Syracuse University
SURFACE

Northeast Parallel Architecture Center | College of Engineering and Computer Science

1994

Design of an Application Development Toolkit for HPF/Fortran 90D

Manish Parashar
University of Texas at Austin

Salim Hariri
Syracuse University

Tomasz Haupt
Syracuse University, haupt@npac.syr.edu

Geoffrey C. Fox
Syracuse University

Follow this and additional works at: https://surface.syr.edu/npac

 Part of the Software Engineering Commons

Recommended Citation
Parashar, Manish; Hariri, Salim; Haupt, Tomasz; and Fox, Geoffrey C., "Design of an Application Development Toolkit for HPF/Fortran 90D" (1994). *Northeast Parallel Architecture Center*. 93.
https://surface.syr.edu/npac/93

This Article is brought to you for free and open access by the College of Engineering and Computer Science at SURFACE. It has been accepted for inclusion in Northeast Parallel Architecture Center by an authorized administrator of SURFACE. For more information, please contact surface@syr.edu.

Design of an Application Development Toolkit for HPF/Fortran 90D*

Manish Parashar
Center for Relativity &
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712-1081
parashar@einstein.ph.utexas.edu

Salim Hariri, Tomasz Haupt, & Geoffrey C. Fox
Northeast Parallel Architectures Center
Syracuse University
Syracuse, NY 13244-4100
{hariri,haupt,gcf}@npac.syr.edu

Presented at IWPP, December, 1994

Abstract

The development of efficient application software capable of exploiting available High Performance Computing (HPC) systems is non-trivial and is largely governed by the availability of sufficiently high-level languages, tools, and application development environments. In this paper we describe the design and operation of a toolkit for HPF/Fortran 90D application development. The toolkit incorporates the following systems: (1) **ESP***: An Interpretive Framework for HPF/Fortran 90D Performance Prediction; (2)* **ESP-i***: A HPF/Fortran 90D Functional Interpreter; and (3)* **ESPial***: An Integrated Environment for HPF/Fortran 90D Application Development & Execution.*

*The toolkit has been implemented on the iPSC/860 hypercube system, and is supported by an interactive, graphical user interface (*ESPView*) which provides application developers with the following functionality: design evaluation capability, functional verification capability, performance visualization support, experimentation capability, compilation support, and execution support.*

1 Introduction

Although currently available High Performance Computing (HPC) systems possess large computing capabilities, few existing applications are able to fully exploit this potential. The fact remains that development of efficient application software capable of exploiting available computing potential is non-trivial and is largely governed by the availability of sufficiently high-level languages, tools, and application development environments. Critical issues that need to be addressed by application development environments include [1]:

- The need for a high-level application description medium (language) that is architecture independent and can be compiled into performance codes for different machines.
- The ability to assist the developer in making appropriate design decision, in effectively resolving and tuning the available degrees of freedom so as to optimize the implementation, and in ensuring that the implementation meets required performance constraints on different machine configurations and under different run-time situations.
- The ability to verify the functionality of an implementation for different machine configurations and under different run-time situations.

In this paper we describe the design and operation of a HPF/Fortran 90D application development toolkit that addresses these issues. The toolkit incorporates the following systems:

1. **ESP**: An Interpretive Framework for HPF/Fortran 90D Performance Prediction;
2. **ESP-i**: A HPF/Fortran 90D Functional Interpreter; and
3. **ESPial**: An Integrated Environment for HPF/Fortran 90D Application Development & Execution.

High Performance Fortran (HPF) is a dialect of Fortran that has been designed to provide portable, high-level expression to parallel algorithms on a variety of

*The presented research has been jointly sponsored by DARPA under contract #DABT63-91-k-0005 and by Rome Labs under contract #F30602-92-C-0150. The content of the information does not necessary reflect the position or the policy of the sponsors and no official endorsement should be inferred.

parallel machines, and is the emerging language standard for HPC. HPF/Fortran 90D refers to a formally defined subset of HPF and is the application description medium used by the toolkit.

ESP is a framework for HPF/Fortran 90D performance prediction. This framework uses a novel interpretive approach [2, 3] to provide accurate and cost-effective performance prediction of HPF/Fortran 90D applications and can be effectively used to make design decisions during application development. ESP addresses the second issue listed above.

ESP-i extends ESP to interpret the functionality of HPF/Fortran 90D and provides the means for verifying the functionality of an implementation.

Finally, ESPial integrates ESP and ESP-i with a HPF/Fortran 90D source to source compiler to form an integrated environment for HPF/Fortran 90D application development and execution. The source to source compiler has been developed at NPAC, Syracuse University and generates efficient Fortran + Message Passing code for a number of MIMD platforms.

The HPF/Fortran 90D application development toolkit has been implemented on the iPSC/860 hypercube system, and is supported by an interactive, graphical user interface (*ESPView*) which provides application developers with the following functionality: design evaluation capability, functional verification capability, performance visualization support, experimentation capability, compilation support, and execution support.

The rest of the paper is organized as follows: Section 2 presents an overview of HPF/Fortran 90D. Section 3 introduces the interpretive performance prediction approach and describes the design of the ESP performance prediction framework. Section 4 describes the design of ESP-i, the HPF/Fortran 90D functional interpreter. Section 5 describes the development of ESPial and outlines the structure and operation on its graphical user interface. Finally Section 6 presents some concluding remarks and discusses future extensions.

2 An Overview of HPF/Fortran 90D

High Performance Fortran (HPF) [4] is based on the research language Fortran 90D [5] and provides a minimal set of extensions to Fortran 90 to support the data parallel programming model[1]. Extensions incorporated into HPF/Fortran 90D provide a means for explicit expression of parallelism and data mapping. These extensions include compiler directives which are used to advice the compiler how data objects should be assigned to processor memories, and new language features such as the *forall* statement and construct.

HPF adopts a two level mapping using PROCESSORS, ALIGN, DISTRIBUTE, and TEMPLATE compiler directives to map data objects to abstract processors. The data objects (typically array elements) are first *aligned* with an abstract index space called a *template*. The template is then *distributed* onto a rectilinear arrangement of abstract *processors*. The mapping of abstract processors to physical processors is implementation dependent. Data objects not explicitly distributed are mapped according to an implementation dependent default distribution (e.g. replication). Supported distributions include BLOCK and CYCLIC.

Our current implementation of the HPF compiler and performance prediction framework supports a formally defined subset of HPF. The term HPF/Fortran 90D is used to refer to this subset.

3 ESP: An Interpretive Framework for Application Performance Prediction

The ESP performance prediction framework uses the interpretive approach outlined below to predict the performance of HPF/Fortran 90D applications on the Intel iPSC/860 hypercube system. Application characterization is performed at compile time by the framework. The parameters required to characterize the system are generated off-line using existing techniques and system specification. Performance metrics generated by the framework include cumulative execution times, the communication time/computation time breakup, existing overheads, and wait times. Further, this information can be obtained at all levels of the application, i.e. at application level, processing node level, process level, procedure level, or even a single line of code. The frameworks interfaces with the ParaGraph [6] performance visualization package, to provide a graphic view of the predicted metrics.

3.1 The Interpretive Approach to Performance Prediction

The essence of the approach is the application of *interpretation* techniques to performance prediction through an appropriate characterization of the HPC system and the application. A *system characterization* methodology is defined to hierarchically abstract the HPC system into a set of well defined parameters which represents its performance. A corresponding

[1] The data parallel programming model is defined as single threaded, global name space, loosely synchronous parallel computation.

application characterization methodology is defined to abstract a high-level application description into a set of well defined parameters which represents its behavior. Performance prediction is then achieved by interpreting the execution costs of the abstracted application in terms of the parameters exported by the abstracted system.

Application characterization is based on the decomposition of the application description into well defined units of abstraction which are standard, structured programming constructs extended to include communication and synchronization primitives. The structure of these abstraction units is then parameterized. Units can be composed to generate different granularities of application characterization.

Correspondingly, system characterization is based on a hierarchical decomposition of the HPC system. At each level of this decomposition, the performance of individual units is parameterized. The characterization methodology is applicable to any parallel/distributed (possibly heterogeneous) HPC system.

Interpretation functions are defined to interpret the performance of the units of application abstraction in terms of the parameters exported by the abstracted system. A global ordering superimposed onto the system during interpretation enables the approach to model interactions in the system and application. Detailed descriptions of the system and application characterization methodologies and the performance interpretation approach can be found in [7, 8].

3.2 Design of the HPF/Fortran 90D Performance Prediction Framework

The design of ESP is based on the HPF source-to-source compiler technology [9] which translates HPF into loosely synchronous, SPMD (single program, multiple data) Fortran 77 + Message-Passing codes. It uses this technology in conjunction with the performance interpretation model to provide performance estimates for HPF/Fortran 90D applications on a distributed memory MIMD multicomputer.

HPF/Fortran 90D performance prediction is performed in two phases: Phase 1 uses HPF compilation technology to produce a SPMD program structure consisting of Fortran 77 plus calls to run-time routines. Phase 2 then uses the interpretation approach to abstract and interpret the performance of the application. A detailed description of the design of ESP and the application of the interpretive approach to HPF/Fortran 90D as well as the experimental validation of the framework is presented in [3].

4 ESP-i: A HPF/Fortran 90D Functional Interpreter

The ESP-i functional interpreter provides the means for verifying the functionality of a HPF/Fortran 90D implementation during application development. It also provides debugging capabilities to a certain extent. Given a particular system configuration and selection of directives, the tool enables the developer to emulate the execution of an HPF/Fortran 90D application. Further, it allows the developer to identify break points in the application description, and to query the system for the state of application variables at these points during interpretation. Finally, the developer has the capability of experimenting with system parameters and interpreting the application functionality on different system configurations. Functional interpretation performed by ESP-i is based on the system and application characterization models defined by the interpretive performance prediction approach and uses these abstractions to emulate the behavior of the system and application.

ESP-i is implemented as an extension to ESP. It uses the compiler front end to generate a parse tree from the HPF/Fortran 90D application description. The parse tree is then translated into a SPMD program structure using the ESP compilation step. Functional interpretation, then, consists of a sequence of parses over this SPMD structure which performs the following steps:

1. **Abstraction Step:** The abstraction step is an extension to the corresponding step in ESP. It uses the abstraction methodologies defined by the interpretive approach to abstract the HPC system and the application.

2. **Functional Interpretation Step:** The functional interpretation step emulates the execution of the abstracted application on the selected system configuration. This step, again, is an extension of the interpretation step in ESP, and can be viewed as a special case of performance interpretation.

3. **Interactive Step:** The interpretive step is entered when interpretation of the entire application is completed or if a user-defined break point is encountered. In this step, the user is allowed to query the system for the status of application variables. In case of a break point, the user can instruct the system to continue interpretation.

5 ESPial: An Integrated Environment for HPF/Fortran 90D Application Development & Execution

ESPial integrates the ESP and ESP-i systems described above, with the HPF/Fortran 90D source to source compiler to form an integrated environment for HPF/Fortran 90D application development and execution. ESPial enables the user to develop an application, experiment with it and tune it, verify its functionality, select a target machine configuration, automatically compile, load and run the application, and view the outputs produced. The current implementation of ESPial is targeted to the iPSC/860 hypercube system. The ESPial environment provides the developer with the following functionality:

Design Evaluation Capability: Design evaluation refers to the ability to visualize the effects of various design choices, such as, algorithms, data distributions, system configurations, and communication schemes, on the performance of the application, and thereby enabling the developer to make the appropriate design decisions. In the case of HPF/Fortran 90D, this refers to the selection of appropriate directives; i.e. the virtual processor configurations, the distribution of the data elements onto the virtual processors, and their relative alignments.

Functional Verification Capability: Functional verification refers to the ability to verify that the application is functionally correct and produces the required results. Further, it provides the ability of tracking application variables during interpretation by setting appropriate break points in the application description.

Performance Debugging Capability: Performance debugging refers to the ability to analyze the execution of the implementation, so as to identify inefficiencies and bottlenecks. It provides insight into the expected performance of the implementation, and the contribution of various factors effecting this performance. The goal is to ensure that the implementation meets the specified performance constraints. The performance debugger enables the developer to identify regions in the implementation where further refinement or tuning is required.

Performance Visualization Support: Visualization support refers to the ability to graphically view the interpreted application performance. This is achieved using a performance visualization package (ParaGraph) that has been interfaced with ESPial.

Experimentation Capability: Experimentation refers to the ability to evaluate the effects of changes in system run-time status (e.g. load, contention, etc.) and its configuration on the performance of the application, and to study the scalability of the application with respect to system and problem sizes. The effect of the above parameters on the functionality of the application can also be verified.

Compilation Support: Compilation support is provided at two levels: The first level is a translation to Fortran + Message Passing performed by the HPF/Fortran 90D compiler based on the compiler directives. The second level is a node level compilation on the target machine using its compiler. This level generates the program executable.

Execution Support: Execution support includes allocating processing nodes in the HPC system, loading the application executable onto these nodes, and running the application. Outputs and error messages produced during execution are displayed via the ESPial interface.

5.1 ESPView - The Graphical User Interface

ESPial is supported by an interactive, graphical user interface, ESPView (shown in Figure 1), through which, the developer can access the functionality outlined above.[2] ESPView has been implemented using the X based SUIT (Simple User Interface Toolkit) libraries[3] and runs on SUN workstations. The operation of the interface is described below:

File Operations The **File** menu enables the developer to select an application description, to save a description (possibly under another name), and to save interpretation and execution outputs.

Directive Selection ESPView displays the current values of the HPF compiler directives. These directives can be edited by the user. The modified source can be saved using the File menu.

[2] "HPF Extensions" in the interface refer to the extensions to HPF developed at NPAC, Syracuse University to support irregular problems.

[3] SUIT [10] is a highly portable GUI development package which is freely available in the public domain. It is based on the external control model which is commonly used in event driven scenarios and has become the norm in windowing environments.

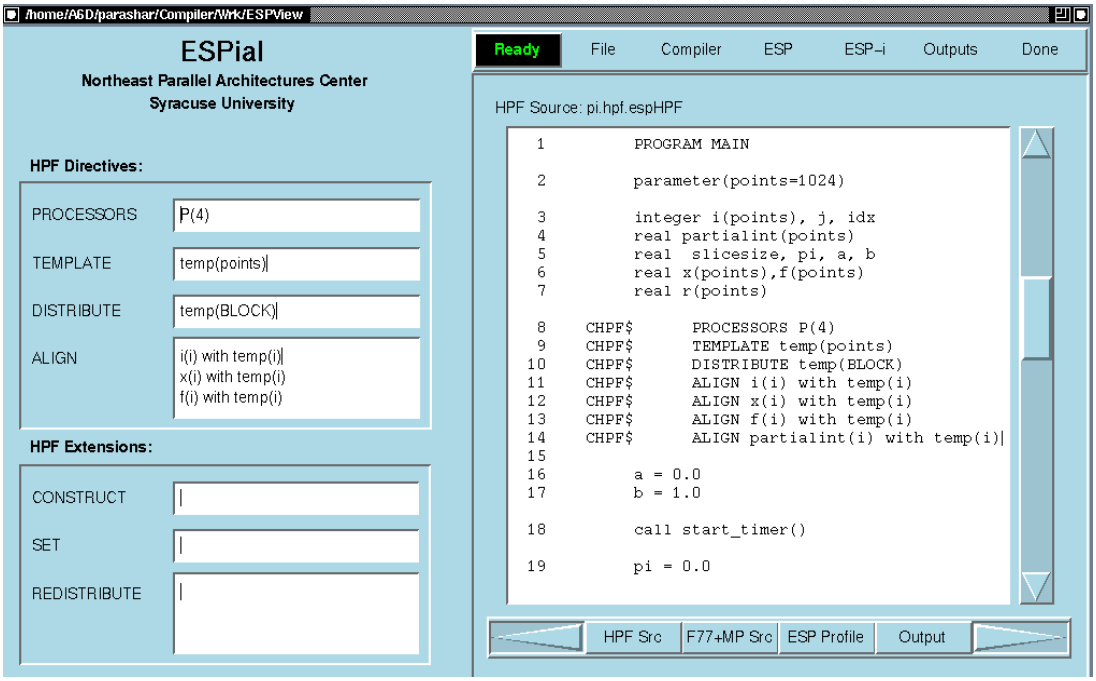

Figure 1: ESPView - The ESPial Graphical User Interface

Display Window The display window is used to display the application description (**HPF Src**) as well the outputs generated by the source to source compiler (**F77+MP Src**), the performance interpreter (**ESP Profile**), and the application itself during execution (**Output**). The user can switch between these displays.

Performance Interpretation Performance interpretation is activated via the **ESP** menu. The menu also provides the interface for setting different interpretation parameters. A special input interface enables the developer to specify the values of critical variables.

Functional Interpretation The **ESP-i** menu houses the commands for functional interpretation. These include setting and clearing break points and activating ESP-i. When stopped at a break point, a special input interface enables the developer to query and view the status application variables.

Compiling & Executing The **Compiler** menu provides access to both levels of compiling; HPF Compile and F77+MP Compile. Compiler options can be set from within this menu. The application can be executed on the desired configuration of the iPSC/860 using the Run button in this menu.

Outputs The **Output** menu enables the developer to select the type and level of output. In addition to the generic profile, the developer can zoom in to specific parts of the application description and view the relevant performance metrics. The menu also provides the interface to ParaGraph performance visualization package. Figure 2 illustrates the ESPial-ParaGraph interface and the use of ParaGraph to visualize the interpreted application performance.

6 Conclusions and Future Work

In this paper we presented the design and operation of an application development toolkit that addresses the key issues in HPC software development. The toolkit provides an interactive environment that supports HPF/Fortran 90D application development and execution. It integrates individual development tools using a graphical user interface and can be easily extended to span other stage of the software development process.

The current implementation of the toolkit is targeted to the Intel iPSC/860 hypercube system, and provides application developer with the following functionality: design evaluation capability, functional verification capability, performance visualization support, experimentation capability, compilation support, and execution support.

Currently, we working on moving the toolkit to dif-

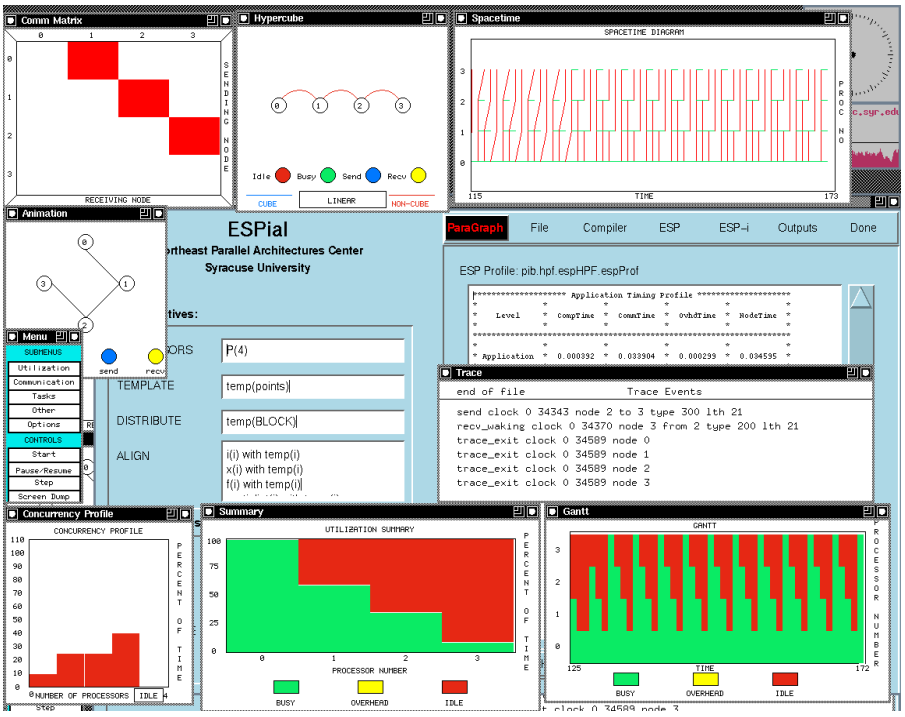

Figure 2: ESPView - Using ParaGraph to Visualize Interpreted Performance

ferent parallel and distributed platforms. Future research will consist of expanding the toolkit to span different stages of applcation development.

References

[1] Manish Parashar, Salim Hariri, Tomasz Haupt, and Geoffrey C. Fox, "A Study of Software Development for High Performance Computing", in Karsten M. Decker and Renè M. Rehmann, editors, *Programming Environments for Massively Parallel Distributed Systems*. Birkhauser Verlag, Basel, Switzerland, Aug. 1994.

[2] Manish Parashar, Salim Hariri, Tomasz Haupt, and Geoffrey C. Fox, "An Interpretive Framework for Application Performance Prediction", *Proceedings of the 1993 International Conference On Parallel and Distributed Systems*, pp. 668–672, Dec. 1993.

[3] Manish Parashar, Salim Hariri, Tomasz Haupt, and Geoffrey C. Fox, "Interpreting the Performance of HPF/Fortran 90D", *Proceedings of Supercomputing '94, Washington DC*, pp. 743–752, Nov. 1994.

[4] High Performance Fortran Forum, *High Performance Fortran Language Specifications, Version 1.0*, Jan. 1993, Also available as Technical Report CRPC-TR92225 from Center for Research on Parallel Computing, Rice University, Houston, TX 77251-1892.

[5] Geoffrey C. Fox, Seema Hiranandani, Ken Kennedy, Charles Koebel, Uli Kremer, Chau-Wen Tseng, and Min-You Wu, "Fortran D Language Specifications", Technical Report SCCS 42c, Northeast Parallel Architectures Center, Syracuse University, Syracuse NY 13244-4100, Dec. 1990, Available via WWW at `http://www.npac.syr.edu`.

[6] M. Heath and J. A. Etheridge, "Paragraph", Technical report, Oak Ridge National Laboratory, Oak Ridge, Tennessee 37831, Oct 1991.

[7] Manish Parashar, Salim Hariri, Tomasz Haupt, and Geoffrey C. Fox, "An Interpretive Framework for Application Performance Prediction", Technical Report SCCS-479, Northeast Parallel Architectures Center, Syracuse University, Syracuse NY 13244-4100, Apr. 1993, Available via WWW at `http://godel.ph.utexas.edu/Members/parashar/ESP/esp.html`.

[8] Manish Parashar, *Interpretive Performance Prediction for High Performance Parallel Computing*, PhD thesis, Syracuse University, 121 Link Hall, Syracuse, NY 13244-1240, July 1994, Available via WWW at `http://godel.ph.utexas.edu/Members/parashar/ESP/esp.html`.

[9] Zeki Bozkus, Alok Choudhary, Geoffrey Fox, Tomasz Haupt, and Sanjay Ranka, "Compiling HPF for Distributed Memory MIMD Computers", in David Lilja and Peter Bird, editors, *Impact of Compilation Technology on Computer Architecture*. Kluwer Academic Publishers, 1993.

[10] Matthew Conway, Randy Pausch, and Kimberly Passarella, "A Tutorial For SUIT: The Simple User In-

terface Toolkit", Technical report, Computer Science Department, University of Virginia, 1992.