A Portable, Object-Based Parallel Library and Layered Framework for Real-Time Radar Signal Processing

Cecelia DeLuca, Curtis W. Heisey, Robert A. Bond, and Jim M. Daly
cdeluca@ll.mit.edu, heisey@ll.mit.edu, rbond@ll.mit.edu
Massachusetts Institute of Technology Lincoln Laboratory*
244 Wood Street, Lexington, MA 02173-9108

Abstract. We have developed an object-based, layered framework and associated library in C for real-time radar applications. Object classes allow us to reuse code modules, and a layered framework enhances the portability of applications. The framework is divided into a machine-dependent kernel layer, a mathematical library layer, and an application layer. We meet performance requirements by highly optimizing the kernel layer, and by performing allocations and preparations for data transfers during a set-up time. Our initial application employs a space-time adaptive processing (STAP) algorithm and requires throughput on the order of 20 Gflop/s (sustained), with 1 s latency. We present performance results for a key portion of the STAP algorithm and discuss future work.

1 Introduction

For large-scale, real-time applications on parallel computers such as the Cray T3E and the Mercury RACE series, both performance and modularity are critical. We have developed an object-based, layered framework and an associated signal processing library for real-time radar applications. Initially the library will be used to perform space-time adaptive processing (STAP) algorithms, which are used in radar systems to suppress interference and detect targets. The algorithms consist of a large data cube transiting through a sequence of signal processing stages, or subsystems, and include routines such as matrix multiplies, matrix factorizations, and FFTs (Ward 1994). A typical operation at a given stage involves slicing the data cube into matrices and performing independent operations on the set (Figure 1). We have incorporated both task and data parallelism into our libraries so that multiple subsystems can be pipelined, and operations within a subsystem can be performed in parallel. Data redistribution occurs both within and between subsystems.

Our goals are portability, ease of use, and reusable software components along with the competing constraints of high throughput (20 Gflop/s sustained)

* This work was sponsored under Air Force Contract F19628-95-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Air Force.
Fig. 1. The STAP application combines both task and data parallelism. Data (arrows) moves through a series of tasks (boxes), which may themselves be pipelines of tasks.

and low latency (about 1 s). We found no existing software environment which satisfied all of our requirements.

The object-oriented, C-based approach satisfies the demands of portability, modularity, and reusability and allows the use of a standard compiler. We improve portability through layering as well, by restricting machine-dependent code to a low-level communication kernel. To achieve real-time performance, we prepare for data transfers and perform allocations at set-up time. We have also created mechanisms for graceful departure from object-based design when performance requirements demand it, though initial performance comparisons suggest that this capability is rarely needed.

We begin by discussing related research. Next we provide an overview of our parallel programming model, which describes how classes relate to parallelism and flow of control. We describe the layers of our framework, reviewing the function and structure of each level, and provide detailed descriptions of the major classes. We conclude with a performance analysis of a key library routine and discuss our plans for future work. Standard definitions for architectural taxonomy and programming paradigms are used (e.g, Wilson, 1995).

2 Related Work

Our work lies at the intersection of several research areas. One area is that of parallel object-oriented languages for general purpose use; another is the development of serial and parallel object-oriented languages and libraries just for signal processing. We also integrate into our work research on optimal mappings for algorithms such as STAP on parallel machines. Below, we highlight the work in each of these fields which is most closely related to our own.

General-purpose object-oriented libraries and languages. A multitude of general purpose parallel, object-oriented languages, libraries and frameworks have been developed over the past decade, including C++// (Caromel et al. 1996), and POOMA (Reynders et al. 1996). We share basic design constructs with many of these packages. The layers of our software are simpler than but close to the multi-tiered structure of POOMA. Like the active objects in C++//, our software includes data objects which coordinate their own computations, data transfers and synchronizations. However, none of the environments we have en-