A Compilation Approach for Fortran 90D/HPF Compilers

Zeki Bozkus, Alok Choudhary, Geoffrey Fox, Tomasz Haupt, and Sanjay Ranka

Northeast Parallel Architectures Center
3-201, Center for Science and Technology
Syracuse University
Syracuse, NY 13244-4100

Abstract. This paper describes a compilation approach for a Fortran 90D/HPF compiler, a source-to-source parallel compiler for distributed memory systems. Different from Fortran 77 parallelizing compilers, a Fortran 90D/HPF compiler does not parallelize sequential constructs. Only parallelism expressed by Fortran 90D/HPF parallel constructs is exploited. The methodology of compilation of Fortran 90D/HPF programs such as computation partitioning, communication detection and generation are discussed. An example of Gaussian Elimination is used to illustrate the compilation techniques with performance results.

1 Introduction

Distributed memory multiprocessors are increasingly being used for providing high performance for scientific applications. Currently, distributed memory machines are programmed using a node language and a message passing library. This process is tedious and error prone because the user must perform the task of data distribution and communication for non-local data access.

There has been significant research in developing parallelizing compilers. In this approach, the compiler takes a sequential program, e.g. a Fortran 77 program as input, applies a set of transformation rules, and produces a parallelized code for the target machine. However, a sequential language, such as Fortran 77, obscures the parallelism of a problem in sequential loops and other sequential constructs. This makes the potential parallelism of a program more difficult to detect by a parallelizing compiler. Therefore, compiling a sequential program into a parallel program is not a natural approach. An alternative approach is to use a programming language that can naturally represent an application without losing the application's original parallelism. Fortran 90 [1] (with some extensions) is such a language. The extensions may include the forall statement and compiler directives for data partitioning, such as decomposition, alignment, and distribution. Fortran 90 with these extensions is what we call "Fortran 90D", a Fortran 90 version of the Fortran D language [8]. We developed the Fortran D language with our colleagues at Rice University. There is an analogous version of Fortran 77 with compiler directives and other constructs, called Fortran 77D. Fortran D allows the user to advise the compiler on the allocation of data to processor
memories. Recently, the High Performance Fortran Forum, an informal group of people from academia, industry and national labs, led by Ken Kennedy, developed a language called HPF (High Performance Fortran) [13] based on Fortran D. HPF essentially adds extensions to Fortran 90 similar to Fortran D directives. Hence, Fortran 90D and HPF are very similar except a few differences. For this reason, we call our compiler the Fortran 90D/HPF compiler.

This paper presents the design of a prototype compiler for Fortran 90D/HPF. The compiler takes as input a program written in Fortran 90D/HPF. Its output is SPMD (Single Program Multiple Data) program with appropriate data and computation partitioning and communication calls for MIMD machines. Therefore, the user can still program using a data parallel language but is relieved of the responsibility to perform data distribution and communication.

Tremendous effort in the last decade has been devoted to the goal of running existing Fortran programs on new parallel machines. Restructuring compilers for Fortran 77 programs have been researched extensively for shared memory systems[19]. The compilation technique of Fortran 77 for distributed memory systems has been addressed by Callahan and Kennedy [4]. Currently, a Fortran 77D compiler is being developed at Rice [14]. Hatcher and Quinn provide a working version of a C* compiler. This work converts C* - an extension of C that incorporates features of a data parallel SIMD programming model- into C plus message passing for MIMD distributed memory parallel computes[12]. The ADAPT system [17] compiles Fortran 90 for execution on MIMD distributed memory architectures. The ADAPTOR [2] is a tool that transform data parallel programs written in Fortran with array extension and layout directives to explicit message passing. Li and Chen [6] describes general compiler optimization techniques that reduce communication overhead for Fortran-90 implementation on massively parallel machines. Many techniques especially for unstructured communication of Fortran 90D/HPF compiler are adapted from Saltz et al. [3]. Gupta et al. [11] use collective communication on automatic data partitioning on distributed memory machines. Superb [20] compiles a Fortran 77 program into a semantically equivalent parallel SUPRENUM multiprocessor. Koelbel and Mehrotra [15] present a compilation method where a great deal of effort is put on run-time analysis for optimizing message passing in implementation of Kali.

2 Compilation Overview

Our Fortran90D/HPF parallel compiler exploits only the parallelism expressed in the data parallel constructs. We do not attempt to parallelize other constructs, such as do loops and while loops, since they are used only as naturally sequential control constructs in this language. The foundation of our design lies in recognizing commonly occurring computation and communication patterns. These patterns are then replaced by calls to the optimized run-time support system routines. The run-time support system includes parallel intrinsic functions, data distribution functions, communication primitives and several other miscellaneous routines. This approach represents a significant departure from