Parallel Job Scheduling — A Status Report

Dror G. Feitelson[1], Larry Rudolph[2], and Uwe Schwiegelshohn[3]

[1] School of Computer Science and Engineering
The Hebrew University of Jerusalem
91904 Jerusalem, Israel
[2] Laboratory for Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
[3] Computer Engineering Institute
Universität Dortmund
44221 Dortmund, Germany

1 Introduction

The popularity of research on the scheduling of parallel jobs demands a periodic review of the status of the field. Indeed, several surveys have been written on this topic in the context of parallel supercomputers [17, 20]. The purpose of the present paper is to update that material, and to extend it to include work concerning clusters and the grid.

The paper is divided into three major parts. The first part addresses algorithmic and research issues covering the two main approaches: backfilling and gang scheduling. For each, recent advances are reviewed, both in terms of how to perform the scheduling and in terms of understanding the performance results. An underlying theme of the surveyed results is the shift from dogmatic use of rigid formulations to a more flexible approach. This reflects a maturation of the field and improved concern for real-world issues.

The second part of the paper addresses current usage. It presents a short overview of vendor offerings, and then reviews the scheduling frameworks used by top-ranking parallel systems. For vendor offerings, we highlight the distinction between what is done in a research setting and what is actually developed for production use. Regarding actual usage, we consider the alternative options of procurement of an existing system vs. the development of an in-house solution that more directly reflects desired attributes.

The third part of the paper looks both back and forward in time. As with any field, the success, popularity, and influence of a particular approach depends on a range of factors. We review some less successful ones. It is possible that some of these techniques may only be relevant to future machines. The paper, therefor concludes with some observations about the near-term future.

This paper contains a large number of references. In order to highlight the more recent results, i.e. those with publication dates in this millennium, their citation will be superscripted with the last two digits of the publication date.

D. Feitelson, L. Rudolph, and U. Schwiegelshohn (Eds.): JSSPP 2004, LNCS 3277, pp. 1–16, 2005.

2 Advances in Parallel Job Scheduling Research

There are many different ways to schedule parallel jobs and their constituent threads [17], but only a few mechanisms are used in practice and studied in detail. This section reviews backfilling and gang scheduling strategies, their variants, and their connections. The special requirements and strategies for scheduling parallel jobs on a grid are addressed as well.

2.1 Backfilling

The most basic batch scheduling algorithm is First-Come-First-Serve (FCFS) [43] where jobs are considered in order of arrival. Each job specifies the number of processors it requires and is placed in a FIFO queue upon arrival. If there are sufficient available processors to run the job at the head of the queue, the processors are allocated and the job is started. If there are not enough, the scheduler waits for some currently running job to terminate and free additional processors.

Backfilling is an optimization that tries to balance the goals of utilization and maintaining FCFS order. It requires that each job also specifies its maximum execution time. While the job at the head of the queue is waiting, it is possible for other, smaller jobs, to be scheduled, especially if they would not delay the start of the job on the head of the queue. Processors get to be used that would otherwise remain idle.

By letting some jobs execute out of order, other jobs may get delayed. Backfilling will never completely violate the FCFS order where some jobs are never run (a phenomenon known as "starvation"). In particular, jobs that need to wait are typically given a reservation for some future time.

The use of reservations was included in several early batch schedulers [29, 8]. Backfilling, in which small jobs move forward to utilize the idle resources, was introduced by Lifka [33]. This was done in the context of EASY, the Extensible Argonne Scheduling sYstem, which was developed for the first large IBM SP1 installation at Argonne National Lab.

Variations on Backfilling While the concept of backfilling is quite simple, it nevertheless has several variants with subtle differences. We generalize the behavior of backfilling by parameterizing several constants. Judicial choice of parameter values lead to improved performance.

One parameter is the *number of reservations*. In the original EASY backfilling algorithm, only the first queued job received a reservation. Jobs may be scheduled out of order only if they do not delay the job at the head of the queue. The scheduler estimates when a sufficient number of processors will be available for that job and reserves them for this job. Other backfilled jobs may not violate this reservation, they must either terminate before the time of the reservation (known as the "shadow time"), or use only processors that are not required by the first job [33].