High-Level Languages for Parallel Scientific Computing *

Barbara Chapman a  Piyush Mehrotra b  Hans Zima a

a Institute for Software Technology and Parallel Systems, University of Vienna, Liechtensteinstr. 22, A-1090 Vienna, Austria
E-Mail: {barbara,zima}@par.univie.ac.at
b ICASE, MS 132C, NASA Langley Research Center, Hampton VA 23681 USA
E-Mail: pm@icase.edu

Abstract. Highly parallel scalable multiprocessing systems are powerful tools for solving large-scale scientific and engineering problems. Languages such as Vienna Fortran and High Performance Fortran (HPF) have been introduced to allow the programming of these machines at a relatively high level of abstraction, based on the data-parallel Single-Program-Multiple-Data (SPMD) model. Their main features focus on the distribution of data across the processors of a machine. In this paper, we outline the state-of-the-art in this area and provide a detailed description of HPF. A significant weakness of current HPF is its lack of support for many advanced applications, which require irregular data distributions, dynamic load balancing, or task parallelism. We introduce HPF+, an extension of HPF based on Vienna Fortran, that addresses these problems and provides the required functionality.

1 Introduction

The continued demand for increased computing power has led to the development of highly parallel scalable multiprocessing systems (HMPs), which are now offered by all major vendors and have rapidly gained user acceptance. These machines are relatively inexpensive to build, and are potentially scalable to large numbers of processors. However, they are difficult to program: most of the architectures exhibit non-uniformity of memory access which implies that the locality of algorithms must be exploited in order to achieve high performance, and the management of data becomes of paramount importance.

Traditionally, HMPs have been programmed using a standard sequential programming language (Fortran or C), augmented with message passing constructs.

* The work described in this paper was partially supported by the Austrian Research Foundation (FWF Grant P8989-PHY) and by the Austrian Ministry for Science and Research (BMWF Grant GZ 308.9281- IV/3/93). This research was also supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-18605, while the authors were in residence at ICASE, NASA Langley Research Center, Hampton, VA 23681.
In this paradigm, the user is forced to deal with all aspects of the distribution of data and work to the processors, and to control the program’s execution by explicitly inserting message passing operations. The resulting programming style can be compared to assembly language programming for a sequential machine; it has led to slow software development cycles and high costs for software production. Moreover, although MPI is evolving as a standard for message passing, the portability of MPI-based programs is limited since the characteristics of the target architectures may require extensive restructuring of the code.

As a consequence, much research and development activity has been concentrated in recent years on providing higher-level programming paradigms for HMPs. Vienna Fortran, building upon the KALI programming language [12] and experiences from the SUPERB parallelization system [22], was the first fully specified data-parallel language for HMPs. It provides language features for the high-level specification of data distribution and alignment, as well as explicitly parallel loops. High Performance Fortran (HPF) [9], a de-facto standard developed by a consortium including participants from industry, academia, and research laboratories, is based on concepts of CM Fortran [21], Vienna Fortran, and Fortran D [8]. It provides support for regular applications, alleviating the task of the programmer for a certain segment of applications. However, it is generally agreed that the current version of the language, HPF-1, is not adequate to handle many advanced applications, such as multiblock codes, unstructured meshes, adaptive grid codes, or sparse matrix computations, without incurring significant overheads with respect to memory or execution time.

This paper is structured as follows. In Section 2, we provide a detailed description of HPF-1. We then identify some of the weaknesses of the language by considering requirements posed by irregular algorithms and dynamic load balancing. This study leads to the discussion of an HPF extension, “HPF+”, which, based upon Vienna Fortran, addresses many of these problems and thus contributes to the present effort of the HPF Forum for defining a suitable successor to HPF-1 (Section 3). The paper concludes with Section 4.

2 High Performance Fortran (HPF-1)

Recently an international group of researchers from academia, industry and government laboratories formed the High Performance Fortran Forum aimed at providing an approach in which the user and the compiler share responsibility for exploiting parallelism. The main goal of the group has been to design a high-level set of standard extensions to Fortran called, High Performance Fortran (HPF), intended to exploit a wide variety of parallel architectures [9].

The HPF extensions allow the user to carefully control the distribution of data across the memories of the target machine. However, the computation code is written using a global name space with no explicit message passing statements. It is then the compiler’s responsibility to analyze the distribution annotations