Exploiting Task and Data Parallelism on a Multicomputer

Jaspal Subhlok, James M. Stichnoth, David R. O'Hallaron, and Thomas Gross
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

For many applications, achieving good performance on a private memory parallel computer requires exploiting data parallelism as well as task parallelism. Depending on the size of the input data set and the number of nodes (i.e., processors), different tradeoffs between task and data parallelism are appropriate for a parallel system. Most existing compilers focus on only one of data parallelism and task parallelism. Therefore, to achieve the desired results, the programmer must separately program the data and task parallelism. We have taken a unified approach to exploiting both kinds of parallelism in a single framework with an existing language. This approach eases the task of programming and exposes the tradeoffs between data and task parallelism to the compiler. We have implemented a parallelizing Fortran compiler for the iWarp system based on this approach. We discuss the design of our compiler, and present performance results to validate our approach.

1 Introduction

Many applications can be naturally expressed as collections of parallel or pipelined tasks, with coarse grained parallelism among the tasks, and finer grained parallelism within each task [7]. For example, a sonar spectral detection system consists of a sequence of parallel pipelines, where each pipeline consists of a sequence of downsampling time domain filters, followed by an FFT, thresholding, and other postprocessing operators [16].

Two widely used styles of parallelism for private memory multicomputers are data parallelism [9, 11, 20, 19, 10] and task parallelism [6, 13, 12, 14]. Data parallelism is typically expressed as a single thread of control operating on data sets distributed over all nodes. It is especially useful when the size of the data sets can be scaled to fit the size of the parallel machine. Task parallelism is typically expressed as a collection of sequential processes with explicit communication between them.

For some applications it is essential to exploit both styles of parallelism within the same program. In particular, for signal processing applications, physical considerations often make it impossible to arbitrarily increase the size of the data sets. Thus, it may be impossible for a single data parallel thread of control to use all nodes effectively. For example, in a signal processing application the number of input sensors can be limited by size, weight, power, and cost considerations. Further, the number of frequencies of interest is constrained by the sampling rate and the physical characteristics of the signal being analyzed. The result is that, as the number of nodes in modern multicomputers increases, it becomes increasingly difficult to use all nodes effectively on a single task. This phenomenon was encountered in [15], where it was shown analytically that at most 8 nodes could be used effectively in a purely data parallel sonar spectral detection system, primarily because the data set sizes are fairly small and cannot be increased.

The point is that, depending on the sizes of the machine and the data sets, the same source program can benefit from different styles of parallelism: data parallelism, task parallelism, or some combination of the two. In this paper, we discuss a practical approach to accomplish this. We have implemented a parallelizing Fortran compiler for the iWarp multicomputer [2, 3]. The key idea is that a *single source program* can be translated to fit a particular machine or data set size. The style of the generated code is based on compiler

Supported in part by the Defense Advanced Research Projects Agency, Information Science and Technology Office, under the title "Research on Parallel Computing," ARPA Order No. 7330. Work furnished in connection with this research is provided under prime contract MDA972–90–C–0035 issued by DARPA/CMO to Carnegie Mellon University, and in part by the Air Force Office of Scientific Research under Contract F49620–92–J–0131. This material is based upon work supported under a National Science Foundation Graduate Research Fellowship.