Studies of Integration and Optimization of Interpreted and Compiled Languages

— A Proposal to the NSF in response to CCR Programs'97

(NPAC Technical Report SCCS-780)

Geoffrey C. Fox, Xiaoming Li [*], Yuhong Wen, Guansong Zhang

Northeast Parallel Architectures Center at Syracuse University
Syracuse, New York 13244-4100
315-443-2163, {gcf,lxm,wen,zgs}@npac.syr.edu

February 12, 1997

[*] Visiting scholar from Harbin Institute of Technology, China

A. Project summary

The increasing power of computers should be used not just to solve larger more complex problems but also to produce more attractive programming environments. This proposal explores the concept that environments that integrate interpretative and compiled subsystems are an attractive way of combining productivity with high performance. We intend to use Web technology, especially Java, to build prototype systems which explore the interpreter/compiler integration for both sequential and parallel systems.

Our central idea is a front end compiler that produces intermediate code for what we call II/CVM (Integrated Interpretative/Compiled Virtual Machine) which can be implemented by either a high performance runtime or flexible interpretative mode. This research complements and builds on existing commercial activity as we focus on two areas - scientific computing and support of large scale data parallelism - which are outside the commercial mainstream. We have already initiated a community activity studying the use of Java in scientific and engineering computation which will help us make certain that our work is synergistic with and not competitive with commercial efforts. Qualitatively one can view our front end compiler as similar to the javac compiler's function of producing JavaVM bytecodes. The II/CVM will naturally need the study of such issues as Just in Time compilation, dynamic linking, remote loading, garbage collection, stack handling, communication latency hiding and the invocation of high performance pre-compiled runtime. Our research will lead to insights on the tradeoffs in placing optimizations either in the front end compiler or in the dynamic runtime supporting the new hybrid Virtual Machine.

This project builds on our previous research in HPF where we have produced a very robust public domain front end which we will modify to support the II/CVM front end compiler. Further our Darpa funded PCRC activity has produced parallel compiler runtime which can support Fortran, C++ and Java drivers and this will be a central part of the runtime support for the II/CVM. Note that languages such as HPF are natural for hybrid environments as they largely achieve parallelism by identifying (by either automatic methods or user specification) coarse grain parallel constructs such as array primitives and forall statements which are then implemented by efficient runtime. Such a coarse grain analysis can in the spirit of MATLAB or APL, be implemented with an interpreter as this would not have significant overhead. We in fact built a successful but not sustainable prototype HPF interpreter in 1993-94 which we intend to essentially generalize here. We see web technology now gives us appropriate infrastructure which we can use to build far more effective systems than was possible 3 years ago. In summary, the objectives of this research include:

- Identification of issues in optimal integration of interpretation and compilation techniques for parallel language systems.
- Study of the impact of the hybrid environments on performance for scientific and engineering applications.
- Development of a set of technologies that aim at a high performance language system with interpreted (Java) front end supported by compiled parallel runtime for solving scientific and engineering problems. Although not the central theme, this proposal will contribute to the understanding of the different ways of incorporating data parallelism in Java.

This proposal funds work at Syracuse University which is collaborative with the strong Compiler groups at Peking University and Harbin Institute of Technology whose local funding brings a major outside contribution to this proposed work.

B. Table of Contents

C. Project Description

C-1. Motivation of the proposed work

Tremendous advances in microprocessor performance and networking technology in the past decades has produced the following scenario: a personal computer today delivers higher performance than a mainframe could do 20 years ago; a vast sum of accessible computing power exists world wide.

Many consequences result from this scenario. Although the debate over compiler vs interpreter led to a focus for scientific computing in compilation about two decades ago, the fact that an interpreted program runs fast today than a compiled program ran 20 years ago suggests that the use of interpretation technology may be revisited. Besides the popularity of Perl, Tcl, and Python, etc., this evidence is best manifested by the overwhelming success of Java programming language, which is largely due to its underlying interpreted nature.

Nevertheless, it remains a fact that interpretation is slower than compilation-execution model within the same hardware technology. Thus, a trend is to merge (integrate) the two technologies, as exampled by JIT compiler in Java virtual machine, built-in computational functions in MATLAB, and as well as an earlier instance — interpreted front-end supported by data parallel runtime in *LISP on CM2. Parallel computing research has in fact shown the importance of relative coarse grain units such as array primitives in HPF and they are naturally supported by interpretive front ends.

Another important trend is the growing availability of excellent distributed computing software aimed at web applications — this includes Java language support as well as sophisticated servers such as Jeeves and Jigsaw. We have suggested that this provides good infrastructure on which to build scientific computing environments that combine user productivity and high performance execution [47]. As part of a set of studies of high end computing ten years from now (the so called Petaflop studies), we proposed the hybrid environment shown in Figure 1 which links a sophisticated interpreted problem solving environment to a machine specific optimized runtime [56]. In this project, we focus on a small part of this environment — what are the key issues raised by hybrid interpreted/compiled environments to support computational science.

We intend to base our implementations on adapting infrastructure developed for parallel Fortran (HPF) to a Java based environment. We are encouraged by the recent meetings (http://www.npac.syr.edu/projects/javaforcse) at Supercomputing'96 and Syracuse. There several papers suggested Java can offer a combination of an attractive object oriented language, high performance compilers, and interpreted applet mode.

However our research will focus on the general issues of hybrid (compiled and interpreted) environments using the Java based system as a particular exampler. As well as native compilers, we will also integrate Just in Time (JIT) concepts as they have shown already dramatic performance increases including more than an order of magnitude on the Linpack benchmark.

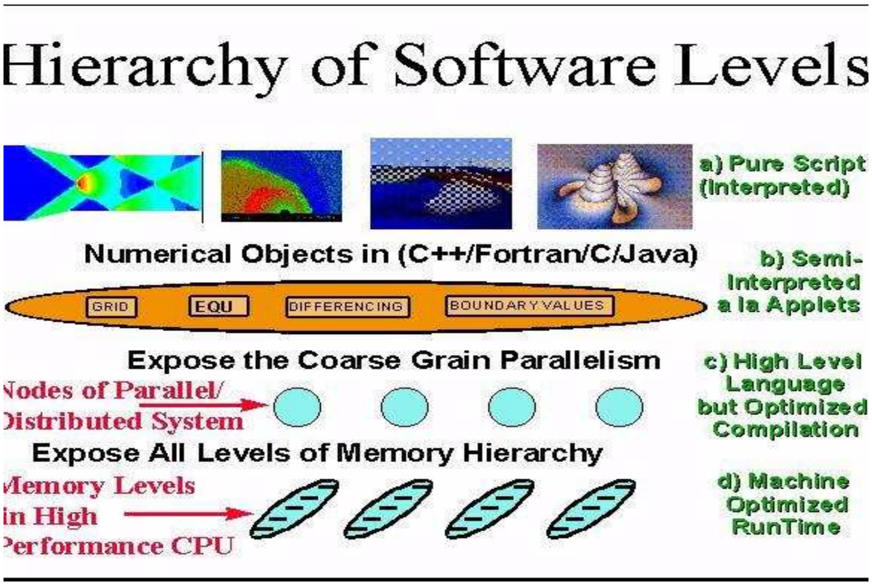

Figure 1: A Software Structure

C-2. Objective and expected significance

The objectives of this research include:

- Identify issues in optimal integration of interpretation and compilation techniques for parallel language systems.
- Study the impact of these issues on performance for scientific and engineering applications.
- Develop a set of technologies that aim at a high performance language system with interpreted (Java) front end supported by compiled parallel runtime for solving scientific and engineering problems.

The significance of the proposed activity may be observed in the following facets.

- While the effectiveness of interpreted languages on text processing tasks (or in general, $O(n)$ algorithms) is well accepted, how it can best help (or can not help at all) scientific and engineering applications, in the context of contemporary computer technologies, is not well understood.

While the higher performance of compiled programs over interpreted counterparts is well known, the inconvenience (and inefficiency !) of

`edit-compile-link-run-crash-start-it-all-over-again`

process is also commonly recognized.

This research addresses the potentials of bringing the benefit of both technologies together in a contemporary setting.

- Compilation and interpretation technologies are actually merging, as seen in JIT compiler and graphics runtime support in Java system, but this is happening in an *ad hoc* way, which does not consider the special constraints of parallel computing and science and engineering applications.
- While issues in parallel processing with compiled code have been studied extensively, not much is known as to how large scale parallel processing may benefit from an interpreted environment, which has obvious advantages for users.
- As we have described [16], Java naturally links (data parallel) time stepped simulations with large scale event driven object based systems.

C-3. Proposed Research and Experimental System

We intend to build our research on top of a set of experiments using an experimental system which will allow us to examine different ways of integrating compilation and interpretation techniques. We introduce, as in Figure 2, a new virtual machine (II/CVM or Integrated Intepreted/Compiled VM) which can be thought of as generalizing (to data parallelism) and focusing (to scientific computing) the successful Java VM. This hybrid virtual machine is fed by a *frame processor* or compiler (somewhat analogous to javac for the Java VM) called II/CVM compiler.

- The *frame processor* converts the application program into an intermediate form, and performs certain analysis, and possible restructuring, in addition to syntax and semantics checking. The result will be a partially ordered set of *interpretive frames.*
- Under supervision of the virtual machine/scheduler, the actions specified in a frame may be performed in one of the three ways.
  - Executed by an *interpretive module* directly.
  - Some precompiled *computational library* function is invoked locally to accomplish the task; this function may be executed sequentially or in parallel.
  - The frame is sent to some *registered remote system*, which will get the work done, once again either sequentially or in parallel.

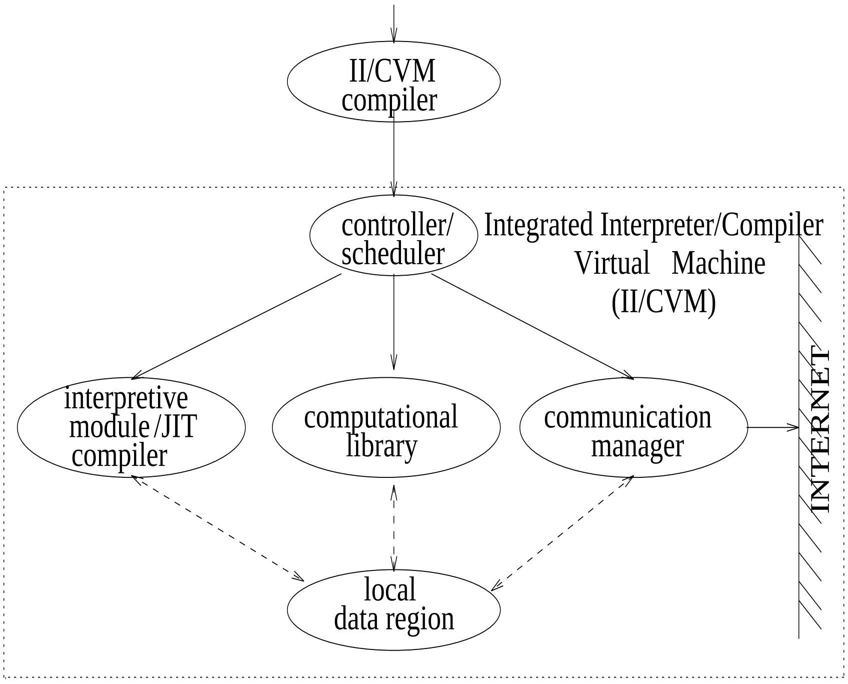

Figure 2: Architecture overview of a target system

- Necessary data flow coordination must be observed using appropriate mechanisms around the common *data region*.
- Relation between the interpretive frames and source code is well documented internally for diagnostic and debugging purpose. This will use web information technology including linked databases.

The distinction between *local processor* and *registered remote system* would be application dependent, a machine on the same local area network may be considered local or remote for different situations.

We distinguish fine grain parallelism (threads or data parallelism) within the local data region and typical coarse grain parallelism coming from metaproblems. As we are implementing in separate research (WebFlow), web technology can naturally implement coarse grain parallelism using networks of Java servers (Jeeves, Jigsaw) and emerging concepts such as Javabeans, [57], [16]. These issues are interesting but not the subject of this proposal.

C-3.1. Component descriptions

In what follows, we'll give a brief introduction to each of the components in the system. This introduction is not supposed to be a complete description of either the system or components. It serves as an outline on possible issues or techniques that one may encounter or employ when trying to construct such a system. It's also not our intention to deal with all the issues raised here in subsequent research, or try to incorporate all known techniques in a system. Rather, the research will be concentrated on identifying and evaluating those key issues and techniques which when integrated together properly constitute an optimal hybrid language system.

C-3.1.1. II/CVM compiler

The function of the II/CVM compiler (or frame processor) is to generate a partially ordered set of frames for the virtual machine to execute. It will be composed of the following parts:

- Language front-end
- Program analyzer/restructurer
- Interpretive frame generator
- Coordination interface

We propose the following features in each part.

Language front-end The language front-end is similar to that in a conventional compiler[39]. It will perform syntax and semantics checking on input programs, and construct an intermediate representation of the source code.

The intermediate representation may be some lower level language or even a set of highly structured data items, recording the information in the source program for efficient manipulation later in the frame processor. The design and implementation of the language front-end will make use of research and development results from the PCRC project conducted in NPAC, Syracuse University, in particular, the HPF front-end [40].

Program analyzer/restructurer The program analyzer/restructurer is used to obtain control flow and data flow information from the intermediate representation of the source program, and prepare things necessary for frame generator. Its main purpose is to *identify* and *organize* the parallelism in the program.

Both task parallelism and data parallelism may be exploited. While we are not emphasizing, in this effort, fine-tuned loop analysis techniques that aim at parallel execution of a loop, we'll concentrate on discovering coarse-grained task units that can be independently processed by either local or remote computational runtimes. Fine grain data parallelism will be handled by the runtime system.

Parallelism in program can be derived either explicitly or implicitly. For example, the explicit thread mechanism in Java language provides a simple yet effective way for expressing task parallelism, and an array of threads is effectively equivalent to the `FORALL` construct as in HPF. Recent progress in HPF project[41] helps us understand more about data parallel processing. An experimental work on "High Performance Java" conducted in NPAC has also provided us some insight on parallel execution of Java programs, [35].

On the other hand, techniques developed in parallelizing compiler, particularly inter-procedural and data flow and dependence analysis related methods[46], [44], [42], may be used to explore implicit parallelism among chunks of code segments.

Aside from this, pattern matching methods will be used to recognize some special frames which correspond to pre-compiled computational library functions, though maybe executed sequentially. This will be addressed more deeply in the section on the computational library.

Interpretive frame generator An *interpretive frame* is a machine-independent task description. We do not understand yet exactly how this will be implemented, but a typical form of it will be a type of data package associated with a name of function that has been implemented in the pre-compiled runtime.

The machine-independent feature is needed by the execution mechanism, since a frame may be executed somewhere else in Internet. Like Java bytecode, frames on different machine should be the same, so that the task can be dispatched onto different resources. Unlike Jave bytecode, the granulity of the frames may vary significantly. The execution expressed by a frame possibly ranges from a simple scalar assignment statement to a complicated computational procedure. This allows the potential power of pre-compiled computational library to be fully exploited.

Dependence relation among frames will be maintained in terms of a partially ordered set. As we described earlier, parallel activities are distributed over local interpretive module, local runtime library, and remote runtime libraries. The dependence relation is used by scheduler to dispatch frames at proper times.

Coordination interface The coordination interface refers to both the interface between the frame processor and scheduler, and the interface between the programmer and frame processor.

The simplest form of coordinating interface may be, just like Java language, an intermediate file, which is generated by the frame processor and then interpreted and debugged on the virtual machine.

A more ambitious design is that the user may have more interactive interface with the system. This may includes a programming editing environment, communications between the environment with the interpreter and the communication manager directly. Therefore it is possible for a more user-friendly environment that supports things like program tracing, error reporting or even letting user dynamically interfere with the system scheduler.

C-3.1.2. Scheduler

The scheduler module is aimed at management of the set of interpretive frames generated by the frame processor, determination of the execution modes of them [24], and coordination of the execution sequence. As mentioned earlier, a frame may be executed in one of the three distinct ways: locally interpreted by *interpretive module*, locally executed by an invocation to a *local runtime* function, or sent to some *registered remote system*, which has specific computational facilities, for execution. When doing the computation, the scheduler will monitor the execution of the interpretive frames, and return the results back to the II/CVM compiler, for the purpose of interactive interface to the users.

The execution by local runtime or remote runtime may be *sequential* or *in parallel*. We emphasis both because *sequential* execution in this context is also meaningful. For one thing, it's pre-compiled, which presumably delivers higher performance than interpretation. For another, some components of the problem, perhaps because of their small size, may not be suitable for parallel execution. This way, we still can make use of them.

Making use of remote computing resources on the Internet presents much more complicated problems. Many research and development projects are aimed at this direction. Indeed, with rapid growth of Internet, all the computer servers on the Internet form a super-parallel computer system around the globe. And each day, there are hundreds of computers added into the picture. Millions of connected computers collectively deliver a vast amount of CPU cycles every second, which is certainly underutilized nowadays. Effectively making use of this global computing ability for scientific computing is many people's dream [33].

However as we have emphasized, this proposal is aimed at the use of web infrastructure for the hybrid support of the fine grain synchronization and optimization issues for scientific environments. The scheduler through the communication manager will link to coarse grain computational systems that aim to support web based metacomputing. These we assume will be layered on top of the work we proposed here.

There are many interesting issues in this area, such as resource management, task migration, security, and fault tolerance, etc, where we will limit ourselves to a very simple strategy, since these issues are not our focus in this proposed effort.

More specifically, our resource management will be simply based on *registration*. A frame will only be sent to *registered remote server*. Further, the functionalities of those remote servers are limited to pre-compiled scientific and engineering libraries. The remote servers will not talk to each other. There will be no fault tolerance mechanism built in the system. They are assumed to be reliable.

Thus, a simple, static *resource bank* is adequate for the scheduler to determine where to dispath a frame.

C-3.1.3. Interpretive module

The interpretive module in the system is a functional unit to execute "small" frames.

Basically, three kinds of frames will be generated by the frame processor of the system. The scheduler will let them invoke remote computation resources, the computational library and the interpreter respectively. The interpretive module usually will handle the computation in the source program which can not be *formulated* as a call to runtime, either locally or remotely.

The implementation of this module is relatively easy. A set of procedures will be enough for the purpose. Given the availability and popularity of Java, we may just use the corresponding part in Java virtual machine as the interpretive module in our system.

C-3.1.4. Computational library

More generally, we may call it a *runtime library* with emphasis on CPU intensive computation. The purpose of this module is two-fold: efficient execution of compiled code as opposed to slower interpretation for *well formed* program segments; parallel execution of typical parallelizable subtasks in scientific and engineering applications.

As we know, library techniques have been very successful in supporting compiled programming languages. With existence of pre-compiled library functions, application developers can focus on the application problems and algorithms themselves.

Libraries, existing as part of runtime, are also playing a very important role in interpreted systems. This is best exampled by graphics capability built in Java virtual machine. As part of our Darpa PCRC activity, we are building Java native class interfaces to computational library we have built to support HPF compilation. We expect many more such optimized libraries to become available for the Java VM.

Over the past years, many different kinds of libraries have been provided for scientific computing. But most of them are developed for supporting compiled languages, namely, they exist in the form of object code to be linked with caller's object code. This is the traditional way of providing the library functions for computing. We will study the mechanisms that efficiently link interpreters with the libraries in our integrated system.

Another critical aspect of our computational library is parallel computing support. A parallel library should be a collection of high performance mathematical subroutines used by application programmers to solve large problems [34], [38]. From the II/CVM compiler, we will get a partially ordered set of interpretive frames. While the parallelism among different frames is seen by the scheduler, parallel library routines are aimed at exploiting parallelism within a frame.

Another issue or opportunity brought up by the idea of integration of interpreter and pre-compiled computational runtime support is that the routines in the library may not only be some well-known scientific function, such as solving a system of equations, etc. A routine may also correspond to some typical program segment pattern. Identifying and formulating those patterns will be part of our research agenda.

C-3.1.5. Communication manager

As we have observed, there are two kinds of parallelism in the system, task parallelism among different frames and possible data parallelism within a frame, each suitable for some specific parallel processing applications, respectively. When the scheduler module determines that there exists task parallelism in the interpretive frames, and that especially some frames are needed to be sent out to external systems, the communication manager module comes into the play. It will keep track of external resources and let the scheduler know when something is finished.

The communication manager should talk to a counterpart in registered remote server. As explained earlier, the communication manager is important for a complete architecture but will not be a major focus for this research.

C-3.1.6. Data region

The data region is a memory space in the virtual machine which holds the global data of the problem. It is initialized according to the problem size when the program started.

Since the global data may be changed by the interpretive module, the computational library and communication manager, synchronization method for these three parts is needed here. For simplicity, message passing method will be used and remote socket communication among process in UNIX system will serve this purpose.

Some data will need to be moved between the data region and remote memory locations in external system. Communication channels will be built up between Java virtual machine memory and other memory pages with message passing native classes. We have built an MPI interface to Java which can be used here.

C-3.2. A three year research plan

Year 1: Environment setup and experiment design.

> The goal is to reach a design of a set of experiments on key issues and techniques possibly involved in the hybrid language system as described above.

The completion of the design is signified by a detailed description of the experiments and establishment of an effective environment that allows the experiments to be conducted. Thus, with the architecture proposed above on mind, we'll work on collecting and testing various tools and infrastructure needed for experiment and issue investigation, with emphasis on integration of the tools. For instance, we may use Sun's Java front-end generator `JACK` to produce a front-end; we may use Stanford's `SUIF` package to serve as a basis for program analysis; and our own `PCRC` runtime may be used as a major component of the computation library, etc. Clearly, finding the most suitable tools and 'gluing' them together requires substantial research.

Year 2: Build initial system, conduct experiments and data analysis.

The goal is to obtain some meaningful quantitative conclusions on the impact of various issues and techniques on the performance of typical scientific and engineering applications.

The detailed agenda will depend on the first year's work. But in general we'll try to conduct the experiments along several different fronts. We will identify a few key benchmark problems of appropriate intermediate complexity which can be run with several different parameter choices. We can expect the tradeoff between interpretation and compilation to be very sensitive to problem size.

In this second year, we expect to have several modules of the system of Figure 2 operational but not yet a full integration.

Year 3: new idea testing and system integration.

The goal is to arrive at an operational system that demonstrates merits of integration of interpretation and compilation technologies. This system will be derived from the experimental environment, but it may or may not be of the same architecture or execution mechanism as we described in previous subsections. Obviously findings during the first two years of research may suggest new approaches that improve overall performance and usability.

C-3.3. International collaboration

Substantial collaboration among NPAC at Syracuse University, USA, and Computer Science Department of Peking University, China, and Computer Science Department of Harbin Institute of Technology, China will be a distinctive feature of our working plan. This is enabled by our successful collaborative activities in both research and education areas during past two years.

Since arrival of Dr. Xiaoming Li at NPAC as a visiting scholar in January, 1995, who then was a professor in Computer Science Department of Harbin Institute of Technology and now is a professor in Computer Science Department of Peking University, NPAC has been engaged in several significant and highly visible activities with those two universities and other Chinese institutes, including:

- In summer of 1995 and 1996, NPAC director, Geoffrey Fox and senior project manager Don Leskiw were invited to visit China. During these trips, they gave talks on various topics ranging from traditional HPCC and contemporary web technology in nine Chinese universities and research institutes, including Peking University, Tsinghua University, Beijing University of Aeronautics and Astronautics, Beijing University of Post and Telecommunication, Harbin Institute of Technology. Zhejiang University, Northwest University, Northwest Polytechnical University, and Institute of Computing Technology. These visits uncovered great interests in collaboration in basic research.

- Started from March and ended in July of 1996, NPAC conducted an Experimental Internet Based Computational Science Education Program. Five graduate students from Harbin Institute of Technology participated in the program. Course materials and instructions were delivered by an NPAC instructor to China via Internet on a weekly basis. Students also submitted their homework and raised questions back to NPAC via the Internet. This program caused a sensation in Chinese media. Almost all major newspapers and TV stations reported it. Upon completion, all students expressed very positive opinions about the program.

- Correspondingly, NPAC received a delegation of "Pan Deng" Programme (a national programme sponsored by Commission of Science and Technology of China) in December of 1995, a delegation from a key national software lab of China in February of 1996, Professor Zheng Weimin of Tsinghua University in August of 1996, and a delegation of 863 Programme in November of 1996.

- Also, coordinated by Dr. Xiaoming Li, a group in Harbin Institute of Technology, a group in Peking University, and NPAC Parallel Software Systems group have collaboratively developed a quality public domain HPF front-end, and a joint HPF compiler effort is ongoing.

From the above substantial activities, we have established a solid base for further collaboration with Chinese researchers. Inviting Dr. Xiaoming Li as an 'official' participant of this project, who will be visiting NPAC every summer, will strengthen this collaboration. In fact, a group of faculty members and graduate students has been formed in his Peking university and a similar project is being simultaneously proposed to Chinese NSF.

Graduated from Stevens Institute of Technology in 1986, Dr. Xiaoming Li is an active computer scientist in China. He has published three books, more than 40 technical papers, and led several sponsored projects, in parallel processing, computer architecture, artificial intelligence, as well as graph theory. Awarded many grants by various Chinese agencies, he is also a Fellow of Chinese Computer Federation and a member of Advisory Committee for Higher Education in Computer Science, appointed by the Education Commission of China. He also served as program committee chair of 1991 International Conference for Young Computer Scientists, Beijing, China, and program committee chair of the First National Symposium for Young Reliability Professionals, Qinhuang Dao, China.

We value the collaboration with Dr. Xiaoming Li, and believe his participation in this project adds significantly to its quality.

C-4. Related work

We'll first summarize other people's work in related areas such as *interpreter*, *compiler*, and *parallel runtime* in the context of Java programming language, followed by a review of the applicant's other related activities and research results.

Other related research topics

There was active research in the early 70's, as to whether a language would be best executed in interpreted or compiled form. People had well understood and agreed on the advantages and disadvantages of both. Reference [1] presents a typical analysis at that time. The basic conclusion was that interpreter was too slow to be widely useful.

As microprocessors and personal computers became popular in late 70's and early 80's, use of an interpreter was once again favored by some people and applications. Reference [2] and [4] promoted the use of interpreters and discussed techniques employed in construction of interpreters. Since then, more interpretive language systems have been built. Among them, there are the popular BASIC, Perl, etc., which are still in widespread use.

Nevertheless, the compilation-execution paradigm was still a main stream technology in language processing due to its continuously improved performance and introduction of tools and environments that help program development process. Relatively speaking, we have seen much more progress and advancement in compiler technology than in interpreter technology during past 15-20 years. This is evidenced by vast amount of literatures on compilers, in contrast to a little on interpreters during those years.

The popularization of World Wide Web and advent of Java programming language [9] have injected new energy into interpretation technology. Besides its traditional advantages, interpretation has also proved its value in cross-platform executability and security [10]. This development has already motivated some dedicated research in interpreters such as project Rocky being conducted in University of Washington [5].

There is substantial research on optimizing Java both in academia and industry, including industry leaders such as Sun and Microsoft. Many issues are being explored, we note

- Optimizing Java compilers

  IBM is developing optimizing Java compiler that produces binary code for IBM machine directly (reported by Susan Flynn Hummel of IBM at Workshop on Java for Scientific Computing, Dec., 1996)

- Just in Time compilers

  `Kaffe` system by Tim Wilkinson, http://www.tjwassoc.demon.co.uk/kaffe/faq.htm

- Optimization and restructuring of bytecode generated by *javac*

  Rice University, reported by Zoran Budimlic of Rice at the Workshop on Java for Scientific Computing, Dec., 1996.

  Rochester University, reported by Wei Li at the Workshop on Java for Scientific Computing, Dec., 1996.

- Parallel execution of Java program by making connection between Java and well established message passing systems

  `JavaPVM` from Georgia Tech. This is an interface written using Java native methods capability which allows Java applications to use PVM facility.

  http://www.isye.gatech.edu/chmsr/JavaPVM/

  `JPVM` from University of Virginia. This is a PVM-like library of object classes implemented in and for use with Java.

  http://www.cs.virginia.edu/ ajfzj/jpvm.html

  `HPJava` from Syracuse University. This is an experimental work demonstrated at Supercomputing'96, which links Java with MPI.

  http://www.npac.syr.edu/user/gcf/hpjava.html

- Java source restructuring

  `Javar` from Indiana University. This is a parallelizer for Java programs, see

  http://www.extreme.indiana.edu/hpjava/

- Parallel interpretation of bytecode

  IBM JVM, released in spring 1996, an implementation of JVM on shared memory architectures.

  UIUC, newly started a project aiming at parallel interpretation of Java bytecode (private communication.)

We believe that this other work confirms the importance of our proposed research activity. However we believe that we have identified many issues where substantial research is still needed and to which we can make meaningful contribution.

Applicant's related work

Fox has worked on parallel programming environments for the last 15 years with the early focus being message passing, [14], [12].

Since 1987 he has collaborated first with Ken Kennedy and then others, on parallel high level languages-especially Fortran. This work was largely part of Fox's activity in CRPC (The Center of Research in Parallel Computation) which played a major role in the development of HPF with prototype compilers and community language definition [15]. This led a major project from ARPA to develop in collaboration with Rice a prototype High Performance Fortran (Fortran90D) compiler, including a High Performance Fortran interpreter that was demonstrated at Supercomputing'93, [6], [13]. This early HPF compiler was licensed by the Portland Group whose resultant commercial product is quite highly regarded. The prototype HPF interpreter was our first study of some of the issues we wish to follow up in this proposal.

We believe Web technology is now advanced enough that we can actually build and study sophisticated interpreters (linked to compilers) for science and engineering computation which were prohibitively expensive in 1994 when we decided not to proceed further with our HPF interpreter. In fact the MOVIE system built by Furmanski and used as basis of HPF interpreter, lovingly constructed many of the capabilities now offered as pervasive robust technology by the Web (and especially Java).

Fox is now the PI of another major Darpa project, the parallel compiler runtime consortium (PCRC), which is developing a common runtime for high performance (data parallel) languages including Fortran C++ and Java. This project is a collaboration with Cooperating Systems, Florida, Harvard, Indiana, Maryland, Rochester, Rice and Texas.

Fox has recently focused on looking at the opportunities for using Web technologies to improve the infrastructure and functionality of scientific and engineering programming environments, [57], [16]. In particular , Fox was the Chair of Workshop on Java for Scientific Computing held in Syracuse, Dec., 1996, and he will be co-chairing the ACM 1997 Workshop on Java for Science and Engineering Computation to be held in Las Vegas, Nevada, June 21, 1997. Some of the ideas discussed here have come from discussion in the PetaFlop initiative where a set of meetings in which Fox played an active role, has discussed the software and hardware issues of possible parallel systems 10 years from now, [56]. Related research has built a Web based virtual programming laboratory which was used in a parallel computing class last semester and is being tried out in Cornell's Virtual Workshop at this time, [55]. This technology supports HPF and MPI in a user friendly environment and we believe that interpreted environments - as we propose to investigate here - are very promising for improving HPCC education and training.

References

[1] John A.N. Lee, The Anatomy of a Compiler.(2nd ed). Van Nostrand Reinhold Company, 27-38 (1974).

[2] Thomas C. McIntire, Software Interpreters for Microcomputers. John Wiley & Sons, New York, 1978.

[3] Patrick D. Terry, Programming Language Translation, A Practical Approach. Addison-Wesley, 1986.

[4] Paul Klint, Interpretation Techniques, Software — Practice and Experience, Vol. 11, 963-973(1981).

[5] Theodore H. Romer, Dennis Lee, Geoffrey M. Voelker, et al, The Structure and Performance of Interpreters, ASPLOS VII, 1996.

[6] HPFF, High Performance Fortran Language Specification (version 1.0). May 3, 1993.

[7] J. R. Bell, Threaded code, CACM, 16, 370-372 (1973).

[8] R. B. K. Dewar, Indirect threaded code, CACM, 18, 330-331 (1975).

[9] James Gosling, Bill Joy, and Guy Steele. The Java Language Specification. Addison-Wesley, Reading, Massachusetts, 1996, ISBN 0-201-63451-1.

[10] Ken Arnold and James Gosling. The Java Programming Language. Addison-Wesley, Reading, Massachusetts, 1996, ISBN 0-201-63455-4.

[11] Tim Lindholm, Frank Yellin. The Java Virtual Machine Specification. Addison-Wesley, Reading, Massachusetts, 1997, ISBN 0-201-63452-X.

[12] G. C. Fox, P. Messina, R. Williams. Parallel Computing Works. Morgan and Kaufman, 1994.

[13] G. C. Fox, W. Furmanski, P. Hornberger, J. Niemiec, and D. Simini, "Towards Interactive HPCC: High Performance Fortran Interpreter," Demo at the Supercomputing'93, Portland, Oregon, 1993.
http://www.npac.syr.edu/PROJECTS/PUB/wojtek/hpsin/doc/tvr.ps

[14] Fox, G.C., Johnson, M.A., Lyzenga, G.A., Otto, S.W., Salmon, J.K., Walker, D.W., Solving Problems on Concurrent Processors, Vol. 1, Prentice-Hall, Inc. 1988; Vol. 2, 1990.

[15] Fox, G.C., Hiranadani, S., Kennedy, K., Koelbel, C., Kremer, U., Tseng, C.W., Wu, M.Y., "FortranD Language Specifications", Rice COMP TR90079, December 1990, Revised, April 1991.

[16] Fox, G and Furmanski, W. "Computing on the Web – New Approaches to Parallel Processing – Petaop and Exaop Performance in the Year 2007", Submitted to IEEE Internet Computing, http://www.npac.syr.edu/users/gcf/petastuff/petaweb/

[17] W. Furmanski, "MOVIE — Multitasking Object-oriented Visual Interactive Environment," in G. C. Fox, P. Messina, and R. Williams, Parallel Computing Works. Morgan and Kaufman, 1994.

[18] Andrew S. Grimshaw, Jon B. Weissman, and W. Timothy Strayer," Portable Runtime Support for Dynamic Object-Oriented Parallel Processing," ACM Transactions on Computer Systems, Vol. 14(2), 139-170 (May 1996).

[19] P. J. Brown, Writing Interactive Compilers and Interpreters. John Wiley & Sons, New York, 1979, ISBN 0-471-27609-X.

[20] Harold Abelson and Gerald Jay Sussman, Structure and Interpretation of Computer Programs. MIT Press, 1985.

[21] Leonard Bolc (ed), The Design of Interpreters, Compilers, and Editors for Augmented Transition Networks. Springer-Verlag, 1983, ISBN 0-387-12789-5.

[22] Dr. Samuel H. Russ, Dr. Brian Flachs, Jonathan Robinson, and Bjorn Heckel, Hector: Automated Task Allocation for MPI, 10th International Parallel Processing Symposium, (IPPS 96)

[23] Jeremy Casas, Dan L. Clark, Ravi Konuru, Steve W. Otto, Robert M. Prouty, and Jonathan Walpole, MPVM: A Migration Transparent Version of PVM, Computing Systems, Vol. 8, no. 2, pp. 171-216, Spring 1995.

[24] Andrea C. Dusseau, Remzi H. Arpaci, David E. Culler, Effective Distributed Scheduling of Parallel Workloads, Sigmetrics'96, May 1996, Philadelphia, PA

[25] Casas, Jeremy, et al., "MIST: PVM with Transparent Migration and Checkpointing," Proceedings of the Third Annual PVM User's Group Meeting, Pittsburgh, PA, May 1995

[26] Neuman, B. Clifford, and Rao, Santosh, "The Prospero Resource Manager: A Scalable Framework for Processor Allocation in Distributed Systems", Concurrency: Practice and Experience, Vol. 6(4), June 1994, pp. 339-355.

[27] KONEN, Wolfgang and VORBRGGEN, Applying Dynamic Link Matching to Object Recognition in Real World Images. Proceedings of the International Conference on Artificial Neural Networks (ICANN), Ed. Stan, Gielen and Bert Kappen, Publ. North-Holland, Amsterdam, pp. 982-985.

[28] B. Schnor and H. Langendrfer and S. Petri: Einsatz neuronaler Netze zur Lastbalancierung in Work stationclustern. In Praxisorientierte Parallelverarbeitung, Ed. H. Langend rfer, Hanser, Mnchen, October 1994, pages 154-165 (in German).

[29] D. Wessels and J. Muzio, Analyzing and improving delay defect tolerance in pipelined combinational circuits, IEEE International Workshop on Defect and Fault Tolerance in VLSI Systems, 1995, pp. 181-188.

[30] Sukumar Ghosh, Arobinda Gupta, Ted Herman, and Sriram V. Pemmaraju, "Fault-Containing Self-Stabilizing Algorithms," Proceedings of the 15th ACM PODC, pp. 45-54, Philadelphia, 1996.

[31] PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing

[32] Message Passing Interface Forum, "MPI: A Message-Passing Interface Standard", June, 1995

[33] http://www-lmmb.ncifcrf.gov/EP96/, Use of the Internet for Scientific Communication and Databasing

[34] D.B. Carpenter, An Outline Proposal for the PCRC Library Interface, Technical Report, Northeast Parallel Architectures Center (NPAC) at Syracuse University, 1996

[35] D.B. Carpenter, Yuh-Jye Chang, et al, "Experiemnts with HPJava," Workshop on Java for Scientific and Engineering Applications, Syracuse, December, 1996.

[36] Brown, Perter. J., Writing Interactive compilers and Interpreters, A Wiley-Interscience Publication, 1979

[37] Leonard Bolc, The Design of Interpreters, Compilers, and Editors for Augmented Transition Net works, Springer-Verlag, 1983

[38] PCRC, Common Runtime Support for High Performance Data Parallel Languages, Project proposal, May, 1994.

[39] A. V. Aho, R. Sethi, and J. Ullman, "Compilers: Principles, Techniques, and Tools", Addison-Wesley, Reading, MA, second edition, 1986

[40] Xiaoming Li, Guansong Zhang, et al, "HPFfe: a Front-end for HPF", Technical Report, SCCS-771, NPAC at Syracuse University, 1996.4.

[41] Xiaoming Li, Runtime Oriented HPF Compilation. Technical Report, SCCS-773, NPAC at Syracuse University, 1997.1.

[42] U. Banerjee, R. Eigenmann, A. Nicolau and D. A. Padna, " Automatic Program Parallelization", Proceedings of the IEEE, vol, no.2, 1993

[43] M. Wolfe, "Optimizing Supercompilers for Supercomputers", Pitman Publishing, London, 1989

[44] H. Zima, and B. Chapman, "Supercompilers for Parallel and Vector Computers", Addison-Wesley Publishing Company, 1990

[45] S. Hiranandani, K. Kennedy, and C. Tseng, "Evaluating Compiler Optimizations for Fortran D", Journal of Parallel and Distributed Computing, vol.21, pp.27-45, 1994

[46] M. W. Hall, S. Hiranandani, K. Kennedy and C. Tseng, "Interprocedural Compilation of Fortran D for MIMD Distributed-Memory Machines", Supercomputing'92, 1992.

[47] Fox, Geoffrey C., Furmanski, Wojtek, Chen, Marina, Rebbi, Claudio, and Cowie, James H., "WebWork: Integrated Programming Environment Tools for National and Grand Challenges," Syracuse University Technical Report SCCS-715, June, 1995.

[48] Fox, Geoffrey C., and Furmanski, Wojtek, "The Use of the National Information Infrastructure and High Performance Computers in Industry," in Proceedings of the Second International Conference on Massively Parallel Processing using Optical Interconnections, IEEE Computer Society Press, Los Alamitos, CA, 298-312. Syracuse University Technical Report SCCS-732, October 1995.

[49] Fox, Geoffrey C., "High Performance Distributed Computing," Syracuse University Technical Report SCCS-750, December 1995. To appear in Encyclopedia of Computer Science and Technology.

[50] Fox, Geoffrey C., "An Application Perspective on High-Performance Computing and Communications," Syracuse University Technical Report SCCS-757, April 1996.

[51] Fox, Geoffrey C., "A Tale of Two Applications on the NII," Syracuse University Technical Report SCCS-756. In the proceedings of the 1996 Sixth Annual IEEE Dual-Use Technologies and Applications Conference, June 1996.

[52] Fox, Geoffrey C., and Wojtek, Furmanski, "SNAP, Crackle, WebWindows!", published as RCI Management White Paper Number 29, Syracuse University Technical Report SCCS-758, April 1996.

[53] Taubes, Gary, "Do-It-Yourself Supercomputers," Science, 274:5294, 1840, December 13, 1996.

[54] K. Dincer and G. C. Fox, "Building a World-Wide Virtual Machine Based on Web and HPCC Technologies", In the Proceedings of Supercomputing 96, Pittsburg, Penn., Nov. 17-22, 1996.

[55] Kivanc Dincer and Geoffrey C. Fox, "Using Java and JavaScript in the Virtual Programming Laboratory: A Web-Based Parallel Programming Environment," NPAC Technical Report, SCCS-780, January, 1997.

[56] Geoffrey C. Fox, "Software for HPCC Petaflops Architectures - A White Paper," NPAC Technical Report, December 18, 1996.

[57] Wojtek Furmanski and Geoffrey Fox, "Prototyping a WebVM based WebFlow Environment for Distributed Computing and Visual Programming," presented at the IEEE Dual-Use Technologies & Applications Conference, Syracuse, NY, June 1996.

E. Biographical sketch

Geoffrey Charles Fox

Address: 111 College Place, NPAC, SU, 13244 gcf@nova.npac.syr.edu , http://www.npac.syr.edu, Phone: (315) 443-2163, Fax: (315) 443-4741

Citizen Status: Permanent Resident Alien; Citizen of United Kingdom

Education: B.A. in Mathematics from Cambridge Univ., Cambridge, England (1961-1964) Ph.D. in Theoretical Physics from Cambridge University (1964-1967) M.A. from Cambridge University (1968)

Professional Experience:

1990- Professor of Computer Science, Syracuse University
1990- Professor of Physics, Syracuse University
1990- Director of Northeast Parallel Architectures Center
1979-1990 Professor of Physics, California Inst. of Tech.
1986-1988 Associate Provost for Computing, California Inst. of Tech.
1983-1985 Dean for Educational Computing, California Inst. of Tech.
1981-1983 Executive Officer of Physics, California Inst. of Tech.
1974-1979 Associate Professor of Physics, California Inst. of Tech.
1971-1974 Assistant Professor of Physics, California Inst. of Tech.
1970-1971 Millikan Research Fellow in Theoretical Physics, Caltech
1970 Visiting Scientist (April-May), Brookhaven National Laboratory
1969-1970 Research Fellow at Peterhouse College, Cavendish Lab.,Cambridge
1968-1969 Research Scientist, Lawrence Berkeley Lab., Berkeley, Calif.
1967-1968 Member of School of Natural Science, Inst. for Advanced Study, Princeton, New Jersey

Awards and Honors: Senior Wrangler, Part III Mathematics, Cambridge (1964) Alfred P. Sloan Foundation Fellowship (1973-75) Fellow of the American Physical Society (1990)

Journal Editorships:

Principal: Concurrency: Practice and Experience (John Wiley, Inc.) Physics and Computers (International Journal of Modern Physics C - World Scientific)

Associate: Journal of Supercomputing,

Selected List of Publications:

[1] Fox, G.C., Johnson, M.A., Lyzenga, G.A., Otto, S.W., Salmon, J.K., Walker, D.W., Solving Problems on Concurrent Processors, Vol. 1, Prentice-Hall, Inc. 1988; Vol. 2, 1990.

[2] Fox, G.C., Copty, N.,Ranka, S.,Shankar, R. "Solving the region growing problem on the Connection Machine," in Proceedings of the 22nd International Conference on Parallel Processing, volume 3, pages 102-105, 1993.

[3] Fox, G. C., Messina, P., Williams, R., Parallel Computing Works!, Morgan Kaufmann, San Mateo Ca, 1994.

[4] Fox G.C., Mills K., "InfoMall: an Innovative strategy for high-performance computing and communications application development", Internet Research, 4:31- 45, 1994.

[5] Fox, G.C., Hiranadani, S., Kennedy, K., Koelbel, C., Kremer, U., Tseng, C.W., Wu, M.Y., "FortranD Language Specifications", Rice COMP TR90079, December 1990, Revised, April 1991.

[6] Fox, G. C. "Approaches to Physical Optimization," in Proceedings of 5th SIAM Conference on Parallel Processes for Scientific Computation, pp 153-162, March 25-27, 1991, Houston, TX, J. Dongarra, K. Kennedy, P. Messina, D. Sorensen, R. Voigt, editors, SIAM, 1992. C3P-959, CRPC-TR91124

[7] Fox G.C., Mansour N., "Parallel Physical Optimization Algorithms for allocating data to multicomputer nodes", Journal of Supercomputing, 8:53-80,1994.

[8] Fox, G. C. "Parallel Computing and Education," Daedalus, Journal of the American Academy of Arts and Sciences, Vol. 121, No. 1, pps 111-118, Winter 1992. C3P-958, CRPC-TR91123.

[9] Fox, G, Bozkus, Z., Choudhary, A., Haupt, T., and Ranka, S. ""A compilation approach for Fortran 90D/HPF compilers on distributed memory MIMD computers," in Proceedings of the Sixth Annual Workshop on Languages and Compilers for Parallel Computing. Lecture Notes in Computer Science, Springer-Verlag, pp. 200–215. U. Banerjee, D. Gelernter, A. Nicolau, and D. Padua (editors).

[10] Fox, G and Furmanski, W. "Computing on the Web – New Approaches to Parallel Processing – Petaop and Exaop Performance in the Year 2007", Submitted to IEEE Internet Computing, http://www.npac.syr.edu/users/gcf/petastuff/petaweb/

Summary of Interests: Fox leads an significant project to develop prototype high performance Fortran compilers and their runtime support. He is also a leading proponent for the development of computational science as an academic discipline and a scientific method. He has established at Syracuse University both graduate and undergraduate programs which cover both simulation and information technologies. All course have been made available on the Web and his research includes HPCC technology to support education at both K-12 and University level. His research on parallel computing has focused on development and use of this technology to solve large scale computational problems. Fox directs InfoMall, which is focused on accelerating the introduction of high speed communications and parallel computing into New York State industry and developing the corresponding software and systems industry.

Ph.D Advisor: Dr. Richard Eden Cambridge University, England

Xiaoming Li

Address: Computer Science Dept., Peking University, Beijing 100871, People's Republic of China, (will be visiting NPAC and working on the project during summer time.)

Education: B.E. in Computer Science and Engineering from Harbin Institute of Technology, Harbin, China, (1978-1982). M.S. in Computer Science from Stevens Institute of Technology, Hoboken, USA, (1982-1983); Ph. D. in Computer Science from Stevens Institute of Technology, Hoboken, USA, (1983-1986).

Professional Experience:

1997-: Professor of Computer Science, Peking University

1995-1997: Visiting Research Scientist, Syracuse University

1989-1997: Professor of Computer Science, Harbin Institute of Technology

1988-1989: Postdoctoral researcher, Harbin Institute of Technology

1987: Assistant Professor, EECS Dept., Stevens Institute of Technology

1983-1986: Teaching Assistant, EECS Dept., Stevens Institute of Technology

1991: Program Committee Chair of 1991 International Conference for Young Computer Scientists, Beijing, China.

1994: Program Committee Chair of the First National Symposium for Young Reliability Professionals, Qinhuang Dao, China.

1991-: Member of Advisory Committee for Higher Education in Computer Science, appointed by Education Commission of China

1986-: Member of Eta Kappa Nu, IEEE CS, New York Academy of Sciences

Awards and Honors: Outstanding Teaching Assistant Award (Stevens Tech, 1986), Chou Peiyuan International Academic Exchange Award (Chinese Science and Technology Association, 1991), Fellow of Chinese Computer Federation (1992-).

Research Projects Accomplished (as PI):

Distributed Computing Architecture (1989-1992, Grant from the Ministry of Aeronautic and Astronautic Industry, China)

Reliable Network Topology (1990-1992, Grant from the Education Commission of China)

Multiprocessor Performance Evaluation Techniques (1991-1993, Grant from the Commission of National Defence Industry, China)

Data Parallel Processing Based on Networked Workstations (1994.1 – 1995.7, Grant from the Commission of National Defence Industry)

Processor Architecture for MPP systems (1993.9 – 1995.9, Grant from the Education Commission of China)

Selected List of Publications:

[1] Xiaoming Li, Runtime Oriented HPF Compilation. Technical Report, SCCS-779, NPAC at Syracuse University, 1997.1.

[2] Xiaoming Li, et al, HPFfe: a Front-end for HPF, Technical Report, SCCS-771, NPAC at Syracuse University, 1996.4.

[3] Xiaoming Li and Xu Cheng, Core Techniques and Primary Architectures of Contemporary Processors. Electronics Industry Publisher, Beijing, 1995.5

[4] Guansong Zhang, Xiaoming Li, "Barrier Synchronization and Pipeline Synchronization in Distributed Memory Machines", Science Bulletin, 1996

[5] Guansong Zhang, Xiaoming Li, "Predicting Execution Time of Parallel Program Based on Simulation Software", Chinese Journal on Computers., 1995

[6] Xiaoming Li, et al, "DADENT: A Cost-effective Environment for Developing Distributed Software," PROC. of the 15th COMPSAC, Sept. 1991, Tokyo Japan.

[7] Guansong Zhang, Xiaoming Li, "Analysis of Interconnection Functions Needed for a New Nonlinear Skewing Scheme", Chinese Journal on Computers., 1994

[8] F.T.Boesch, Xiaoming Li, and J. Rodriguez, "Graphs with most number of three point induced connected subgraphs," Discrete Applied Mathematics, Vol 59, No 1, 21 April 1995, pp. 1-10.

[9] F.T. Boesch, Xiaoming Li, and C. Suffle, "On the Existence of Uniformly Optimally Reliable Networks," NETWORKS, Vol.21, (1991) 181-194.

[10] Xiaoming Li and B. Fang, "An Optimal Algorithm for Optimal Dual-Ring Networks," J. of Computers, 1990.7

Ph.D. Advisees: Lijie Jin, "A Study in Adjustable Coupling Mechanism for Multicomputers", July 1992; Xu Cheng, "Speculative Execution and Multi-control Flow: a key to exploit instruction level parallelism", May 1994; Guansong Zhang, "Computation Partitioning on Workstation Clusters," June 1995; Hongwei Liu, "Scheduling in Multithreaded Architectures," June 1995.

Ph.D Advisor: Dr. F. T. Boesch, Stevens Institute of Technology, Hoboken, New Jersey.

Yuhong Wen

Address: 111 College Place, Northeast Parallel Architecture Center (NPAC), Syracuse University, Syracuse, NY, 13244

Education:

B.S., Department of Computer Science and technology, Tsinghua University, Beijing, P.R.China, (1985-1990).

M.S., Department of Computer Science and Technology, Tsinghua University, Beijing, P.R.China, (1990-1991).

Ph.D., Department of Computer Science and Technology, Tsinghua University, Beijing, P.R.China, (1991-1994).

Professional Experience:

June, 1996 – Present: Research Scientist at Northeast Parallel Architecture Center (NPAC), Syracuse University

Dec., 1994 – June, 1996: Assistant Professor at Tsinghua University

Sep., 1991 – Dec., 1994: Teaching Assistant, CS, Tsinghua University

Awards and Honors: Execellent Graduate Student Award (Tsinghua University, 1990, 1994)

Research Projects Participated:

Parallel Compiler Runtime Consortium (PCRC) Northeast Parallel Architecture Center (NPAC), Syracuse University (June, 1996 – Present)

High Performance Cluster Workstation System National 863 High-Technology Project Tsinghua University, P.R.China (1994 – June, 1996)

Parallel Programming Environment Research and Design National 863 High-Technology Project Tsinghua University, P.R.China (1992 – 1994)

PGR (Parallel Graph Reduction) Parallel Accelerator National 863 High-Technology Project Tsinghua University, P.R.China Won National Second Prize (1991 – 1992)

VLIW Architecture and Compiler Techniques National 863 High-Technology Project Tsinghua University, P.R.China (1990 – 1991)

Distributed Knowledge Base System Research and Design National 7th Five-year project Tsinghua University, P.R.China (1989 – 1990)

Selected List of Publications:

[1] Yuhong Wen, HPF Parallel Prefix / Suffix Intrinsic Functions, Technical Report, NPAC, SU, 1996.10

[2] Xiaoming Li and Yuhong Wen, Efficient Compilation of Forall Statement with Runtime Support, Technical Report, NPAC, SU, 1997.1

[3] Yuhong Wen, Dingxing Wang, et al, "An Optimal Processor Allocation Algorithm in Heterogeneous Network Computing of Workstation Cluster Systems", Chinese Journal of Computers, Vol. 19, No.3, pp. 161-167, Mar., 1996.

[4] Yuhong Wen, Dingxing Wang, et al, "A Communication Technology and Parallel Programming Methods Based on Message Passing", Computer Research and Development, vol. 33, No.3, pp. 211-216, Mar., 1996.

[5] Yuhong Wen, Weiming Zheng, et al, "Parallel Programming Environments and System Porting Method", Mini-Micro Systems, Vol. 17, No.1, PP 13-19, Jan. 1996.

[6] Yuhong Wen, Weiming Zheng, et al, "A Parallel Programming Environment Based on Message Passing", in the proceedings of ICPADS'94, Taiwan, Dec. 19-21, 1994.

[7] Yuhong Wen, et al, "A Parallel Programming Techniques Based on Message Passing", Mini-Micro Computer Systems, vol.16, no.5, 1995.

[8] Yuhong Wen, The Function and Design of Distributed Parallel Programming Environment", Internal report of Tsinghua University, Dec., 1992.

Ph.D Advisor: Prof. Dingxing Wang, Department of Computer Science and Technology, Tsinghua University, Beijing, P.R.China

---

Guansong Zhang

Address:

NPAC,Syracuse University, 111 College Place, Syracuse, NY, 13244, U.S.A.

Education:

B.E. in Computer Science and Engineering from Harbin Institute of Technology, Harbin, China, (1986-1990).

Ph. D. in Computer Science from Harbin Institute of Technology, Harbin, China, (1986-1990).

Professional Experience:

1995-: Research scientist, NPAC, Syracuse University

1992-1995: Teaching Assistant, CS Dept., Harbin Institute of Technology

Awards and Honors:

Anchongen scholarship for outstanding postgraduate student (HIT, 1995)

Research Projects Accomplished:

Data Parallel Processing Based on Networked Workstations (1994-1995, Grant from the Commission of National Defence Industry)

Multiprocessor Performance Evaluation Techniques (1991-1993, Grant from the Commission of National Defence Industry, China)

Distributed Computing Architecture (1989-1992, Grant from the Ministry of Aerospace, China)

Research Projects Working on:

Frontend system implementation and integration

HPF compiler and PCRC run time library

Selected List of Publications:

[1] Xiaoming Li, Guansong Zhang, et al, “HPFfe: a Front-end for HPF”, Technical Report, SCCS-771, NPAC at Syracuse University, 1996.4.

[2] Guansong Zhang, “Partationing Data Parallel Program for Workstation Cluster”, Ph.D thesis, Harbin Institute of Technology., 1995

[3] Guansong Zhang, Xiaoming Li, “Barrier Synchronization and Pipeline Synchronization in Distributed Memory Machines”, Science Bulletin, 1996

[4] Guansong Zhang, Xiaoming Li, “Predicting Execution Time of Parallel Program Based on Simulation Software”, Chinese Journal on Computers., 1995

[5] Guansong Zhang, Xiaoming Li, “Analysis of Interconnection Functions Needed for a New Nonlinear Skewing Scheme”, Chinese Journal on Computers., 1994

[6] Guansong Zhang, et al, “Software System for ABC-90jr., an Array Based Computer”, IEEE TENCON'93, 1993

Ph.D Advisor:

Dr. Xiaoming Li, Computer Science Department, Peking University, Beijing 100871, People's Republic of China.

H. Facilities, equipment, and other resources

Local Computer Facilities

NPAC has a substantial computer infrastructure, including networked workstations, Mac and PC equipment, and various parallel computers. Parallel and distributed platforms include a 12-node IBM SP2, an 8-node DEC Alphafarm interconnected with a Gigaswitch, an 8-node Sun UltraSPARC cluster which will be interconnected with an ATM network shortly. These clusters are shared by all NPAC users, but available in dedicated parallel mode on a reservation basis. An 8-CPU SGI machine with R10000 processors is also available, providing a shared-memory parallel platform for testing purposes.

NPAC has a system support group which handles day to day operation and maintenance of the computer facilities.

Office

NPAC provides adequate office facilities for all participants.

Major equipment

The computer systems described above constitute the major equipment for this effort.

Other resources

NPAC provides basic secretarial services, duplicating, and the like.