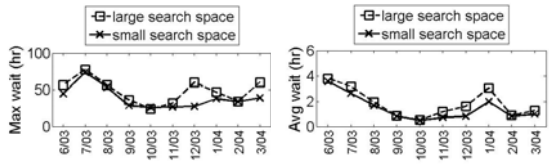
Several scheduling performance measures widely used in the field [6,7,8,9,10] are studied. The monthly performances are measured from the jobs submitted during the month. To be realistic, the simulator is loading with jobs submitting during the last week of the previous month and the jobs from the next month continue to arrive until the jobs from the measured month are all started. To study the full potential of the proposed policies, the scheduler uses actual job runtime information (i.e., accurate runtime information) to make scheduling decisions however the scheduler does not have the job arrival information ahead of time (i.e., online setting). The impact of user runtime estimates are left for future works.

4. Results and discussions

First, the scheduling performances of the two proposed multi-partition goal-oriented parallel computer job scheduling policies are presented in Section 4.1. Second, the performances of the multi-partition goal-oriented parallel computer job scheduling policies are compared against those of the two priority backfilling policies in Section 4.2. Section 4.3 shows the performances of all policies under a non-equal-size two-partition parallel computer system.

4.1 Impact of the search space size

Since the two multi-partition goal-oriented policy implementation ideas proposed in this work result in different size of the search space, this section investigates the impact of the search space size. Figure 3 presents the overall scheduling performances of the two ideas on an equal-size two-partition system each of which contains 64 computational nodes. Only the maximum and the average wait measures are presented because the average bounded slowdown performance is similar to that of the average wait.



(a) Maximum wait (b) Average wait

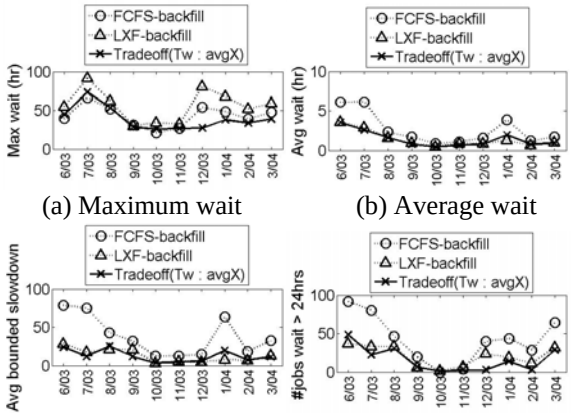
Figure 3 Performance of goal-oriented policies

According to Figure 3, the second implementation idea (i.e., small search space) outperforms the first idea on all measures on all months studied. Even though the performance differences on most months are small, the performance differences are clearly shown on 12/03 and 1/04 months. This is caused by the number of waiting jobs which is directly related to the size of the search space. The search space is double under the first idea causing the searching process to stop (i.e., exceeding the node limit) before reaching a good solution.

4.2 Goal-oriented versus priority backfill policies

In this section, the performances of the second idea of the multi-partition goal-oriented policy are compared with those of the two backfill policies. Figure 4(a)-(d) shows the maximum wait, the average wait, the average bounded slowdown and the number of jobs waiting longer than 24 hours under Best-fit-FCFS-backfill (denoted FCFS-backfill), Best-fit-LXF-backfill (denoted LXF-backfill) and the goal-oriented policy (denoted Tradeoff(Tw:avgX)) under an equal-size two-partition system.

As expected, the FCFS-backfill policy provides good maximum wait performances and poor average performances (i.e., both average wait and average bounded slowdown). The LXF-backfill policy has totally opposite performances. Thus, the results confirm that priority backfill policies still cannot achieve good performances on all three measures simultaneously on all months studied. The goal-oriented policy however can achieve the best or close to the best performances on all measures even when it is operated on an equal-size two-partition system.



(a) Maximum wait (b) Average wait

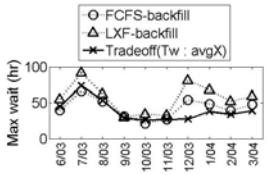
(c) Average bounded slowdown (d) Number of jobs wait longer than 24 hours

Figure 4 Performances of priority backfill versus those of the proposed goal-oriented policies

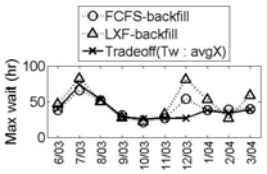
To analyze the detail performances of each policy, Figure 5 shows the performances of the jobs executed on each partition during the measured month. The jobs are assigned the partition according to the best-fit heuristic. Thus, each policy will result in different number of jobs. Table 2 shows the characteristics of jobs executed on each partition under each policy.

According to Figure 5, the performance trends on each partition are similar to the trends observed on

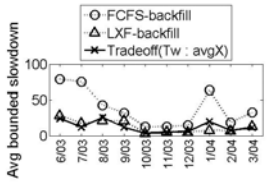
the overall graphs (Figure 4). That is, the multi-partition goal-oriented policy still produces good scheduling performances on each partition. Note that the average bounded slowdown on the 2nd partition can be higher than that of the overall graph because the values are calculated from a different set of jobs. That is, the value shows in Figure 5(d) is calculated from a subset of jobs used in Figure 4(b) because only the performances of jobs executed on the 2nd partition are used in Figure 5(d) while the performances of all jobs of the month are used in Figure 4(b).



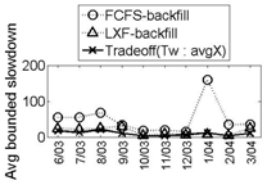
(a) Maximum wait 1st partition



(b) Maximum wait 2nd partition



(c) Average bounded slowdown 1st partition



(d) Average bounded slowdown 2nd partition

Figure 5 Performances of jobs on each partition under an equal two-partition system

Table 2. Information of jobs on each partition

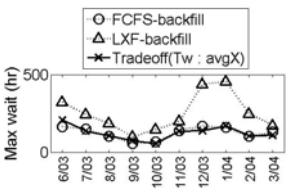
Number of jobs						
	FCFS-backfill		LXF-backfill		goal-oriented	
partition	1st	2nd	1st	2nd	1st	2nd
6/03	1268	923	1078	1113	1281	910
7/03	879	521	905	495	795	605
8/03	2034	1187	1677	1544	2199	1022
9/03	1921	1136	2143	914	1607	1450
10/03	2801	1348	3044	1105	1935	2214
11/03	2545	898	2409	1034	1635	1808
12/03	2441	1080	2405	1116	1581	1940
1/04	2241	915	2133	1023	1692	1464
2/04	2810	1159	2689	1280	1902	2067
3/04	2409	1057	2399	1067	1541	1925
Average job size (NT) in node-hour						
	FCFS-backfill		LXF-backfill		goal-oriented	
partition	1st	2nd	1st	2nd	1st	2nd
6/03	29.3	39.9	34.4	33.1	29.1	40.3
7/03	39.2	68.4	38.4	71.5	42.9	59.4
8/03	15.9	28.5	19.8	21.3	14.9	32.7
9/03	16.3	26.3	14.0	34.3	18.6	21.6
10/03	12.4	23.1	10.9	29.8	17.7	14.3
11/03	12.7	33.3	13.7	28.2	18.4	17.8
12/03	14.6	31.2	14.5	30.9	21.5	18.2
1/04	16.8	31.4	17.1	29.2	21.4	20.6
2/04	11.4	26.8	11.8	24.5	16.3	15.5
3/04	14.5	31.6	14.8	30.9	21.6	18.2

According to Table 2, the two backfill policies usually arrange more jobs in the 1st partition while the goal-oriented policy tends to evenly distribute jobs to both partitions. The average job size in node-hour information shows that the two backfill policies send large jobs to the 2nd partition while the small jobs are mostly executed on the 1st partition. The goal-oriented policy, however, sends similar jobs to both partitions. Since there are equal resources available on both partitions (each with 64 computational nodes), evenly distributing jobs to each partition reduces the chance of letting a few jobs wait for a long time resulting in good maximum wait time performances. The average wait time (Figure 4(b)) and the number of jobs waiting longer than 24 hour information (Figure 4(d)) demonstrate that the goal-oriented policy achieves good maximum wait performance and also provides good average performances.

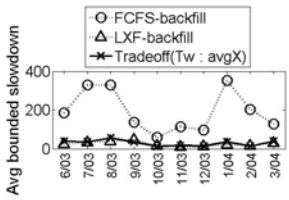
4.3 Impact of non-equal size partitions

Results reported in the previous two sections are on an equal-size two-partition system each of which provides 64 computational nodes. To study the impact of non-equal-size systems, all policies are evaluated under a two-partition system where the 1st partition contains 32 computational nodes and the 2nd partition contains 64 computational nodes.

Figure 6 shows the performances under a non-equal-size two-partition system of both backfill policies and the 2nd idea of the multi-partition goal-oriented policy. Figure 6(a) shows the maximum wait performance while Figure 6(b) shows the average bounded slowdown performances. The average wait performances (not show) have similar trends with those of the average bounded slowdown.



(a) Maximum wait



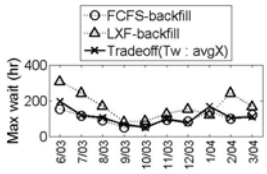
(b) Average bounded slowdown

Figure 6 Performances under a non-equal size system

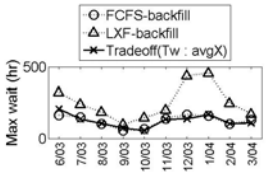
The results observed here are similar to those under an equal-size partition system. That is, the FCFS-backfill policy achieves good maximum wait performances on all months while the LXF-backfill policy achieves good average bounded slowdown performances on all months. The goal-oriented policy achieves the best or close to the best on both measures under a non-equal-size two-partition system.

Figure 7 shows the maximum wait and average

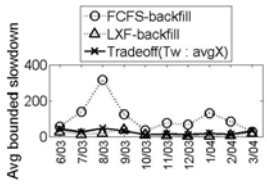
bounded slowdown performances of the jobs executed on each partition. The trends of the average bounded slowdown performances on both partitions are similar to that of the overall graph (see Figure 6(b)). The trend of the maximum wait performance of the jobs executed on the 2nd partition (i.e., the 64 computational node partition) is also similar to that of the overall graph (Figure 6(a)). However, the trend of the maximum wait performance of the 1st partition (i.e., the 32 computational node partition) is slightly different from that observed in the overall graph and in the 2nd partition, especially in January 2004 month. That is, the performances of all three policies on the 1st partition are similar on January 2004 month while the overall graph (Figure 6(a)) and the 2nd partition graph (Figure 7(b)) clearly show that the FCFS-backfill policy produces a significantly worse performance than the other two policies.



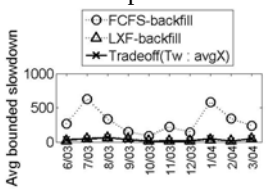
(a) Maximum wait 1st partition



(b) Maximum wait 2nd partition

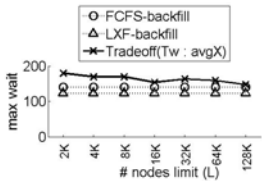


(c) Average bounded slowdown 1st partition

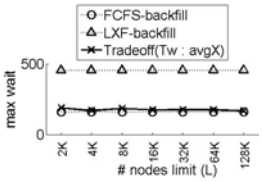


(d) Average bounded slowdown 2nd partition

Figure 7 Performances of jobs on each partition under a non-equal two-partition system.



(a) 1st partition



(b) 2nd partition

Figure 8 Maximum wait performances versus node limit on January 2004

Figure 8 shows the maximum wait performance of the goal-oriented policy on January 2004 workload on a non-equal-size two-partition system when varying the node limit from 2000 to 128000 nodes. The performances of both backfill policies are given for comparison purposes. Figure 8(a) clearly shows that the goal-oriented policy produces a smaller maximum wait value (i.e., a better solution) as the node limit increases (i.e., longer search time). While, Figure 8(b) shows that the better performance on the 1st partition does not degrade the performance of the 2nd partition.

5. Conclusions

Two implementation ideas of multi-partition goal-oriented parallel job scheduling policy are proposed in this work. The first implementation idea enlarges the search space by including the partition selection into the search tree. The second implementation idea keeps the same search space as that in the original goal-oriented policy by first assigning the partition according to some heuristic. The experimental results show that applying a simple best-fit partition selection heuristic to the original goal-oriented policy (i.e., the second implementation idea) produces the best or closed to the best performances in comparison with the two priority backfill policies. The results are observed on both equal-size and non-equal-size two-partition systems. Furthermore, allowing longer search time will result in an equally good or a better performance.

6. References

[1] S.-H. Chiang & S. Vasupongayya, "Design and potential performance of goal-oriented job scheduling policies for parallel computer workloads". IEEE Tran. on Parallel and Distributed Systems. 19(12):1642-1656, 2008.

[2] T. Walsh, "Depth-bounded discrepancy search", proc. of IJCAI, 1997.

[3] S. Vasupongayya, "Goal-oriented parallel job scheduling: A revisit", Proc. of the 2nd UBU-Research, Ubonratchathani, Thailand, July 2008.

[4] S. Vasupongayya, "Achieving fair share objectives via goal-oriented parallel computer job scheduling policies", Proc. WASET ICCSE'09, Bangkok, Thailand, December 25-27, 2009.

[5] D. Lifka. The ANL/IBM SP scheduling system. In Workshop on JSSPP, 1995.

[6] S.-H. Chiang, A. Arpaci-Dusseau & M. Vernon, "The impact of more accurate request runtimes on production job scheduling performance". In Lecture Notes in Computer Science (2537):103-127, 2002.

[7] S.-H. Chiang & C. Fu, "Benefit of limited time-sharing in the presence of very large parallel jobs". Proc. of IEEE IPDPS, 2005.

[8] S.-H. Chiang & M. Vernon, "Production job scheduling for parallel shared memory systems". Proc. of IEEE IPDPS, 2001.

[9] D. Talby & D. Feitelson, "Supporting priorities and improving utilization of the IBM SP2 scheduler using slack-based backfilling". Proc. of IPPS. 1999.

[10] D. Talby & D. Feitelson, "Improving and stabilizing parallel computer performance using adaptive backfilling". Proc. of IEEE IPDPS, 2005.