Chapter 1

Research Summary

1.1 Research Accomplishments

1.1.1 Fortran Parallel Programming Systems

Participants: Ken Kennedy (Director), Vikram Adve, Erol Akarsu, Ruth Aydt, Rajesh Bordawekar, Alan Carle, Bryan Carpenter, Alok Choudhary, Keith Cooper, Luiz DeRose, Kivanc Dincer, Ian Foster, Geoffrey Fox, Robert Fowler, Bill Gropp, Tom Haupt, Paul Havlak, Chuck Koelbel, David Kohr, Rakesh Krishnaiyer, Don Leskiw, Xiaoming Li, Xinying Li, Rusty Lusk, Nat McIntosh, John Mellor-Crummey, Paul Messina, Michael Paleczny, Dan Reed, Joel Saltz, Alan Sussman, Rajeev Thakur, Linda Torczon, Yuhong Wen, Guansong Zhang, and Ying Zhang.

HPF and HPF Testbed. High Performance Fortran (HPF) provides a relatively high-level language for programming scientific computations.

In the past year, the HPF version 2.0 language specification was published as a CRPC technical report. This updated definition includes corrections, simplifications to the HPF core language, and a set of "approved extensions" to provide better support for irregular computations, task parallelism, and asynchronous I/O operations. Implementations of the core language are expected by early 1998.

We also have finished the development and documentation of a set of HPF codes which both illustrate the use of the language and can be used to benchmark compilers. We have submitted a summary of this work to the *NHSE Review.*

Support for Machine-Independent Parallel Programming. This project focuses on compiler and tool support for exploiting data parallelism. Over the past year, the group's efforts have focused on research and development of the dHPF compiler, a second generation compiler for HPF. Three principal thrusts of this effort have been to design and develop (a) code generation support for shared memory systems, (b) advanced optimizations for message-passing systems, and (c) compiler support for source-level programming tools. The tools effort is nearing completion of a prototype interface that visualizes complex relationships between source program fragments and their corresponding generated code.

Performance Tools for HPF. We have continued to work with the Portland Group (PGI) to augment our tools for correlating dynamic performance data with HPF source code. The result of this collaboration among Illinois, Rice, and PGI is a tool called svPablo that can correlate the dynamic execution behavior with HPF source code. This new tool provides a variety of performance hierarchy displays, ranging from procedural to individual source code construct and integrates both software and hardware (MIPS R10000 counter data on the SGI Origin) performance measurements. The svPablo distribution is available via the WWW at `http://www-pablo.cs.uiuc.edu/Projects/HPF/svpablo.html`.

Scalable I/O Support. The Scalable I/O Consortium is launching a broad-based attack on the parallel I/O problem, i.e., efficiently transferring large volumes of data into and out of parallel machines. The Fortran Parallel Programming Systems group is concentrating on a subproblem, support for out-of-core computations. In past years, we developed compiler techniques for converting in-core computations to use out-of-core data structures. This year saw the first fully automatic implementation of those techniques, based on using the dHPF compiler framework to generate calls to the PASSION parallel I/O library. This allows the programmer to concentrate on the logic of the program, rather than spending great efforts in managing the I/O directly.

Irregular Problem Support. Compiler work at Maryland has concentrated on the support of programs with pointers. Constructing DEF/USE chains for statements with pointers provides compilers with information about the set of statements that may traverse the pointer links constructed by a particular statement. One application of this information is to generate communication schedules for programs with recursive data structures (e.g. linked lists or trees), because DEF/USE information describes where the referenced recursive data structures are constructed and modified.

Data-Parallel Library Interoperability. Runtime library work at Maryland has focused on developing techniques for transferring data between distributed data structures owned by different data-parallel runtime libraries. These techniques have been implemented in the Maryland Meta-Chaos library, which currently is capable of transferring data between (distributed) arrays that have been created using High Performance Fortran, the Maryland CHAOS and Multiblock PARTI libraries, and the Indiana pC++ runtime library, Tulip.

Parallel Compiler Runtime. We have released an HPF 2.0 front end with an extensive checking package. We have developed a simple Fortran to Java translator. A complete documented and tested public domain HPF/HPC++ regular runtime will be released soon.

Integration of Programming Languages with the Web. The CRPC co-organized and participated in workshops that explored the role of Java and the Web in Computational Science. Research has focused on three areas: (1) for the PCRC runtime, we are developing a Java interface that includes a Java MPI linkage; (2) we have adapted our basic course in computational science to include a tutorial in Java as a scientific language and included Java plus MPI in our Virtual Programming Laboratory used in this course; and (3) we have developed an extensive Java and JavaScript front end for scientific programming, which includes SciVis, a Java-based visualization package.

Optimizing Java. We have begun an investigation of performance-enhancing optimizations of Java, especially for scientific applications. Our initial focus has been on inlining code in an

optimizer that uses the Java Virtual Machine language as both source and target. Systematic inlining has been shown to yield integer factor improvements on many Java programs. A challenge is to implement inlining in a way that does not create security holes. This research has developed a novel strategy for inlining whole objects in situations where method inlining would violate security. The strategy has been implemented in a prototype optimizer developed jointly with JavaSoft, Inc.

1.2 Research Plan

1.2.1 Fortran Parallel Programming Systems

HPF. During the next year, the major activities related to HPF will be technology transfer and user education. We will work with PGI and other vendors to move compiler techniques from our research dHPF compiler to production systems. HPF tutorials will be presented at the SC '98 conference and used in the DOD HPC Modernization Program.

Support for Machine-Independent Parallel Programming. The dHPF compiler has matured to the point where it can be used with application benchmarks in addition to kernels. A significant focus over the next year will be performance studies of application benchmarks to assess the benefits of the compiler's advanced analysis and optimizations.

Performance Tools for HPF. We plan to continue to focus on performance scalability instrumentation and predictions, with emphasis on predictions for distributed shared memory (DSM) systems. We will continue to work to integrate our software with the dHPF compiler. We will explore the effects of code generation models and DSM data movement on HPF application performance.

Scalable I/O Support. We intend to link the Rice out-of-core compiler with the performance instrumentation available through Pablo (another component of the SIO effort).

Irregular Problem Support. We will continue to develop new compiler techniques that take advantage of code traversal patterns for linked data structures and to investigate the application of these techniques to programs containing conflicting data structure and traversal patterns.

Data-Parallel Library Interoperability. We will work with Syracuse to implement the inquiry functions required to use Meta-Chaos with the new Syracuse HPF compiler.

Optimizing Java. We will continue to investigate new strategies for obtaining high performance from Java code using ideas developed in the context of interprocedural optimization of Fortran codes, including the use of more aggressive global inlining strategies.

Performance Analysis. Syracuse and Maryland are collaborating on an ARPA sponsored program to develop performance estimation tools oriented to the rapid prototyping of designs as needed for instance in exploring architectures for petaOP performance. Deliverables include key generic applications and an object-oriented performance estimation tool called PetaSim.