Adaptive Metric-Aware Job Scheduling for Production Supercomputers

Wei Tang, Dongxu Ren
Illinois Institute of Technology
Chicago IL, USA
{wtang6, dren1}@iit.edu

Narayan Desai
Argonne National Laboratory
Argonne IL, USA
desai@mcs.anl.gov

Zhiling Lan
Illinois Institute of Technology
Chicago IL, USA
lan@iit.edu

ABSTRACT

Job scheduling is a critical and complicated task on large-scale supercomputers where a scheduling policy is expected to fulfill aphormous and conflicting goals from both users and system owners. Moreover, the effectiveness of a scheduling policy is largely dependent on workload characteristic which varies from site to site and from time to time. Thus it is not easy to design a versatile scheduling policy being effective under all cirumstances. To address this issue, we propose an adaptive metric-aware job scheduling strategy. First, we propose metric-aware scheduling which enables the scheduler to balance the scheduling goals based on different interested metrics such as job waiting time, fairness, and system utilization. Second, we enhance the existing scheduler to be able to adaptively adjust scheduling policies based on the feedback of the metrics under real-time monitoring. We evaluate our strategy using real workload from supercomputing centers and the experimental results show our methods can effectively improve system performance.

Keywords

job scheduling, resource management, supercomputing, metric

1. INTRODUCTION

Job scheduling is a critical task for large-scale supercomputers. The effectiveness of a job scheduling policy directly influnces the satifaction of both users or system owners. For users, fast job turnaround and fairness are the major interested metrics, while for system owers, the system utilization is as important. Moreover, large production computing centers now face an increasing set of considerations in scheduling, such as avoiding failure interrupts and energy efficiency. All these considerations (or metrics) are correlated but sometimes conflicted. Even worse, the priorities differ from site to site and from time to time, which complicated the design of a versatile job scheduling policy.

HPDC '12 Delft, the Netherlands

Traditional scheduling policies can serve certain purposes but not balance them well. For example, using "first-come, first served (FCFS)" can achieve strict user fairness but not good for response time and also cause system fragmentation. On the other hand, using "short-job first (SJF)" can achieve best response time but violates job fairness and will likely cause job starvation. Some policy like "max expansion factor first" can act in the middle but also cannot balance the wanted metrics at will.

Meanwhile, the user benifit and system cost are not considered as a whole. Typically, in decision making, job prioritizing and resource allocation are separated into two subsequent phrases. This causes the latter has much fewer options to achieve more reasonable decisions. For example, when a first-priority job has no enough nodes to run, it reserve some resource while draining some idle resources, which may be used by the second job being scheduled instead. This kind of resource draining causes external fragmentations. Backfilling can reduce fragmentation but it is only a mitigation action after those fragmention already generated by job prioritizing [22].

Further, scheduling policy heavily depends on the workload characteristic. Eventhough we can identify a policy to achieve our integrated interests well, a different kind of workload may invalid it entirely. Although event-driven similutors can help to test a policy over a workload [23], it cannot enable the scheduler adapt scheduling policies to the changing situation efficiently.

Motivated by these issues, we propose an adaptive metric-aware job scheduling mechanism. Our solutions are two folded. First, we propose a metric-aware job scheduling mechanism which schedules and allocates jobs in an integrated fashion. Specifically, each job's schedule (when and where to run) is correlated with the impact it could make to the metrics we are interested. And the final decision is made to balance different metrics, representing both user benifits and system costs.

Second, we propose a mechanism to adaptively tune scheduling policies based on workload characteristic change. By monintoring the metrics values in real time, the scheduler can tune its scheduling policy to favor the metrics that are less satisfied recently, thereby mitigating the impact of workload characteristic on the scheduling policy. For example, if in the latest period of time the system utilization rate is reported below a certain threshold (or a longer-term average), the metric-aware scheduling policy will be tuned to favor the metric system utilization more than others. The second mechanism is built on the basis of the first one.

We implemented our schedule mechanism in a production resource management system named Cobalt [1]. We evaluated our mechanism using recent real job traces from production Blue Gene/P system at Argonne National Laboratory as well as archived job traces published on Parallel Workload Archive [2]. The experimental results show that our scheduling mechanism is superior in balance user benifit and system costs.Particully the targeted metrics can be achieved by tuning scheduling policy in advance. Also the adaptive policy tuning is demonstrated effective to adapt to varying workload characteristic.

The remainder of this paper is organized as follows. Section 2 further discusses our motivation and methodology. Section 3 describes the detailed design of the metric-aware job scheduling mechansim. Section 4 describes the detailed design of adaptive policy tuning. Section 5 illustrates our experimental results Section 6 discusses some related work. Section 7 summarizes the paper with some discussion on future work.

2. MOTIVATION AND METHODOLOGY

In job scheduling context, the effectiveness of a job scheduling policy are measured by certain metrics. A metric represents certain resource management goals or interests. Numerals scheduling research has been discussed various scheduling metrics. Generally, based on the category of the interests origin, the metrics can be grouped into two categories: user-centric metrics, reflecting levels of user satisfaction, and system-centric metrics, measuring the system utilization and costs.

Regarding user satisfaction, one of the most commonly used metrics for scheduling is the average response time (the time from when a job was submitted until it terminated) and waiting time(the time from a job's submission to its start). An alternative is the slowdown (the response time normalized by the job's actual running time). Meanwhile, user utility function is also used to associate the user satisfaction with a function over the response time [8][13]. While above metrics are considered reflecting the "efficiency" (meaning few delays and fast turnround), "fairness" is also a major metric. Likely being the reason why queues were formed in the first place, "fairness" may be cared about more by users than actual delays [15].

From system perspective, the system utilization and cost are equally important with user satisfaction. The supercomputers are expensive resources invested to serve applications and it is not desirable to let them idle often. Thus, the system utilization rate (the ratio of busy node-hours to the total node-hours over the examined time period) is a key metric that system owners care about. Because of the scheduling algorithm and workload characteristic, fragmentation is inevitable. How to reduce fragmentation is big challenge to improve system utilization. Loss of capacity (the ratio the idle node-hours while jobs are queuing compared to total capacity) [27][24] is a related metric that can be aimed to improve. Meanwhile, increasing concerns such as power consumption and fault interruptions have related metrics to measure. For example, delivered node-hours per watt can measure the energy efficiency.

All these metrics seems individual but are correlated, or sometimes conflicted. However, exsting scheme mechanism has some intrinsic limitations to balance these considerations as a whole. First, the job prioritizing and resource allocation are separated into phrases. That is, the latter have limited choices when allocate jobs. Since job prioritizing mainly focus on user experience and the job location influience system utilization and coly balance these considerations.

Even within job prioritizing, different goals are not balanced well. For example, using "first-come, first served (FCFS)" can achieve good user fairness but not good for average response time. On the other hand, using "short-job first" can achieve best response time but violates job fairness.

A mechanism is needed to explicitly balance these considerations. Moreover, this balance is not universal; different systems (or same system during different time) have varied priorities that result in some considerations prevailing over others. Therfore we propose an adaptive metric-aware job scheduling mechanism. The goal of this mechanism is two-fold. First, it shall provide a mechanism to connect job priorities explicitly to the impact (in metric space). Second, it shall be able to tune scheduling policies based on the feedback of monitored metrics.

The diagram of our design is shown in Figure 4, where there are three major components: a metrics balancer, a scheduling algorithm, and a metrics moinitor. The metrics balancer is used to balance different priorities or scheduling goals in composing an integrated scheduling algorithm. By running the scheduling algorithm, relevant metric values will be monitored. The feedback from metrics monitor will be sent back to metrics balancer so that it can adjust the weight of different factors to maintain the desired balance. This feedback process can be conducted manually or automatically. We call the process how the metrics balancer impacts the scheduling algorithm as "metric-aware job scheduling", and the process how the metrics mointor influences the adjustment to metrics balancer as "dynamic policy tuning". In following two sections we will present the detailed design of the metric-aware scheduling and dynamic policy tuning, respectively.

3. METRIC-AWARE JOB SCHEDULING

In this section we will present the detailed design of metric-aware job scheduling. The goal of metric-aware job scheduling is to provide a metric-balancer which can determine a scheduling algorithm that balances the interested metrics. In our current design, the metrics to be balanced are of three aspect. One reflect system utilization and other two reflects user satfication: one is "fairness", the other is "efficiency" (i.e. fast turnaround).

As mentioned earlier, the user-centric metrics are mainly influenced by job queuing policies and the system metrics are correlated with job allocation. Existing scheduling scheme allocate jobs one by one in their priority order. The problem is that when allocate one job at one time, it can achieve the best allocation for that single job but neglects the potential influience of other jobs in the queue (one simple exampe is shown in Figure 2). Thus we propose to schedule and allocate a group of jobs at one time so that the job allocation can result in more optimal system utilization. We call the number of jobs that are scheduled at one time as "window size", which varies from one to up to the number of queued jobs. Intuitively, the larger the window size is, more reasonable job allocation can be achieved but may violate user-centric metrics especially for the fairness.

To balance fairness and efficiency, we sort the queue using priority score factering both waiting time and running

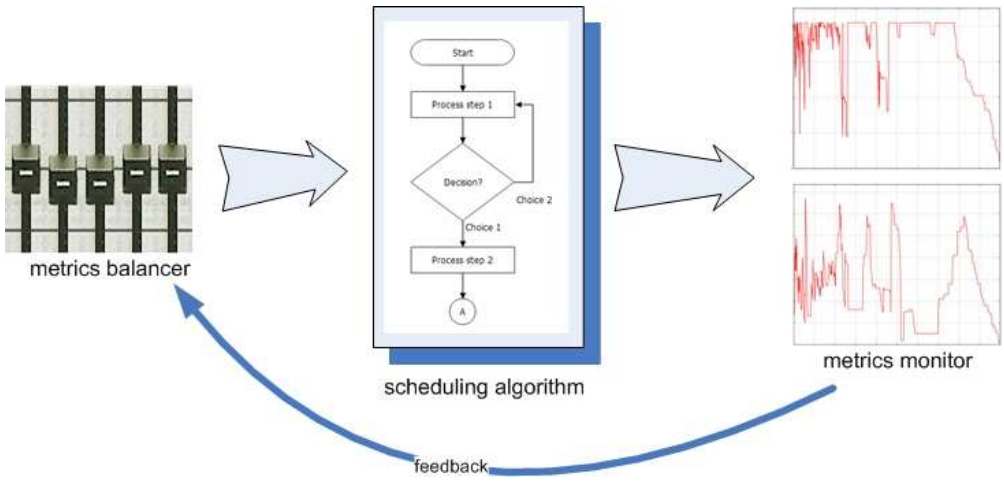


Figure 1: Diagram of adaptive metric-aware job scheduling framework.

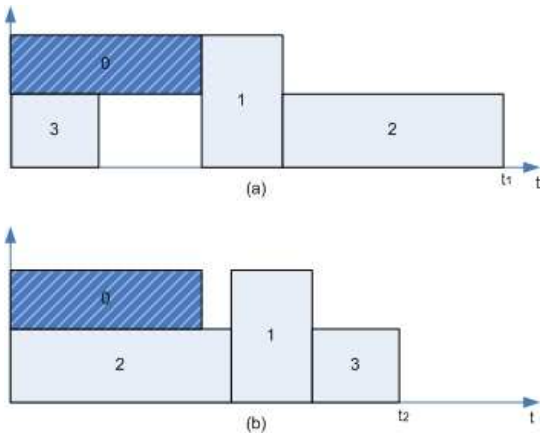


Figure 2: An example showing the limitation of scheduling and allocate jobs one by one. Job 0 is running, Jobs 1, 2, and 3 are waiting . (a) schedule and allocate job one by one in priority order; (b) schedule and allocate in a group as a whole. Apparently (b) achieves better system utilization.

time. By tune the weight of these two parameter, we can balance the priority on fairness and efficiency. The detailed scheduling steps are described as follows.

Step 1: Calcaulate job i's score regarding waiting time, maping the values to [0, 100]:

$$S_w = 100 \times \frac{wait_{max}}{wait_i}, \tag{1}$$

where $wait_{max}$ is the maxium waiting time of the current queue and $wait_i$ is the current waiting time of job i.

Step 2: Calcaulate job i's score regarding requested walltime, maping the values to [0, 100],

$$S_r = 100 \times \frac{walltime_{max} - walltime_i}{walltime_{max} - walltime_{min}}, \tag{2}$$

where $walltime_{max}$, $walltime_{min}$ are the maxium and minimum walltime times of the current queue, respectively, and $walltime_i$ is the walltime of job i.

Step 3: Calcaulate job i's balanced priority:

$$S_p = \alpha \times S_w + (1 - \alpha) \times S_r, \tag{3}$$

where α is a balance factor, varying from 0 to 1. A value closer to 0 means favoring fairness more; that closer to 1 means favoring efficiency more. This value is preset can be adjusted dynamically.

Step 4: Sort the jobs with S_p.

Step 5: Group jobs with window size β, for each job window, do job allocation. The job allocation algorithm with window size β runs as follows: based on the permutation of the jobs, do greedy job allocation, and select one schedule with the least makespan, meaning that the jobs in the window generate a schedule with highest utilization rate.

4. ADAPTIVE POLICY TUNING

The metric-aware scheduling allows system owners to tune the scheduling policy leaning toward their interested metrics. However, the effectiveness of a scheduling policy is also dependent on the workload characteristic which varies from time to time. To mitigate the impact of workload change, we introduce adaptive policy tuning as a compliment to metric-aware scheduling. Basically, we enable the scheduler dynamically change scheduling policy (by tuning the balance factor and allocation window) based on the feedback of the monitored metrics. For example, if the monitored average waiting time is below the expected value, the balance factor will be adjusted to a value to enhance the waiting time. This may impact the fairness metric. But when the fairness metric is found below a threshold, the balance factor will adjusted again to favor fairness more.

5. EXPERIMENT

In this section, we present a series of experiments to evaluate our proposed scheduling mechanism. We begin with introduction of the experiment, followed by discussion on evaluation metrics and then experimental results.

5.1 Experiment setup

5.2 Evaluation metrics

5.3 Experimental results

5.3.1 *Effect of metric-aware scheduling*

5.3.2 *More insight from metrics monitoring*

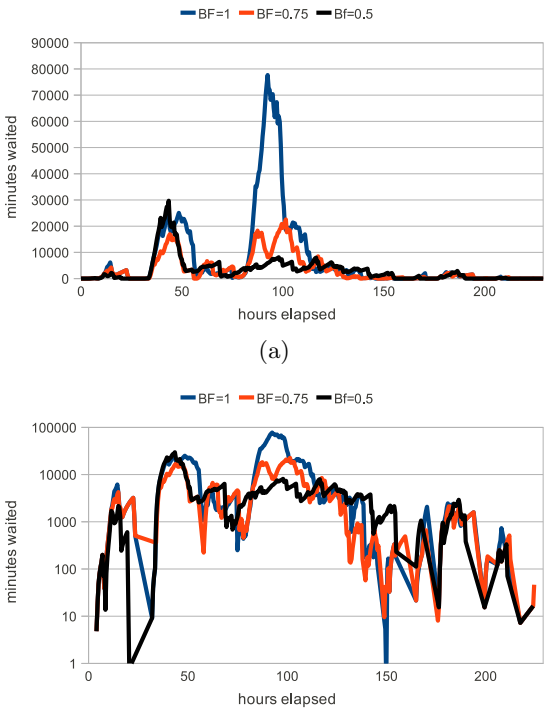


Figure 5: Monitoring of instant aggregate waiting times

5.3.3 *Effect of adaptive policy tuning*

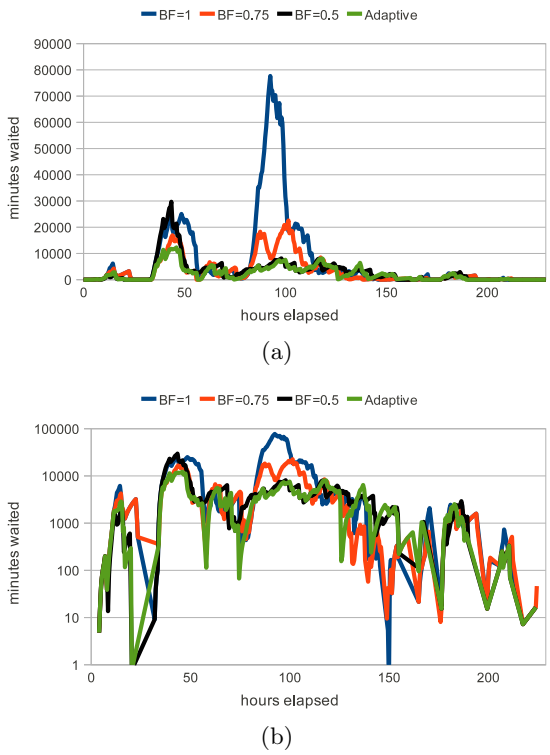


Figure 7: Results of adaptively tuning balance factor.

6. RELATED WORK

7. SUMMARY AND FUTURE WORK

8. ACKNOWLEDGMENTS

This work is supported in part by National Science Foundation grants CNS-0720549, and CCF-0702737. The work at Argonne National Laboratory is supported by DOE Contract DE-AC02-06CH11357.

9. REFERENCES

[1] Cobalt project. *http://trac.mcs.anl.gov/projects/cobalt.*

[2] Parallel workload archieve. *http://www.cs.huji.ac.il/labs/parallel/workload.*

[3] I. Ababneh and S. Bani-Mohammad. A new window-based job scheduling scheme for 2D mesh multicomputers. *Simulation Modelling Practice and Theory*, 19(1):482–493, 2011.

[4] N. Desai, R. Bradshaw, C. Lueninghoener, A. Cherry, S. Coghlan, and W. Scullin. Petascale system management experiences. In *Proc. USENIX Large Installation System Administration Conference (LISA)*, 2008.

[5] D. Feitelson. Experimental analysis of the root causes of performance evaluation results: a backfilling case study. *IEEE Transactions on Parallel and Distributed Systems*, 16(2):175–182, 2005.

[6] S. Grothklags and A. Streit. On the comparison of cplex-computed job schedules with the self-tuning dynp job scheduler. In *Proc. of IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2004.

[7] S. M. Hussain Shah, K. Qureshi, and H. Rasheed. Optimal job packing, a backfill scheduling optimization for a cluster of workstations. *J. Supercomput.*, 54(3):381–399, 2010.

[8] D. Irwin, L. Grit, and J. Chase. Balancing risk and reward in a market-based task service. In *Proc. of IEEE International Symposium on High performance Distributed Computing (HPDC)*, 2004.

[9] D. Jackson, Q. Snell, and M. Clement. Core algorithms of the Maui scheduler. In *Job Scheduling Strategies for Parallel Processing, LNCS 2221*, pages 87–102, 2002.

[10] J. P. Jones and B. Nitzberg. Scheduling for parallel supercomputing: A historical perspective of achievable utilization. In *Proc. of the Job Scheduling Strategies for Parallel Processing (JSSPP)*, 1999.

[11] N. Kaushik, S. Figueira, and S. Chiappari. Flexible time-windows for advance reservation scheduling. In

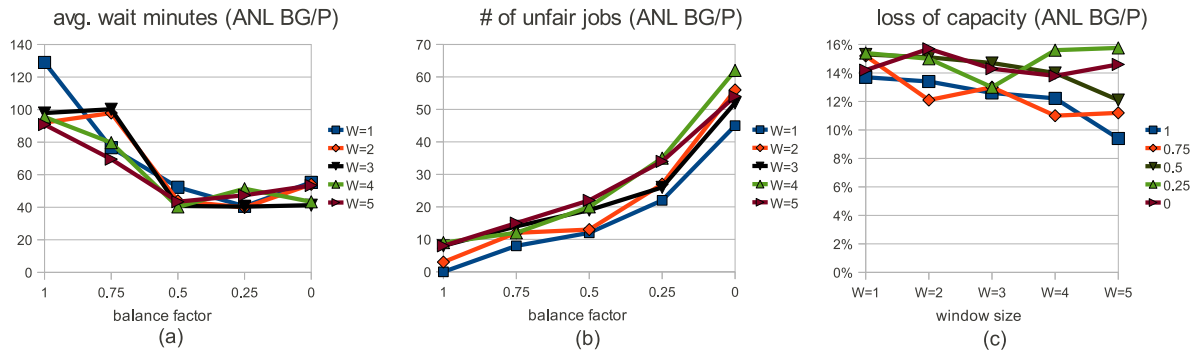


Figure 3: The effect of using balance factor and window size (ANL BG/P workload).

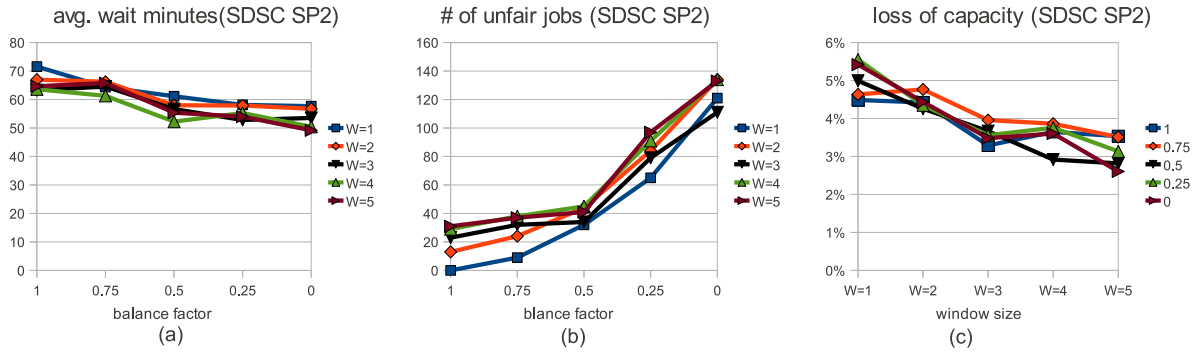


Figure 4: The effect of using balance factor and window size (SDSC SP2 workload).

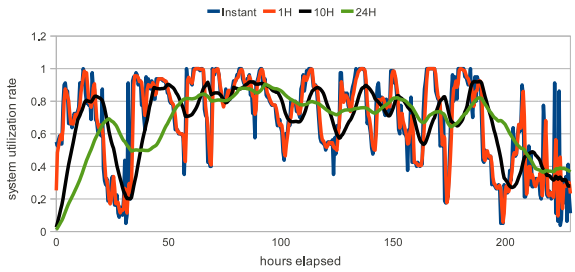


Figure 9: The effect on utilization rate of using 2D adaptive policy adjusting.

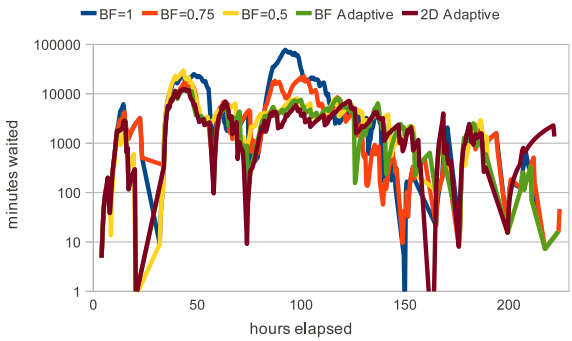


Figure 10: The effect on aggregate waiting of using 2D adaptive policy adjusting.

Proc. of IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS), 2006.

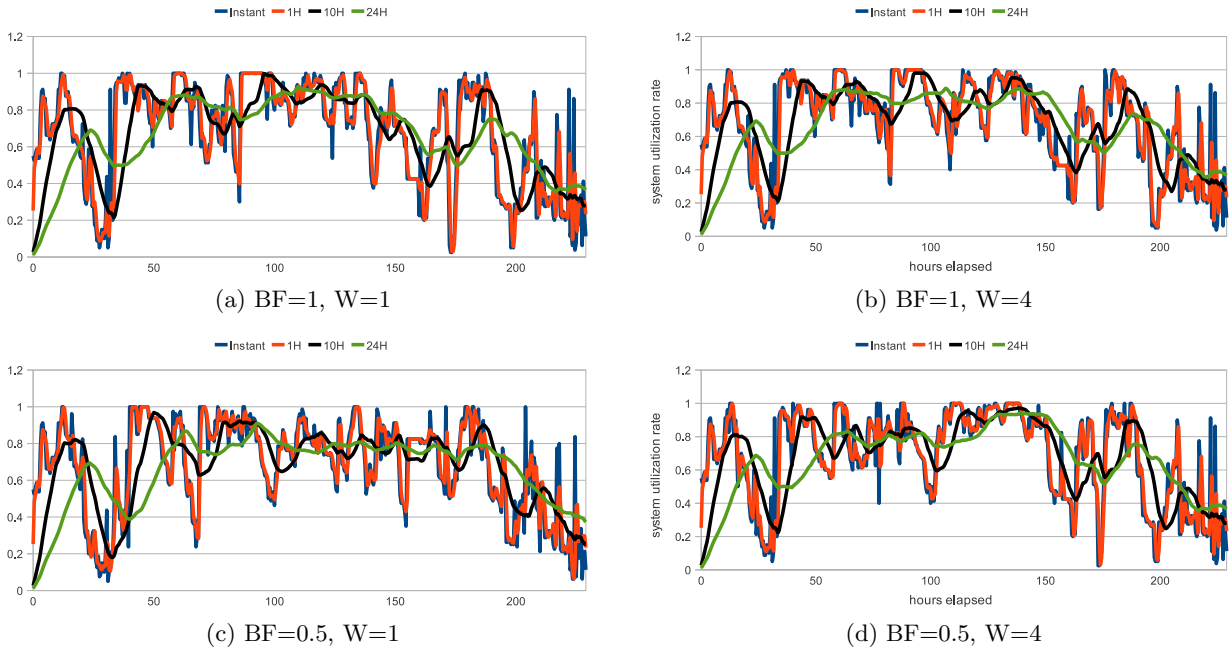
[12] D. Krishnamurthy, M. Alemzadeh, and M. Moussavi. Towards automated HPC scheduler configuration tuning. *Concurrency and Computation: Practice and Experience*, 23(15):1723–1748, 2011.

[13] C. B. Lee and A. E. Snavely. Precise and realistic utility functions for user-centric performance analysis of schedulers. In *Proc. of International Symposium on High Performance Distributed Computing (HPDC)*, 2007.

[14] C. Lu, J. A. Stankovic, G. Tao, and S. H. Son. Design and evaluation of a feedback control EDF scheduling algorithm. In *Proc. of IEEE Real-Time Systems Symposium (RTSS)*, 1999.

[15] D. Raz, H. Levy, and B. Avi-Itzhak. A resource-allocation queueing fairness measure. In *Proc. of the joint international conference on Measurement and modeling of computer systems (SIGMETRICS)*, 2004.

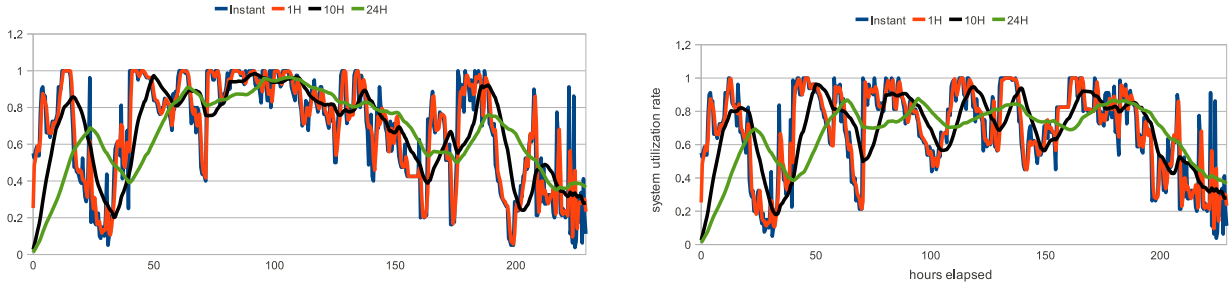
[16] G. Sabin, G. Kochhar, and P. Sadayappan. Job fairness in non-preemptive job scheduling. In *Proc. of International Conference on Parallel Processing (ICPP)*, 2004.



(a) BF=1, W=1 (b) BF=1, W=4

(c) BF=0.5, W=1 (d) BF=0.5, W=4

Figure 6: Monitoring of instant aggregate waiting times



(a) BF=1 (b) BF=0.5

Figure 8: Monitoring of instant aggregate waiting times

[17] E. Shmueli and D. G. Feitelson. Backfilling with lookahead to optimize the packing of parallel jobs. *J. Parallel Distrib. Comput.*, 65(9):1090–1107, 2005.

[18] S. Srinivasan, R. Kettimuthu, V. Subramani, and P. Sadayappan. Selective reservation strategies for backfill job scheduling. In *Job Scheduling Strategies for Parallel Processing, LNCS 2357*, pages 55–71, 2002.

[19] A. Streit. A self-tuning job scheduler family with dynamic policy switching. In *Job Scheduling Strategies for Parallel Processing, LNCS 2357*, pages 1–23, 2002.

[20] A. Streit. Evaluation of an unfair decider mechanism for the self-tuning dynp job scheduler. In *Proc. of IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2004.

[21] A. Streit. The self-tuning dynP job-scheduler. In *8th International Heterogenous Computing Workshop (HCW)*, 2004.

[22] W. Tang, N. Desai, D. Buettner, and Z. Lan. Analyzing and adjusting user runtime estimates to improve job scheduling on the Blue Gene/P. In *Proceedings of IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2010.

[23] W. Tang, Z. Lan, N. Desai, and D. Buettner. Fault-aware, utility-based job scheduling on Blue Gene/P systems. In *IEEE International Conference on Cluster Computing*, 2009.

[24] W. Tang, Z. Lan, N. Desai, D. Buettner, and Y. Yu. Reducing fragmentation on torus-connected supercomputers. In *Proc. of IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2011.

[25] W. A. Ward, C. L. Mahood, and J. E. West. Scheduling jobs on parallel systems using a relaxed backfill strategy. In *Job Scheduling Strategies for Parallel Processing, LNCS 2357*, pages 88–102, 2002.

[26] Y. Yuan, G. Yang, Y. Wu, and W. Zheng. PV-EASY: a strict fairness guaranteed and prediction enabled scheduler in parallel job scheduling. In *Proc. of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC)*, 2010.

[27] Y. Zhang, H. Franke, J. Moreira, and

A. Sivasubramaniam. Improving parallel job scheduling by combining gang scheduling and backfilling techniques. In *Proc. of IEEE International Parallel & Distributed Processing Symposium (IPDPS)*, 2000.