Adaptively Parallel Processor Allocation for Cilk Jobs

Siddhartha Sen, Kunal Agrawal
Computer Science and Artificial Intelligence Laboratory (CSAIL)
MIT

***Abstract*— The problem of allocating processor resources fairly and efficiently to parallel jobs has been studied extensively in the past. Most of this work, however, assumes that the instantaneous parallelism of the jobs is known and used by the job scheduler to make its decisions. In this project, we consider different ways of estimating the parallelism of jobs during runtime, as well as different ways of using this information to allocate processors in a fair and efficient manner.**

The goal of our project is to design and implement a dynamic processor-allocation system for adaptively parallel jobs. Adaptively parallel jobs are jobs for which the number of processors which can be used without waste—in other words, the parallelism of each job—varies during execution. We call the problem of allocating processors to adaptively parallel jobs the adaptively parallel processor-allocation problem.

We propose to investigate the adaptively parallel processor-allocation problem for jobs running on a shared-memory multiprocessor (SMP) system. We focus on the specific case of parallel jobs that are scheduled with the randomized work-stealing algorithm, as is used in the Cilk multithreaded language (later, we will expand the scope of our research to include other kinds of parallel jobs). Our problem can be defined as follows:

Consider an SMP system with P processors and J jobs. At any given time, each job has a desire dj, representing the maximum number of efficiently usable processors, and an allotment mj, representing the number of processors allocated to it. Our problem is to design a processor-allocation system that achieves a fair and efficient allocation of processors among these jobs, where the terms "fair" and "efficient" are defined as follows: an allocation is fair if whenever a job receives fewer processors than it desires, then no other job receives more than one more processor than this job received (the allowance of one processor is due to integer roundoff); an allocation is efficient if no job receives more processors than it desires and, if there exists a job that receives fewer processors than it desires, then all P processors are in use.

The successful design of such a system can contribute significantly to our understanding of the adaptively parallel processor-allocation problem. If the task of estimating processor desires is designated to the jobs themselves—and if algorithms are devised to perform this estimation for different types of parallel jobs—then the system can easily be generalized into a core scheduling service provided by the kernel. In addition, the system can be extended to respect the job priority mechanism supported by the kernel, or to take into account processor usage histories when making the allocation decisions. Moreover, we can consider other ways of characterizing the fairness and efficiency of processor allocations beyond the definitions we have provided above.

Currently, the prototype of the processor-allocation system we are developing is a user-mode extension to Cilk that uses the steal rate of Cilk jobs to estimate processor desires and a randomized, pair-wise exchange of processors to achieve a fair and efficient allocation. The prototype has a well-defined interface and a modular architecture, allowing us to investigate different desire-estimation and processor-allocation algorithms without too much difficulty.

[Full Text Not Available]