Implementing a Non-strict Functional Programming Language on a Threaded Architecture

Shigeru Kusakabe†, Kentaro Inenaga†, Makoto Amamiya†, Xinan Tang‡, Andres Marquez‡, and Guang R. Gao‡

†Dept. of Intelligent Systems, Kyushu University, 
‡EE & CE Dept. University of Delaware
E-mail: kusakabe@is.kyushu-u.ac.jp

Abstract. The combination of a language with fine-grain implicit parallelism and a dataflow evaluation scheme is suitable for high-level programming on massively parallel architectures. We are developing a compiler of V, a non-strict functional programming language, for EARTH (Efficient Architecture for Running Threads). Our compiler generates codes in Threaded-C, which is a lower-level programming language for EARTH. We have developed translation rules, and integrated them into the compiler. Since overhead caused by fine-grain processing may degrade performance for programs with little parallelism, we have adopted a thread merging rule. The preliminary performance results are encouraging. Although further improvement is required for non-strict data-structures, some codes generated from V programs by our compiler achieved comparable performance with the performance of hand-written Threaded-C codes.

1 Introduction

Many fine-grain multithreaded architectures have been proposed as promising multiprocessor architectures because of their ability to tolerate communication and synchronization latencies inherent in massively parallel machines[1][3][9][12][14][18]. By providing a lot of threads and supporting fast switches among threads, multithreaded architectures can hide communication and synchronization latencies. Control of execution is switched to a new thread when a long-latency operation is encountered. Many explicit parallel languages have been proposed for fine-grain multithreaded programming[7][8][17][19]. Overlapping computation and communication is programmer’s task when using this kind of explicit languages. Writing explicit parallel programs on parallel machines is still a skilled job.

We are developing a high-level parallel programming language, called “V,” which would minimize the difficulties in writing massively parallel programs[15]. In order to provide a high-level abstraction, the language is a non-strict functional programming language with implicit parallelism. There is no anti-dependence in V programs and it is easy to extract parallelism at various levels including fine-grain parallelism from V programs. The underlying computation model is an optimized dataflow computation model, Datarol[4]. The combination of a functional language
with implicit parallelism and a fine-grain multithread evaluation scheme is suitable for massively parallel computation. The languages abstracts the timing problems in writing massively parallel programs, while fine-grain multithread evaluation supports efficient execution of a large number of fine-grain processes for implicit parallel functional programs in a highly concurrent way.

This approach can exploit irregular and dynamic parallelism, thus support efficient execution of "non-optimal" codes. Since our language does not support explicit descriptions for parallel execution and data-mapping, it is necessary for the compiler to automatically extract parallel codes of optimal grain size, partition data-structures and map them to each processing node. However, this is not an easy task, and the compiler may generate non-optimal codes of non-uniform grain size and ill-mapped data-structures. The multithreaded architectures with the ability to tolerate computation and synchronization latencies alleviate the problems of non-optimal code.

In this paper, we discuss implementation issues of V on a multithreaded architecture EARTH (Efficient Architecture for Running THreads). In order to show the feasibility of our language, we have implemented our language on commercially available machines such as Sequent Symmetry and Fujitsu AP1000[16]. However, since our basic execution model is a multithreaded execution model extended from Datatrol model, multithreaded architectures which have special support for fine-grain parallel processing are preferable for our language. EARTH realizes efficient multithreading with off-the-shell microprocessor-based processing nodes[12]. We are implementing V on EARTH, while there are two explicit parallel languages for EARTH, EARTH-C[11] as a higher-level language and Threaded-C[21] as a lower-level language. The goal of this work is to show adequate compiler support makes our implicit parallel language as efficient as the explicit parallel languages on the multithreaded architecture. Our final aim is to realize a high-level programming environment on a high performance architecture.

In compilation, we use a virtual machine code, DVMC (Datatrol Virtual Machine Code), as an intermediate code, and Threaded-C as a target code. Threaded-C is an explicit multi-threaded language, targeted for the EARTH model. We have developed translation rules, and integrated them into the compiler. Since overhead caused by fine-grain processing may degrade performance for programs with little parallelism, we have adopted a thread merging rule. The preliminary performance results on EARTH-MANNA[12][10] are encouraging. Although further improvement is required for non-strict data-structures, some codes generated from V programs by our compiler achieved comparable performance with the performance of hand-written Threaded-C codes.

This paper is organized as follows: section 2 and section 3 introduce our language V and EARTH respectively. Section 4 explains our compilation method. Section 5 discusses implementation issues of fine-grain data-structures on EARTH. Section 6 makes concluding remarks.