Parallel Computing 39 (2013) 753–768

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

Toward balanced and sustainable job scheduling for production supercomputers

Wei Tang [a,*], Dongxu Ren [b], Zhiling Lan [b], Narayan Desai [a]

[a] *Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL, USA*
[b] *Department of Computer Science, Illinois Institute of Technology, Chicago, IL, USA*

ARTICLE INFO

Article history:
Available online 4 September 2013

Keywords:
Job scheduling
Resource management
Priority balancing
Adaptive policy tuning
Workload characteristic

ABSTRACT

Job scheduling on production supercomputers is complicated by diverse demands of system administrators and amorphous characteristics of workloads. Specifically, various scheduling goals such as queuing efficiency and system utilization are usually conflicting and thus need to be balanced. Also, changing workload characteristics often impact the effectiveness of the deployed scheduling policies. Thus it is challenging to design a versatile scheduling policy that is effective in all circumstances. In this paper, we propose a novel job scheduling strategy to balance diverse scheduling goals and mitigate the impact of workload characteristics. First, we introduce metric-aware scheduling, which enables the scheduler to balance competing scheduling goals represented by different metrics such as job waiting time, fairness, and system utilization. Second, we design a scheme to dynamically adjust scheduling policies based on feedback information of monitored metrics at runtime. We evaluate our design using real workloads from supercomputer centers. The results demonstrate that our scheduling mechanism can significantly improve system performance in a balanced, sustainable fashion.

1. Introduction

Job scheduling is a critical task on large-scale computing platforms. The job scheduling policy directly influences the satisfaction of both users and system owners. Moreover, the success of a job scheduling policy is largely determined by the satisfaction of these stakeholders. Users are concerned with fast job turnaround and fairness, while system owners are interested in system utilization. Also, production supercomputing centers are starting to face new challenges in scheduling, such as avoiding failure interrupts and achieving energy efficiency. All these considerations, which are quantified by system metrics, are related but often conflict with one another. Even worse, the priorities differ from machine to machine and from time to time, further complicating the design of a comprehensive job scheduling policy.

Traditional scheduling policies can achieve specific scheduling goals but not balance them well. For example, using “first-come, first served” (FCFS) achieves good job fairness but results in poor response times and resource fragmentation. On the other hand, using “short-job first” (SJF) achieves best response time in theory but violates job fairness and causes job starvation. Essentially, these approaches attempt to favor a fixed combination of some priorities while ignoring others. Some traditional schedulers provide mechanisms to switch between these policies when particular boundary conditions are encountered; however, this approach provides only a coarse ability to refine goal-based priorities.

* Corresponding author. Tel.: +1 3129121207.
E-mail addresses: wtang@mcs.anl.gov (W. Tang), dren1@iit.edu (D. Ren), lan@iit.edu (Z. Lan), desai@mcs.anl.gov (N. Desai).

0167-8191/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.parco.2013.08.007

Moreover, user satisfaction and system performance are not considered in a holistic fashion. Typically, job prioritizing and resource allocation are separated into two subsequent phases in decision making. This division greatly constrains the resource allocation process. For example, when a high-priority job suffers from insufficient resources, it reserves the resources while draining others; yet, these resources could be used to execute other low-priority jobs, thereby improving system performance. This kind of resource draining causes external fragmentation. While backfilling helps in this case, it only mitigates fragmentation already created by this division [21].

Another issue of job scheduling concerns dynamic workload. Even though we may identify a policy to achieve our integrated goals well for one workload, the policy may fail for a different workload. Although event-driven simulation can be used to evaluate the aggregate effect of a scheduling policy on a historical workload trace, it cannot provide much guidance when workload properties change dynamically at runtime.

Motivated by these issues, we propose an adaptive metric-aware job scheduling mechanism. Our objectives are twofold. First, we develop a metric-aware job scheduling mechanism to prioritize jobs and allocate resources in an integrated fashion. Here, "metric-aware" means we take the targeted performance metrics into account when configuring a specific scheduling policy. Moreover, the prioritized jobs are allowed to be altered in a limited fashion during the resource allocation phase. This approach improves flexibility in schedule creation.

Second, we introduce an adaptive tuning mechanism into job scheduling, which allows the scheduling policy to change dynamically at runtime based on workload characteristics. By monitoring the performance metrics at runtime, the scheduler can adjust its scheduling policy to favor the metrics that are less satisfied recently, thereby mitigating the impact of changing workload characteristics on the scheduling policy. For example, if the system utilization rate is below a certain threshold (e.g., a longer-term average), our scheduling system will adjust its policy to favor system utilization more than other metrics.

We implemented our design in the production resource management system called Cobalt [1]. We evaluated our design using recent real job traces from multiple production supercomputers. The experimental results show that our approach can achieve significant and sustainable performance improvement compared with traditional scheduling strategies.

The remainder of this paper is organized as follows. Section 2 discusses some related work. Section 3 reviews the motivation of our work and describes our methodology. Section 4 illustrates our experimental results. Section 5 summarizes the paper and briefly discusses future work.

2. Related work

The balancing of multiple scheduling objectives is supported by some production job schedulers. One example is the Maui scheduler [10], which uses a number of weighted and combined parameters to prioritize jobs. Moab [2], its commercial descendant, is extremely flexible and supports more than 250 scheduling parameters. To alleviate the tedious work of manual configuration for a Moab scheduling policy, Krishnamurthy et al. [11] provided a toolset that can help a system administrator to automatically configure a scheduling policy. Basically, the toolset uses a genetic algorithm-based scheme working with simulations from historical workloads to find an effective configuration. The open-source Cobalt resource manager [1] uses a simple utility function to prioritize jobs, which can also take into account multiple scheduling considerations. Cobalt provides an event-driven simulator to guide the design of an appropriate utility function [22]. While our approach also provides the ability to balance different scheduling goals, it does not rely on the simulation results of recent workloads. That is, the parameters of a scheduling policy can be tuned at runtime based on the feedback of monitored performance metrics, thereby adapting to different workload characteristics.

A dynamic tuning scheduling policy can be found in some existing work. Grothlags and Streit [19] and Streit [20]8 proposed a self-tuning "dynP" job scheduler that can tune queuing policy dynamically during runtime. Our work shares similar motivation but differs from those works in two ways. First, whereas the "dynP" scheduler switches policy between FCFS, SJF, and LJF (largest job first) based on the number of jobs in the queue, our scheduler supports fine-grained tuning based on more sophisticated monitoring of a number of targeted system metrics. Second, in our work, both the queuing policy and the job allocation policy can be tuned, either independently or in a two-dimensional fashion.

Feedback-based scheduling has been used in operating systems or real-time systems. Blevins and Ramamoorthy [5] proposed using feedback information to adjust the schedule in general-purpose operating systems in the form of multilevel feedback queue scheduling. Lu et al. [12] designed and evaluated a feedback control earliest-deadline-first scheduling algorithm for scheduling in real-time systems. In parallel job scheduling, however, schedulers generally use "open loop" scheduling algorithms in which policies are not adjusted based on continuous feedback. Yet, in production supercomputers the dynamics of the workload is amorphous and can influence the effectiveness of a scheduling policy. Thus, utilizing feedback information of workload change is beneficial in the selection of a scheduling policy. It is one of the motivations of our work. Our work differs from existing feedback-based scheduling efforts, however, in that we do not adjust the schedules directly but instead adjust the scheduling policy that will indirectly change the schedules.

Etsion and Tssafrir [7] reported that the prevalent default scheduler setting is FCFS and that in those management suites that also support backfilling, the governing scheme used is EASY [13]. Indeed, considerable work has been done to enhance either FCFS or EASY backfilling. Ababneh and Bani-Mohammad [4] proposed an enhancement to FCFS that uses a window of consecutive jobs from which jobs are selected for allocation and execution. Shmueli et al. [17] optimized the packing of backfilling jobs by looking ahead into the queue. Srinivasan and Feitelson [18] designed a selective reservation strategy