Final Report for Starr Research Fellowship 2009

Improving Job Scheduling on Large Scale High Performance Computing Systems

Wei Tang
Department of Computer Science
wtang6@iit.edu

Introduction

A resource management system manages the processing load by preventing jobs from competing with each other for limited compute resources. Typically, a resource management system comprises a resource manager and a job scheduler. The scheduler communicates with the resource manager to obtain information about queues, loads on compute nodes, and resource availability to make scheduling decisions. Job scheduling is a critical task on large-scale systems, where small differences in scheduling policies can result in poor use of substantial resources.

We tried ways to improve job scheduling performance on large scale high performance computing system. Our study is based on Cobalt resource manager [5] which is used on production Blue Gene/P systems [3], including Intrepid deployed at Argonne National Laboratory. In this report, we will present two studies which are both published (or accepted). First is "utility-based job scheduling [1]," second is "adjusting user runtime estimates to improve job scheduling [2]."

Background

The IBM Blue Gene/P [3] platform is a high-performance computing architecture. It is highly scalable, with three out of the top ten systems on the June 2009 Top500 list [4] belonging to the Blue Gene family. Blue Gene systems use a novel partitioned 3D torus interconnect in order to provide good network performance to concurrent jobs running on the system. This network partitioning isolates jobs from one another, providing extremely reproducible job performance, as well as improved network performance compared with a single, shared-torus network.

Intrepid is a 557 TF, 40-rack Blue Gene/P system deployed at Argonne National Laboratory. This system comprises 40,960 quad-core nodes, with 163,840 cores, associated I/O nodes, storage servers, and an I/O network. It debuted as No. 3 in the TOP 500 supercomputer list released in June 2008 and was ranked No. 8 in the latest list released in November 2009 [4]. Intrepid has been in full production since the beginning of 2009. The system is used primarily for scientific and engineering computing. The vast majority of the use is allocated to awardees of the DOE Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program [6].

Cobalt is an open-source, component-based resource management suite used on a large number of Blue Gene/L and Blue Gene/P systems worldwide, including Intrepid. Cobalt comprises 12,000 lines of Python code at the current release. Cobalt components correspond to pieces of functionality in resource management systems, such as scheduling, queue management, hardware resource management, and process management. Its component architecture allows easy replacement of key software functionality. This allows Qsim [1], our queue simulation component, to service the queue manager component and system component interface without changing any other software components. Hence, Qsim can interface directly with an unmodified Cobalt scheduler.

Utility-based job scheduling

While traditional metrics for scheduling such as utilization rates and average response times have long been used to assess scheduler policy performance, large centers now face an increasing set of considerations in scheduling. Examples of these considerations include avoiding system faults, minimizing power consumption during peak demand, and sharing I/O resources. At the same time, these individual considerations do not occur in a vacuum; systemwide metrics, such as utilization and average response time, as well as other characteristics such as fairness, remain important. A mechanism is needed to explicitly balance these considerations. Moreover, this balance is not universal; different systems have varied priorities that result in some considerations prevailing over others.

While many scheduling algorithms have been presented in the past, our work is motivated by operational problems in job scheduling and aims to provide a general utility-based scheduling framework to balance various scheduling requirements and priorities. In our framework, scheduling policies are described as job-scoring functions called *utility functions* that can be changed on the fly. During each scheduling iterate, each job's score is evaluated, allowing the scheduler to take the appropriate action. Utility functions are implemented in Python code and can take job parameters, such as length, size, and waiting time, as well as other factors, into consideration. For example, administrators can use them to alter the balance between responsiveness and utilization rate on a dynamic basis.

The performance of scheduling policies is completely dependent of the system workload, so intuitive assessment of scheduling policies is frequently not reliable. To address this issue, we have developed *Qsim*, a simulator of queue behavior over time, based on real system workload inputs. Using this, we can explore the behavior of different scheduling policies. In other words, not only can basic functionality be tested, but likely behavior under system load can be predicted. Thus, by using Qsim to test utility functions machine owners can build confidence in new scheduling and allocation policies prior to deployment.

We have evaluated our fault-aware, utility-based job scheduling with job logs collected from the 40-rack Blue Gene/P system called *Intrepid* at Argonne National Laboratory. The results demonstrate that, as compared to the conventional scheduling policy FCFS (first-come first-serve) with backfilling, our utility functions can lower the average response time and slowdown significantly (by up to 55%). We have also examined how the utility functions achieve their scheduling goals individually, and we have provided instructive comparison among them which results in the deployment of utility functions for the real system.

Adjusting user runtime estimates to improve job scheduling

Backfilling and short-job-first are widely acknowledged enhancements to the simple but popular first-come, first-served job scheduling policy. However, both backfilling and order relaxation schemes depend on user-reported estimates of job runtimes. Backfilling considers the runtimes of currently running jobs as well as the length of candidates, for backfilling to ensure that response time for high-priority jobs is minimally impacted. For relaxed FCFS, estimated job runtimes are used to calculate job priorities. For this reason, these approaches are highly dependent on the accuracy of user estimates of job runtimes, which have been repeatedly demonstrated to be highly inaccurate [7][8].

Does the inaccuracy of user runtime estimates impact job scheduling? In other words, will more accurate user runtime estimates improve performance? A number of studies have addressed the problem. Some have shown that the inaccuracy of runtime estimation barely degrades performance [9]. Such results have led to

the suggestion that estimates should be doubled [10] or randomized [11] to make them even less accurate. However, others have shown that using more accurate estimated runtimes can improve system performance far more significantly than previously suggested [5].

For this study, we quantified the effects of runtime estimation inaccuracy on backfilling, in both FCFS and relaxed FCFS scheduling, in a Blue Gene/P setting. First, we studied how the inaccuracy of user runtime estimates impacts job scheduling policies. Using different priority policies with the same backfilling scheme, we investigated how more accurate runtime estimates affect the scheduling. Next, we customized the scheduler code to explore which part of the scheduling (job queue prioritizing or backfilling) is more sensitive to the accuracy of runtime estimates. Based on the results, we designed several estimation-adjusting schemes using historical workload data. The experimental results demonstrated that our adjusting schemes can achieve up to 20% improvement on job-scheduling performance measured by average waiting time, unitless wait, and slowdown.

In-progress and Future Work

We plan to extend our study in several ways in the future. First, we are in the progress of extend the resource manger to support proactive failure avoidance and post-failure handling. For the former, we have already had simulation-based study to implement a fault-aware job scheduling. For the latter, we have also had paper submission on the topic "automatic and coordinated job recovery for high performance computing". Second, we plan to put effort on coordinate the scheduling of computing resource and the allocation of storage resource. That is, co-scheduling computing and data in large scale system, in order to further improve the performance of the computing center.

Acknowledgement

First, I would thank IIT graduate college to offer me Starr Research Fellowship during the year 2009. Second, I would thank Dr Zhiling Lan, who is my advisor at IIT; Narayan Desai, who is my advisor during my summer internship at Argonne National Laboratory; and Daniel Buettner, who is also working in Cobalt team. All of them have given me great support in my research.

Reference

[1] W. Tang, Z. Lan, N. Desai, and D. Buettner, "Fault-aware, utility-based job scheduling on Blue Gene/P systems," in *Proc. of IEEE International Conference on Cluster Computing*, 2009.
[2] W. Tang, N. Desai, D. Buettner, and Z. Lan "Analyzing and Adjusting user runtime estimates to improve job scheduling on the Blue Gene/P," to appear in *Proc. of IEEE International Parallel and Distributed Processing Symposium*, 2010.
[3] Blue Gene Team, "Overview of the IBM Blue Gene/P project," *IBM Journal of Research and Development* 52(1/2), 199–220, 2008.
[4] TOP500 Supercomputing web site, http://www.top500.org
[5] Cobalt project. *http://trac.mcs.anl.gov/projects/cobalt*
[6] DOE INCITE program. http://www.er.doe.gov/ascr/incite
[7] W. Cirne and F. Berman, "A comprehensive model of the supercomputer workload." in *Proc. of IEEE International Workshop on Workload Characterization*, 2001.
[8] S.-H. Chiang, A. Arpaci-Dusseau, and M. Vernon, "The impact of more accurate requested runtimes on production job scheduling performance," in *Proc. of Job Scheduling Strategies for Parallel Processing*, 2002.
[9] A. Mu'alem and D. 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, 2001.
[10] D. Zotkin and P. Keleher, "Job-length estimation and performance in backfilling schedulers," in *Proc. of IEEE International Symposium on High Performance Distributed Computing*, 1999.
[11] D. Perkovic and P. Keleher, "Randomization, speculation, and adaptation in batch schedulers," *IEEE/ACM Supercomputing Conference*, 2000.