

Power-aware Resource Allocation via Online Simulation with Multiple-queue Backfilling*

Barry Lawson
Department of Math and Computer Science
University of Richmond
Richmond, VA 23173, USA
blawson@richmond.edu

C. Yue, E. Smirni, D. Nikolopoulos
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185, USA
{cyue, esmirni, dsn}@cs.wm.edu

Abstract

Although traditional scheduling policies for high-end parallel systems focus on minimizing average job wait time while maximizing system utilization, actual supercomputer workload traces confirm the existence of significant periods of time of low utilization. Previous work has shown that, in the context of backfilling schedulers, portions of such high-end systems can be selectively powered down to reduce power consumption and cooling costs while maintaining reasonable job performance. This is accomplished by using online simulations of a model of the actual system to automate scheduler parameter adjustment. Here we augment our power-aware scheduling policy by incorporating multiple queues into the backfilling mechanism, which provides dramatic improvement in job performance while retaining significant power savings.

1 Introduction

Contemporary scheduling policies in high-end distributed memory systems (e.g., Maui scheduler and PBS) aim at minimizing job wait time while maximizing system utilization, typically using variations of the basic FCFS discipline. Backfilling [9] is often found in the core of these schedulers [2], and the availability of multiple waiting queues and a variety of configuration parameters allow the system administrator to customize the scheduling policy according to the site's needs. With backfilling, jobs are not executed in strictly FCFS order, but instead certain shorter jobs are allowed to move ahead for execution provided their execution does not delay certain previously submitted jobs. The goal of backfilling is to decrease system fragmentation and increase system utilization [9, 10] by using otherwise idle processors for immediate execution.

Despite these efforts to maximize system utilization, analysis of supercomputer workloads indicate the existence of significant time periods of low system utilization [5]. This observation, along with the growing trend for developing techniques for power management [1], motivated the development of a scheduling policy that integrates scheduling and power management [5] — automating scheduler parameterization in the presence of varying workload conditions to improve power consumption and cooling costs

for high-end systems. The scheduler periodically uses online simulation [6] to model the system under different numbers of available processors, selecting the minimum number of processors necessary to meet negotiated user service level agreements (SLAs). The system then raises or lowers the number of processors available for scheduling in the actual system to match that suggested by the online simulations. Detailed experiments using workloads from the Parallel Workloads Archive [3] indicate that this policy can result in significant savings in the number of “active” (and thus power consuming) processors across the system lifetime without significant compromises on predefined SLAs [5]. Although our power-aware scheduling policy was exhibited in the context of the well established single-queue EASY backfilling policy [8, 9], our policy can be applied to any scheduling policy, even in systems that support multiple levels of SLAs.

Despite the 10% or more savings in the number of active processors exhibited by our power-aware policy, the main drawback is that roughly a factor of two increase in job slowdown can be expected. In this work, we augment our power-aware scheduling policy by incorporating multiple-queue backfilling [4], shown to be more effective than EASY at improving job slowdown. In the current context, the addition of multiple-queue backfilling to the power-aware scheduling policy dramatically improves job slowdown while simultaneously providing comparable or better power savings.

2 Experimental Methodology

In this work we consider high-performance batch scheduling systems in which jobs run to completion (i.e., no preemption). Job scheduling is based on the EASY scheduler [8, 9], which incorporates backfilling and a single queue of jobs. Backfilling requires users to provide an estimate of the job runtime. Based on the estimates, certain short jobs are permitted to move ahead for execution provided specific other jobs are not delayed. Even in the presence of notoriously inaccurate estimates [7, 9], backfilling reduces system fragmentation, increases utilization, and improves performance by using processors that would otherwise remain idle. Unless clearly stated otherwise, for the results in this work we use actual execution times as (exact) user estimates.

Our results to follow are obtained by simulation using the following supercomputer workloads from the Parallel Workloads Archive [3]:

- **CTC:** 77 222 jobs on a 512-node IBM SP2 at the Cornell Theory Center, June 1996–May 1997;

*This work was partially supported by the National Science Foundation under grants CCR-0098278, ACI-0090221, ITR-0428330, CCF-0346867, and ACI-0312980.

- **KTH**: 28 490 jobs on a 100-node IBM SP2 at the Swedish Royal Institute of Technology, September 1996–August 1997;
- **SDSC-SP2**: 59 725 jobs on a 128-node IBM SP2 at the San Diego Supercomputer Center, April 1998–April 2000;
- **Blue Horizon**: 243 314 jobs on a 144-node with 8 processors per node IBM SP at the San Diego Supercomputer Center, April 2000–January 2003.

From the traces, for each job we extract the job arrival time (i.e., submission time), number of processors requested, and the estimated and actual durations of the job. Because we do not use job completion times from the traces, the scheduling strategies used on the corresponding systems are not relevant to our study.

We consider *aggregate* performance measures, i.e., average statistics computed for an entire simulation run, giving a single quantifiable measure of performance. Because workload characteristics vary dramatically across time [4, 5, 6], we also consider *transient* performance measures, i.e., per-week snapshot statistics plotted versus time to illustrate how well a policy reacts to workload variability. Because we are interested in reducing the number of active processors to reduce power and cooling requirements without severely impacting job performance, we consider two critical performance measures. From the job perspective, the measure of interest is each job’s bounded slowdown [9] defined by $s = 1 + (d / \max\{10, \nu\})$, where d and ν are respectively the delay time and actual service time of the job. From the system perspective, utilization provides a measure of goodness of the power-aware policy: if utilization is very high (resulting from too large a reduction in the number of active processors), job performance will suffer; if utilization is very low (resulting from too many active processors), job performance will increase at the expense of increased system support costs. In the context of a fluctuating number of active processors, utilization is computed relative to the current number of processors active [5].

3 Experimental Results

In this work we compare three separate scheduling policies: EASY with a single queue and all processors always active; EASY with a single queue and power-aware online simulation (OLS) [5] to selectively adapt the number of active processors; and multiple-queue backfilling (MQBF) [4] with power-aware OLS. A brief description of the latter two policies follows; we direct the interested reader to the appropriate articles for more details.

To augment single-queue EASY with power-aware OLS, the scheduler periodically executes several lightweight simulations of the actual system (using only the current state of the system and collection of jobs present — no new arrivals are simulated). Each online simulation models the system under a different number of processors, fixed for the lifetime of that simulation. The scheduler then selects the minimum number of processors that gives simulated results meeting a predefined service level agreement (SLA), and modifies the number of active processors in the actual system (e.g., by placing processors in “sleep” mode). In this manner, the scheduler reacts to transient conditions in the workload, predicting the best number of processors for the future with the goal of reducing the amount of power and cooling needed by the system.

In this work we present new results obtained by combining MQBF and power-aware OLS. The power-aware OLS approach remains as described above. MQBF splits the system into multiple partitions (the number of partitions is fixed), with one queue

per partition, and *separates* short from long jobs by assigning incoming jobs to different queues based on user runtime estimates. Initially, processors are distributed evenly among the partitions, but as time evolves processors may move from one partition to another so that processors currently idle in one partition may be used for immediate backfilling in another. In this manner, the system is able to adjust to transient workload conditions, and the average job slowdown is reduced by diminishing the likelihood that a short job is delayed behind a long job.

For the results to follow, we use a slowdown threshold of 200 as the SLA for the online simulations, and we use four queues in the multiple-queue backfilling policy. Different choices for the slowdown threshold and the number of queues yield different but qualitatively similar results.

Figure 1 depicts for each of the four workloads (a) the aggregate slowdown and (b) the utilization for an EASY all-processor run, for an EASY with power-aware OLS run, and for an MQBF with power-aware OLS run. Also depicted in (c) is the proportion of processors that are maintained in an inactive state relative to an EASY all-processor run, i.e., the overall processor savings obtained by using power-aware OLS. As shown in this figure, relative to all processors active, the addition of power-aware OLS to the EASY scheduler results in increased utilization and a processor savings of approximately 10% or more at the expense of roughly a factor of two increase in job slowdown¹. However, replacing the single-queue EASY/OLS policy with MQBF/OLS results in dramatic decreases in job slowdown while maintaining comparable or better utilization and processor savings. (Further note that the MQBF/OLS slowdowns are nearly equal to or less than *all-processor* EASY for SDSC-SP2 and Blue Horizon.) The message is clear: power-aware OLS is able to provide meaningful power and cooling cost savings, while MQBF is able to account for variabilities in the workload to yield significantly better job performance — combining the two policies provides mutual benefits from both.

Figure 2 depicts transient slowdown and utilization versus wallclock time for the KTH workload. The dashed lines represent results from using single-queue EASY/OLS while the solid lines represent MQBF/OLS. These transient results are consistent with the aggregate results in Figure 1. MQBF/OLS generally provides lower slowdown and higher utilization across time. Although there are a few small windows of time during which EASY/OLS performs slightly better (e.g., slowdown for weeks 37 and 38), MQBF/OLS performs better, often significantly so, during many more of the windows. (Transient results from the other three workloads are qualitatively similar but omitted for brevity.)

Furthermore, Figure 3(a) depicts for the SDSC-SP2 workload the number of active processors across time — this number changes according to the results of the power-aware online simulations. The solid line represents MQBF/OLS and the dashed line represents EASY/OLS. The fluctuation in number of active processors tends to be less noisy when using MQBF/OLS compared to EASY/OLS. This observation is important when considering the physical stresses that may result from powering up and down portions of the system. Figure 3(b) depicts an empirical cumulative density function of the percentage of jobs versus job slowdown. Again, the combination of MQBF with power-aware OLS provides better results, yielding a higher percentage of jobs with

¹The CTC workload with all processors active experiences utilization of only 55%. Therefore the dramatic OLS results for CTC are exaggerated when compared to the other systems which have higher utilization with all processors active.

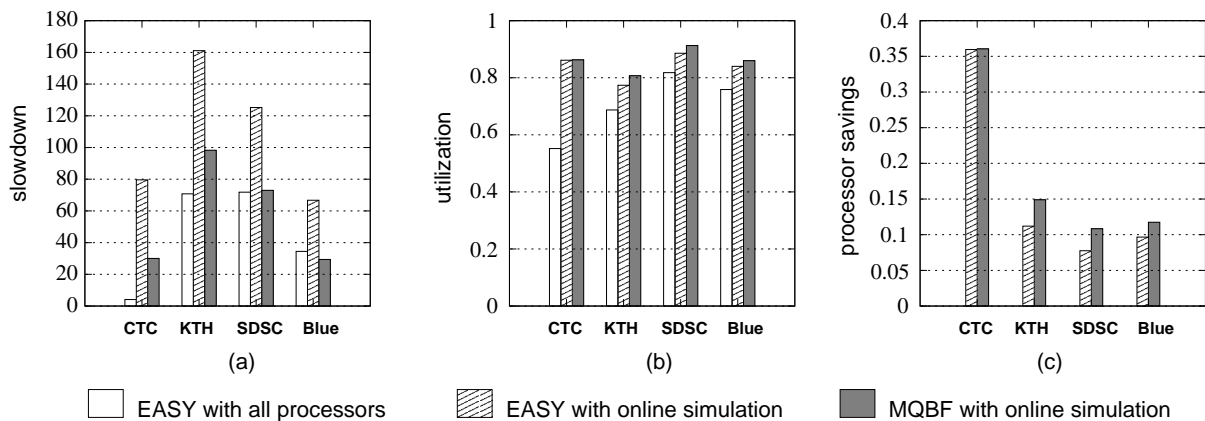


Figure 1. (a) Aggregate slowdown and (b) aggregate utilization for an all-processor EASY simulation run, an EASY/OLS policy run, and an MQBF/OLS policy run; (c) proportion of processors maintained in an inactive state.

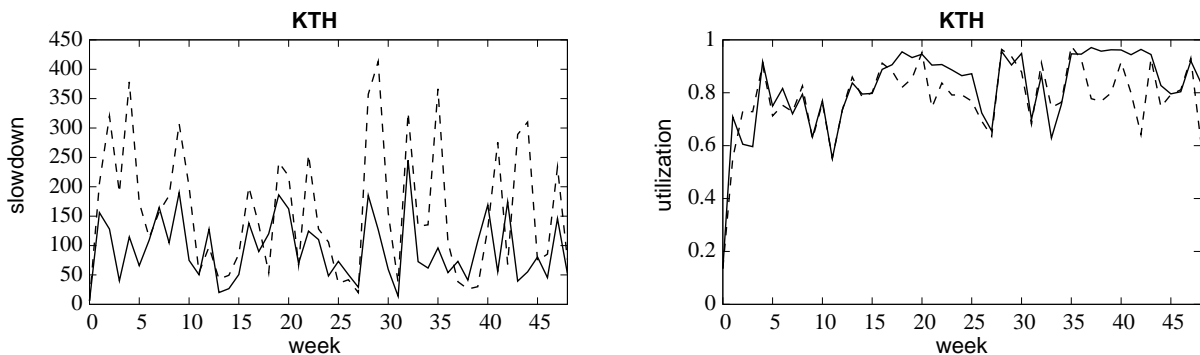


Figure 2. Slowdown and utilization as a function of time for the KTH workload. For dashed lines, the EASY/OLS policy is used; for solid lines, the MQBF/OLS policy is used.

lower slowdown. (Results for the other three workloads are similar.)

Finally, Figure 4 is analogous to Figure 1, except that in this case inexact user runtime estimates are used for backfilling rather than actual (exact estimate) runtimes. When using inexact runtime estimates, the slowdown achieved for each of the three policies (EASY, EASY/OLS, and MQBF/OLS) tends to be higher than when using the same policy in the presence of exact estimates. Nonetheless, as shown in Figure 4, overall trends are similar to those obtained when using exact estimates — EASY/OLS achieves higher utilization and processor savings than EASY at the expense of higher slowdown; MQBF/OLS maintains the high utilization and processor savings of EASY/OLS while reducing job slowdown. These results provide evidence that MQBF/OLS works well even in the presence of inaccurate runtime estimates. (When using inexact runtime estimates, transient results for the four workloads are qualitatively similar to those obtained when using exact estimates but are omitted for brevity.)

4 Conclusions and Further Work

We presented a power-aware scheduling policy, extending previous results from our work, that aims at minimizing power con-

sumption of high-performance systems while maintaining user-negotiated SLAs. Via detailed simulations using traces from the Parallel Workloads Archive, we demonstrated that combining multiple-queue backfilling and power-aware online simulation provides mutual benefits. Multiple-queue backfilling is able to address workload variability, resulting in improved job performance compared to single-queue EASY, while power-aware online simulation simultaneously provides adaptive parameterization to minimize operating costs. The result is a flexible scheduling policy that can address job performance expectations, automate scheduler parameterization for the administrator, and reduce power and cooling costs. Ongoing work addresses multiple levels of SLAs in the form of different classes of service. Future work will focus on constructing and implementing an integrated two-level scheduling framework, leveraging collaborative opportunities at the job scheduling and kernel levels, for optimizing potential power savings while maintaining high performance.

References

- [1] J. Chase and R. P. Doyle. Energy management for server clusters. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*. ACM, 2001.

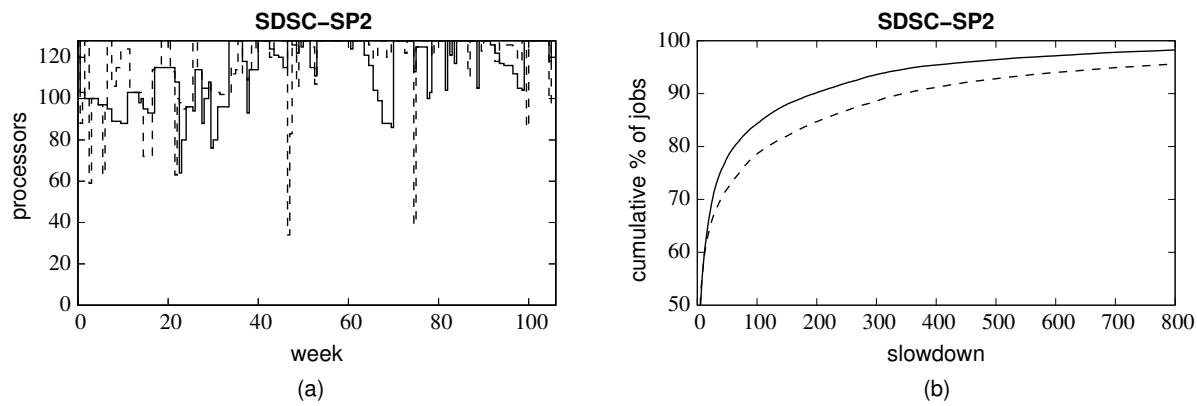


Figure 3. For the SDSC-SP2 workload, (a) number of active processors as a function of time and (b) cumulative percentage of jobs versus job slowdown. For dashed lines, the EASY/OLS policy is used; for solid lines, the MQBF/OLS policy is used.

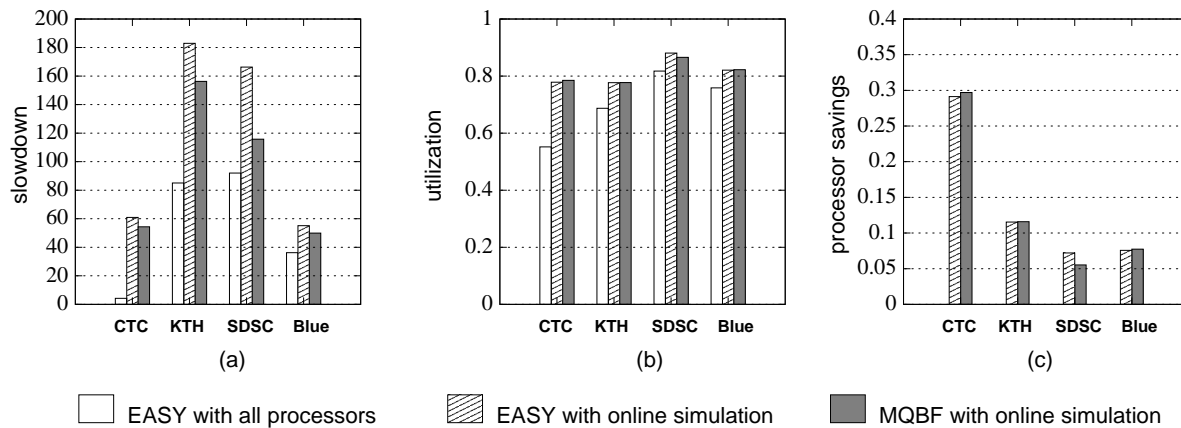


Figure 4. Using inaccurate user runtime estimates, (a) aggregate slowdown and (b) aggregate utilization for an all-processor EASY simulation run, an EASY/OLS policy run, and an MQBF/OLS policy run; (c) proportion of processors maintained in an inactive state.

- [2] D. G. Feitelson, L. Rudolph, and U. Schwiegelshohn. Parallel job scheduling — a status report. In *Job Scheduling Strategies for Parallel Processing (JSSPP 2004)*, pages 1–16. Springer-Verlag, 2004. LNCS vol. 3277.
- [3] Parallel Workloads Archive. <http://www.cs.huji.ac.il/labs/parallel/workload/>.
- [4] B. Lawson and E. Smirni. Multiple-queue backfilling scheduling with priorities and reservations for parallel systems. In *Job Scheduling Strategies for Parallel Processing: (JSSPP 2002)*, pages 72–87. Springer Verlag, 2002. LNCS vol. 2537.
- [5] B. Lawson and E. Smirni. Power-aware resource allocation in high-end systems via online simulation. In *Proceedings of the 19th ACM International Conference on Supercomputing (ICS05)*, Cambridge, MA, June 2005.
- [6] B. Lawson and E. Smirni. Self-adaptive scheduler parameterization via online simulation. In *Proceedings of the 19th International Parallel and Distributed Processing Symposium (IPDPS 2005)*, Denver, CO, April 2005.
- [7] C. B. Lee, Y. Schwartzman, J. Hardy, and A. Snaveley. Are user runtime estimates inherently inaccurate? In *Job Scheduling Strategies for Parallel Processing (JSSPP 2004)*, pages 253–263. Springer-Verlag, 2004. LNCS vol. 3277.
- [8] D. Lifka. The ANL/IBM SP scheduling system. In *Job Scheduling Strategies for Parallel Processing (JSSPP 1995)*, pages 295–303. Springer Verlag, 1995. LNCS vol. 949.
- [9] A. Mualem and D. G. Feitelson. Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. *IEEE Transactions on Parallel and Distributed Systems*, 12(6):529–543, June 2001.
- [10] D. Talby and D. Feitelson. Supporting priorities and improving utilization of the IBM SP2 scheduler using slack-based backfilling. In *Proceedings of the 13th International Parallel Processing Symposium*, pages 513–517, April 1999.