

**INFORMATION ABOUT PRINCIPAL INVESTIGATORS/PROJECT DIRECTORS(PI/PD) and
co-PRINCIPAL INVESTIGATORS/co-PROJECT DIRECTORS**

Submit only ONE copy of this form for each PI/PD and co-PI/PD identified on the proposal. The form(s) should be attached to the original proposal as specified in GPG Section II.B. **DO NOT INCLUDE THIS FORM WITH ANY OF THE OTHER COPIES OF YOUR PROPOSAL AS THIS MAY COMPRISE THE CONFIDENTIALITY OF THE INFORMATION.**

PI/PD Name: Bryan Carpenter

Gender: ☒ Male ☐ Female

Ethnicity: (Choose one response) ☐ Hispanic or Latino ☐ Not Hispanic or Latino

Race:
(Select one or more)

☐ American Indian or Alaska Native
☐ Asian
☐ Black or African American
☐ Native Hawaiian or Other Pacific Islander
☒ White

Disability Status:
(Select one or more)

☐ Hearing Impairment
☐ Visual Impairment
☐ Mobility/Orthopedic Impairment
☐ Other _____
☐ None

Citizenship: (Choose one) ☐ U.S. Citizen ☐ Permanent Resident ☒ Other non-U.S. Citizen

Check here if you do not wish to provide any or all of the above information (excluding PI/PD name): ☒

REQUIRED: Check here if you are currently serving (or have previously served) as a PI, co-PI or PD on any federally funded project ☐

Ethnicity Definition:

Hispanic or Latino. A person of Mexican, Puerto Rican, Cuban, South or Central American, or other Spanish culture or origin, regardless of race.

Race Definitions:

American Indian or Alaska Native. A person having origins in any of the original peoples of North and South America (including Central America), and who maintains tribal affiliation or community attachment.

Asian. A person having origins in any of the original peoples of the Far East, Southeast Asia, or the Indian subcontinent including, for example, Cambodia, China, India, Japan, Korea, Malaysia, Pakistan, the Philippine Islands, Thailand, and Vietnam.

Black or African American. A person having origins in any of the black racial groups of Africa.

Native Hawaiian or Other Pacific Islander. A person having origins in any of the original peoples of Hawaii, Guam, Samoa, or other Pacific Islands.

White. A person having origins in any of the original peoples of Europe, the Middle East, or North Africa.

WHY THIS INFORMATION IS BEING REQUESTED:

The Federal Government has a continuing commitment to monitor the operation of its review and award processes to identify and address any inequities based on gender, race, ethnicity, or disability of its proposed PIs/PDs. To gather information needed for this important tasks, the proposer should submit a single copy of this form for each identified PI/PD with each proposal. Submission of the requested information is voluntary and is not a precondition of award. However, information not submitted will seriously undermine the statistical validity, and therefore the usefulness, of information received from others. Any individual not wishing to submit some or all the information should check the box provided for this purpose. (The exceptions are the PI/PD name and the information about prior Federal support, the last question above.)

Collection of this information is authorized by the NSF Act of 1950, as amended, 42 U.S.C. 1861, et seq. Demographic data allows NSF to gauge whether our programs and other opportunities in science and technology are fairly reaching and benefiting everyone regardless of demographic category; to ensure that those in under-represented groups have the same knowledge of and access to programs and other research and educational opportunities; and to assess involvement of international investigators in work supported by NSF. The information may be disclosed to government contractors, experts, volunteers and researchers to complete assigned work; and to other government agencies in order to coordinate and assess programs. The information may be added to the Reviewer file and used to select potential candidates to serve as peer reviewers or advisory committee members. See Systems of Records, NSF-50, "Principal Investigator/Proposal File and Associated Records", 63 Federal Register 267 (January 5, 1998), and NSF-51, "Reviewer/Proposal File and Associated Records", 63 Federal Register 268 (January 5, 1998).

COVER SHEET FOR PROPOSAL TO THE NATIONAL SCIENCE FOUNDATION

PROGRAM ANNOUNCEMENT/SOLICITATION NO./CLOSING DATE/if not in response to a program announcement/solicitation enter NSF 00-2					FOR NSF USE ONLY	
NSF 99-167			02/16/00		NSF PROPOSAL NUMBER	
FOR CONSIDERATION BY NSF ORGANIZATION UNIT(S) (Indicate the most specific unit known, i.e. program, division, etc.)						
INFORMATION TECHNOLOGY RESEARCH						
DATE RECEIVED	NUMBER OF COPIES	DIVISION ASSIGNED	FUND CODE	DUNS# (Data Universal Numbering System)	FILE LOCATION	
				790877419		
EMPLOYER IDENTIFICATION NUMBER (EIN) OR TAXPAYER IDENTIFICATION NUMBER (TIN)		SHOW PREVIOUS AWARD NO. IF THIS IS <input type="checkbox"/> A RENEWAL <input type="checkbox"/> AN ACCOMPLISHMENT-BASED RENEWAL		IS THIS PROPOSAL BEING SUBMITTED TO ANOTHER FEDERAL AGENCY? YES <input type="checkbox"/> NO <input checked="" type="checkbox"/> IF YES, LIST ACRONYMS(S)		
596001138						
NAME OF ORGANIZATION TO WHICH AWARD SHOULD BE MADE			ADDRESS OF Awardee ORGANIZATION, INCLUDING 9 DIGIT ZIP CODE			
Florida State University			Florida State University			
AWARDEE ORGANIZATION CODE (IF KNOWN)			Tallahassee, FL. 32306			
0014894000						
NAME OF PERFORMING ORGANIZATION, IF DIFFERENT FROM ABOVE			ADDRESS OF PERFORMING ORGANIZATION, IF DIFFERENT, INCLUDING 9 DIGIT ZIP CODE			
PERFORMING ORGANIZATION CODE (IF KNOWN)						
IS Awardee ORGANIZATION (Check All That Apply) (See GPG II.D.1 For Definitions) <input type="checkbox"/> FOR-PROFIT ORGANIZATION <input type="checkbox"/> SMALL BUSINESS <input type="checkbox"/> MINORITY BUSINESS <input type="checkbox"/> WOMAN-OWNED BUSINESS						
TITLE OF PROPOSED PROJECT ITR/ACS: Integration of High Performance Message-based Programming Environments with Distributed Objects and the Web						
REQUESTED AMOUNT \$ 297,364		PROPOSED DURATION (1-60 MONTHS) 36 months		REQUESTED STARTING DATE 08/01/00		SHOW RELATED PREPROPOSAL NO., IF APPLICABLE
CHECK APPROPRIATE BOX(ES) IF THIS PROPOSAL INCLUDES ANY OF THE ITEMS LISTED BELOW						
<input type="checkbox"/> BEGINNING INVESTIGATOR (GPG 1.A.3) <input type="checkbox"/> VERTEBRATE ANIMALS (GPG II.D.12) IACUC App. Date _____						
<input type="checkbox"/> DISCLOSURE OF LOBBYING ACTIVITIES (GPG II.D.1) <input type="checkbox"/> HUMAN SUBJECTS (GPG II.D.12)						
<input type="checkbox"/> PROPRIETARY & PRIVILEGED INFORMATION (GPG II.D.10) Exemption Subsection _____ or IRB App. Date _____						
<input type="checkbox"/> NATIONAL ENVIRONMENTAL POLICY ACT (GPG II.D.10) <input type="checkbox"/> INTERNATIONAL COOPERATIVE ACTIVITIES: COUNTRY/COUNTRIES _____						
<input type="checkbox"/> HISTORIC PLACES (GPG II.D.10) <input type="checkbox"/> FACILITATION FOR SCIENTISTS/ENGINEERS WITH DISABILITIES (GPG V.G.)						
<input type="checkbox"/> SMALL GRANT FOR EXPLOR. RESEARCH (SGER) (GPG II.D.12) <input type="checkbox"/> RESEARCH OPPORTUNITY AWARD (GPG V.H)						
PI/PD DEPARTMENT Comp Sci and Information Tech			PI/PD POSTAL ADDRESS NPAC, 111 College Place			
PI/PD FAX NUMBER 315-443-1973			Syracuse University			
			Syracuse, NY 13244			
			United States			
NAMES (TYPED)	High Degree	Yr of Degree	Telephone Number	Electronic Mail Address		
PI/PD NAME Bryan Carpenter	Ph.D	1984	315-443-5068	dbc@npac.syr.edu		
CO-PI/PD						
CO-PI/PD						
CO-PI/PD						
CO-PI/PD						

CERTIFICATION PAGE

Certification for Principal Investigators and Co-Principal Investigators:

I certify to the best of my knowledge that:

- (1) the statements herein (excluding scientific hypotheses and scientific opinions) are true and complete, and
 (2) the text and graphics herein as well as any accompanying publications or other documents, unless otherwise indicated, are the original work of the signatories or individuals working under their supervision. I agree to accept responsibility for the scientific conduct of the project and to provide the required progress reports if an award is made as a result of this application.

I understand that the willful provision of false information or concealing a material fact in this proposal or any other communication submitted to NSF is a criminal offense (U.S.Code, Title 18, Section 1001).

Name (Typed)	Signature	Social Security No.*	Date
PI/PD Bryan Carpenter		SSNs are confidential and are not displayed *ON FASTLANE SUBMISSIONS*	
Co-PI/PD			
Co-PI/PD			
Co-PI/PD			
Co-PI/PD			

Certification for Authorized Organizational Representative or Individual Applicant:

By signing and submitting this proposal, the individual applicant or the authorized official of the applicant institution is: (1) certifying that statements made herein are true and complete to the best of his/her knowledge; and (2) agreeing to accept the obligation to comply with NSF award terms and conditions if an award is made as a result of this application. Further, the applicant is hereby providing certifications regarding Federal debt status, debarment and suspension, drug-free workplace, and lobbying activities (see below), as set forth in Grant Proposal Guide (GPG), NSF 00-2. Willful provision of false information in this application and its supporting documents or in reports required under an ensuring award is a criminal offense (U. S. Code, Title 18, Section 1001).

In addition, if the applicant institution employs more than fifty persons, the authorized official of the applicant institution is certifying that the institution has implemented a written and enforced conflict of interest policy that is consistent with the provisions of Grant Policy Manual Section 510; that to the best of his/her knowledge, all financial disclosures required by that conflict of interest policy have been made; and that all identified conflicts of interest will have been satisfactorily managed, reduced or eliminated prior to the institution's expenditure of any funds under the award, in accordance with the institution's conflict of interest policy. Conflict which cannot be satisfactorily managed, reduced or eliminated must be disclosed to NSF.

Debt and Debarment Certifications

(If answer "yes" to either, please provide explanation.)

Is the organization delinquent on any Federal debt?

Yes ☐

No ☒

Is the organization or its principals presently debarred, suspended, proposed for debarment, declared ineligible, or voluntarily excluded from covered transactions by any Federal department or agency?

Yes ☐

No ☒

Certification Regarding Lobbying

This certification is required for an award of a Federal contract, grant, or cooperative agreement exceeding \$100,000 and for an award of a Federal loan or a commitment providing for the United States to insure or guarantee a loan exceeding \$150,000.

Certification for Contracts, Grants, Loans and Cooperative Agreements

The undersigned certifies, to the best of his or her knowledge and belief, that:

(1) No federal appropriated funds have been paid or will be paid, by or on behalf of the undersigned, to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with the awarding of any federal contract, the making of any Federal grant, the making of any Federal loan, the entering into of any cooperative agreement, and the extension, continuation, renewal, amendment, or modification of any Federal contract, grant, loan, or cooperative agreement.

(2) If any funds other than Federal appropriated funds have been paid or will be paid to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with this Federal contract, grant, loan, or cooperative agreement, the undersigned shall complete and submit Standard Form-LLL, "Disclosure Form to Report Lobbying," in accordance with its instructions.

(3) The undersigned shall require that the language of this certification be included in the award documents for all subawards at all tiers including subcontracts, subgrants, and contracts under grants, loans, and cooperative agreements and that all subrecipients shall certify and disclose accordingly.

This certification is a material representation of fact upon which reliance was placed when this transaction was made or entered into. Submission of this certification is a prerequisite for making or entering into this transaction imposed by section 1352, title 31, U.S. Code. Any person who fails to file the required certification shall be subject to a civil penalty of not less than \$10,000 and not more than \$100,000 for each such failure.

AUTHORIZED ORGANIZATIONAL REPRESENTATIVE		SIGNATURE	DATE
NAME/TITLE (TYPED) Raymond E.Bye,Jr., Interim VP Rsrch			02/16/00
TELEPHONE NUMBER 850-644-5260	ELECTRONIC MAIL ADDRESS nsfaward@res.fsu.edu	FAX NUMBER 850-644-1464	

*SUBMISSION OF SOCIAL SECURITY NUMBERS IS VOLUNTARY AND WILL NOT AFFECT THE ORGANIZATION'S ELIGIBILITY FOR AN AWARD. HOWEVER, THEY ARE AN INTEGRAL PART OF THE INFORMATION SYSTEM AND ASSIST IN PROCESSING THE PROPOSAL. SSN SOLICITED UNDER NSF ACT OF 1950, AS AMENDED.

A. Summary

This proposal is concerned with enabling parallel computation in a world dominated by Internet technologies. On the hardware side these will include massively parallel engines designed and deployed as Internet servers. On the software side, they will include software developed in network-aware programming languages like Java—software engineered to survive in heterogeneous and very dynamic environments. We will be refining MPI-like programming models and APIs for high performance programming in Java—researching ways to get the fastest possible message passing from Java, and ways to exploit novel Java technologies like Jini to produce richer message-passing environments.

Java introduces implementation issues for message-passing APIs that do not occur in conventional scientific programming languages. We will investigate how to apply ideas from projects such as Jaguar and JaVIA to MPI-like APIs, to reduce the overheads of the Java Native Interface. We will integrate ideas on efficient object serialization from projects like KaRMI with MPI-specific methods for handling complex datatypes. Simply providing efficient MPI-like APIs for Java is not enough. The programming model must address features specific to distributed computing. To better support computing in volatile Internet environments, we will need features like dynamic spawning of process groups, parallel client/server interfaces, and new mechanisms for handling failures. Jini provides a natural framework for implementing these features. An important emphasis will be on researching synergies between parallel message-passing programming and Jini-like systems. By combining ideas from MPI with ideas from Jini we aim to create an environment that encourages scalable, fault-tolerant parallel computing. Finally, we will explore uses of Jini in a middle tier for initiating parallel MPI jobs, where the parallel programs may be written in languages other than Java.

TABLE OF CONTENTS

For font size and page formatting specifications, see GPG section II.C.

Section	Total No. of Pages in Section	Page No.* (Optional)*
Cover Sheet (NSF Form 1207 - Submit Page 2 with original proposal only)		
A Project Summary (not to exceed 1 page)	1	_____
B Table of Contents (NSF Form 1359)	1	_____
C Project Description (including Results from Prior NSF Support) (not to exceed 15 pages) (Exceed only if allowed by a specific program announcement/solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)	15	_____
D References Cited	3	_____
E Biographical Sketches (Not to exceed 2 pages each)	4	_____
F Budget (NSF Form 1030, including up to 3 pages of budget justification)	5	_____
G Current and Pending Support (NSF Form 1239)	1	_____
H Facilities, Equipment and Other Resources (NSF Form 1363)	1	_____
I Special Information/Supplementary Documentation	2	_____
J Appendix (List below.) (Include only if allowed by a specific program announcement/ solicitation or if approved in advance by the appropriate NSF Assistant Director or designee)	_____	_____
Appendix Items:		

*Proposers may select any numbering mechanism for the proposal, however, the entire proposal must be paginated. Complete both columns only if the proposal is numbered consecutively.

C. Project Description

C.1 Introduction

This proposal is especially concerned with enabling parallel computation, and the associated communications, in a world dominated by Internet technologies. We will not assume that parallel computation is necessarily distributed *across* the Internet, although of course this is one possibility that has been widely discussed in the name of metacomputing. We do assume that the software and hardware technologies that will be readily available in the immediate future—the commodity technologies—will be fine-tuned for the Internet environment. On the hardware side these will include parallel—perhaps massively parallel—engines designed and deployed as Internet servers. On the software side, they will include software developed in network-aware programming languages like Java—software engineered to survive in heterogeneous and very dynamic environments.

Over the last few years the prospect of using Java for essentially “scientific” computing has become increasingly realistic. Ongoing activities in the *Java Grande Forum*—complemented by work in academic and industrial sectors on optimizing compilers, JITs, language enhancements and libraries—have helped close an initial credibility gap. It is increasingly accepted that Java environments will meet the performance constraints needed to support large-scale computations and simulations. The work on improving the performance of Java is driven largely by its industrial application as a programming language for high-performance Internet servers. Scientific programmers will also reap the benefits.

One of the most influential developments in parallel computing over the last decade was the publication in 1994 of the Message Passing Interface (MPI) standard [22]. The idea of an agreed standard API for communication in parallel programs was relatively slow in coming. As a result it benefitted from a great deal of accumulated experience from application developers using earlier, proprietary APIs. MPI supports the Single Program Multiple Data (SPMD) model of parallel computing, providing many modes of reliable point-to-point communication, and a library of true collective operations. An extended standard, MPI 2, incorporates additional features like dynamic process creation and one-sided access to memory in remote processes.

The MPI standards specify language bindings for Fortran, C and C++. None of these languages is especially adapted to the Internet, where downloadable and mobile code are norms, resources (including computational resources) may be discovered and lost spontaneously, and fault tolerance is a crucial issue. In the proposed work we will be concerned with using network-oriented languages for high-performance computing. For now this means Java. One immediate preoccupation is with refinement of MPI-like programming models and APIs for high performance programming in Java—researching ways to get the fastest possible message passing from Java, and ways to exploit novel Java technologies like Jini to produce richer message-passing environments. A complementary concern is with use of Jini in a middle tier between client and MPI-based parallel services.

Java introduces implementation issues for message-passing APIs that do not occur in conventional scientific programming languages. One area of research is how to transfer

data between the Java program and the network while reducing overheads of the Java Native Interface. We will investigate how to apply ideas from projects like Jaguar [35] and JaVIA [9] to MPI-like APIs. Another important issue is how to minimize the overheads of serialization in communicating Java objects and multidimensional arrays. We will integrate ideas on efficient object serialization from the KaRMI project [29], for example, with MPI-specific ideas we started to explore in [7]. We will be especially interested in supporting efficient communication of the scientific Array classes supported by the Java Grande Numerics Working Group.

Just providing efficient MPI-like APIs for Java is not enough. The programming model must address features specific to distributed computing. MPI 1 was designed with relatively static platforms in mind. To better support computing in volatile Internet environments, we will need (at least) features from MPI 2 like dynamic spawning of process groups and parallel client/server interfaces. A natural framework for dynamically discovering new compute resources and establishing connections between running programs already exists in Sun's Jini project. In the proposed work, an important emphasis will be on researching synergies between parallel message-passing programming and Jini-like systems. One defining characteristic of distributed computing is the presence of *partial failures*. By combining ideas from MPI with ideas from Jini we aim to create an environment that encourages scalable, fault-tolerant parallel computing.

Finally we will explore uses of Jini in a middle tier for initiating parallel MPI jobs, where parallel program may be written in Java, or some other language that uses MPI-style message passing.

C.2 Motivation of the proposed work

C.2.1 Parallel computing and Java

To realize its full potential parallel computing will have to adapt itself to the Internet environment by embracing current Internet technologies. Many people accept that the Java language and accompanying technologies are likely to continue as major influences on the development of Internet software. But the idea that Java should also be adopted as an important language for large-scale technical computations has met some resistance. The most serious objection has been the perceived *inefficiency* of the Java language when compared with more mature languages like Fortran.

Over the last three years supporters of the *Java Grande Forum* have been working actively to address some of the difficulties. The official goal of the forum has been to develop consensus and recommendations on possible enhancements to the Java language and associated Java standards, for large-scale ("Grande") applications. Through a series of ACM-supported workshops and conferences the forum has also helped stimulate research on Java compilers and systems, and helped lay to rest some of the doubts about the potential performance of Java systems. An interesting series of papers from IBM [24, 25, 37], for example, confirmed that the current generation of Java virtual machines have rather poor performance on Fortran-like, array-intensive computations, but went on to demonstrate how to apply aggressive optimizations in Java compilers to obtain performance competitive with Fortran. In a recent paper [26] they described a case study involving a data mining

application that used the Java Array package supported by the Java Grande Numerics Working Group. Using the experimental IBM HPCJ Java compiler they reported obtaining over 90% of the performance of Fortran.

The Java Grande Forum also has a Concurrency and Applications Working Group. This group has been looking directly at uses of Java in parallel and distributed computing. Participants have studied various approaches, and we will refer to several of these in the following sections. The proposed will be particularly investigating issues relating to adoption of message-passing parallel programming in Java.

C.2.2 Niches for parallel Java programs

Computers that host major Web sites will either be multiprocessors or clusters of workstations. Many are now, and this trend will presumably continue. Increasingly these servers are programmed in Java. Since these technologies—Java and parallel computers—will co-exist in Internet servers, this is fertile place to see roles for Java-based parallel computation emerging. Along with all the other commercial services that will appear on the Web in the near future we are likely to see compute-intensive ones—maybe supporting data-mining queries using parallel algorithms or financial analysis programs involving complex simulations.

Truly scalable servers are likely to be clusters rather than symmetric multiprocessors. As a specific example, consider the *Ninja* vision of the future of the Internet elaborated by researchers at UC Berkeley [32]. In their view a service should be *scalable* (able to support thousands of concurrent users), *fault-tolerant* (able to mask faults in the underlying server hardware), and highly-available. A major concern is with mobile code for service deployment—specialized active proxies that migrate out across the Internet to position themselves close to client devices. But services must maintain persistent state, and the architects of Ninja conclude that distributed, wide-area management of this state is generally intractable. “Hard”, persistent state is maintained in a carefully-controlled environment—the *Base*—engineered to provide high availability and scalability. This is assumed to be a cluster of workstations with fast, local communication, a controlled environment, and a single administrative domain [16]. It is not necessarily homogeneous and it is not completely reliable, so it is not exactly a conventional parallel computer. But this is an environment where we might expect message-passing parallel programs written in Java to appear, either to implement specific Internet services or just because scientific programmers exploit them on account of their availability—parallel Internet servers become commodity hardware.

A completely different place where we might see early uptake Java-based parallel computing is in the classroom. Java has become an important teaching language in Universities. For teaching parallel computing principles to students, Java is likely to be a more attractive language than Fortran. On the basis of project descriptions given when people download our *mpiJava* software, for example, we estimate that perhaps 10% of potential users are teachers looking for classroom software. This is not a dominant proportion, but it is an especially influential one so far as future uptake is concerned. In this context highly tuned implementations are not essential. An MPI-like package that is portable and can be installed easily on available networks of PCs is probably ideal.

The last niche for we will discuss for Java message-passing is perhaps the most obvious.

Because of its platform independence, mobility, and other associations with the Internet, Java is a natural candidate as a language for *metacomputing*. We interpret this to mean computation by parallel programs distributed across the Internet itself. Within the MPI community there is an ongoing effort to extend MPI specifications and implementations to support metacomputing, by allowing logical process groups to span geographically separated clusters and supercomputers. For example, an MPI interoperability standardization effort led by the National Institute of Standards and Technology [18] proposes a cross-implementation protocol for MPI—*Interoperable MPI* or IMPI—to support heterogeneous parallel computing. Java-based metacomputing can exploit and supplement these ongoing MPI activities in various ways. It may be, for example, that only a parallel *sub*-component of a distributed application is particularly suited to implementation in Java. If the Java part is programmed in an MPI-like paradigm the option is open for the Java component to interact with the non-Java, MPI-based part through the inherently parallel IMPI protocols (rather than, say, through a serial, performance-limiting CORBA or RMI gateway).

Many authors have discussed Java approaches to metacomputing, but they have generally emphasized different aspects of Java. Charlotte [5, 6] and Javelin [10, 28] concentrate on harvesting cycles of computers running Web browsers by downloading *applets* to them—a paradigm well-suited to task-farming but not particularly appropriate for applications that need communication between concurrent tasks. JavaParty [29, 31] and Manta [33] support an interacting SPMD style of distributed programming, but emphasize communication through remote method invocation. They provide ways to program with remote objects that are more transparent than the standard RMI interface, together with highly-tuned reimplementations of RMI. This work is clearly important, but it remains uncertain whether remote method invocation is the best model of communication for parallel computing. The message-passing model of synchronization seems a better fit to the requirements. Although not directly relevant to our goals here, we note that even in the distributed computing community there appears to be some movement towards message-passing models. According to [36], one of the lessons of Ninja 1.0 was that RMI was not the best model for their purposes—asynchronous typed message-passing would be better. Hewlett Packard’s *e-speak* architecture [17] adopts message-passing as the underlying model of communication with services.

C.2.3 Jini

Jini is Sun’s Java architecture for making services available over a network. It is built on top of the Java Remote Method Invocation (RMI) mechanism. The main additional features are a set of protocols and basic services for “spontaneous” discovery of new services, and a framework for detecting and handling *partial failures* in the distributed environment.

A Jini lookup service is typically discovered through multicast on a well-known port. The discovered registry is a unified first point of contact for all kinds of device, service, and client on the network. Aside from the initial act of discovery, all Jini-related operations are built on RMI. The Jini model of discovery and lookup is distinct from the more global concept of discovery in, say, the CORBA trading services or HP’s *e-speak*. The Jini version is a lightweight protocol, especially suitable for initial binding of clients and services within multicast range. In the Ninja framework, for example, Jini technology might fit comfortably

at the periphery, near the end-user devices, or *within* the Base, addressing initial federation of nodes, crashes of individual nodes, etc. This latter setting is particularly interesting to us.

The ideas of Jini run deeper than the lookup services. Jini completes a vision of *distributed programming* started by RMI. In this vision *partial failure* is a defining characteristic, distinguishing distributed programming from the textbook discipline of concurrent programming [34]. The principles of concurrent programming are integrated in the Java language and the JVM through support for threads and monitors. But mechanisms that are appropriate within a single JVM must be replaced by more complex techniques when multiple JVMs are federated over a network. Remote objects and RMI replace ordinary Java objects and methods; garbage collection for recovery of memory is replaced by a *leasing* model for recovery of distributed resources; the events of AWT or JavaBeans are replaced by the distributed events of Jini; the synchronized methods of Java are mirrored in the nested transactions of the Jini model. The interesting question arises of whether analogous ideas can be adopted to extend conventional *parallel* programming models.

C.2.4 Bringing these things together

To support the parallel programmers of the future we will need Java implementations of lightweight messaging systems akin to MPI—the single most successful model for parallel computing. A likely physical setting is in the more or less tightly coupled (but probably heterogeneous, multi-user) clusters of trusted workstations that we expect will host the Web services of the future. While models of distributed programming other than message-passing (notably Linda-based models like JavaSpaces or JavaNOW) certainly have a role, we question whether they are the best model for SPMD computing. Most of the experience with earlier generations of parallel computer suggests that the low-latency message-passing model is a better fit.

These are likely to be volatile environments that demand the reliability provided by foundations like Java and Jini. Any software must be adaptive. Availability changes as workloads and network traffic fluctuates; nodes crash, new ones are attached and discovered on the fly, old ones are removed. Jini is a leading Java technology for dealing with these situations. Message-passing parallel programming is not exactly the same discipline as concurrent programming. An interesting research question is whether one can develop a distributed model of parallel programming that extends the conventional MPI model in a manner similar to the way the Jini model extends concurrent programming. At the very least, as discussed in section C.4.3, Jini technology can be exploited to help implement the conventional MPI model in a distributed environment.

C.3 General plan of work

There are two principal strands in the proposed work. One strand will investigate use of Java and specifically *Jini* technologies in a middle tier for initiating parallel MPI jobs. For definiteness we refer to the architecture as *JiniMPI*. So far as the architecture itself is concerned the parallel program could be written in Java or some other language that invokes an MPI-style message layer. The second strand will be researching issues related to

JiniMPI Architecture

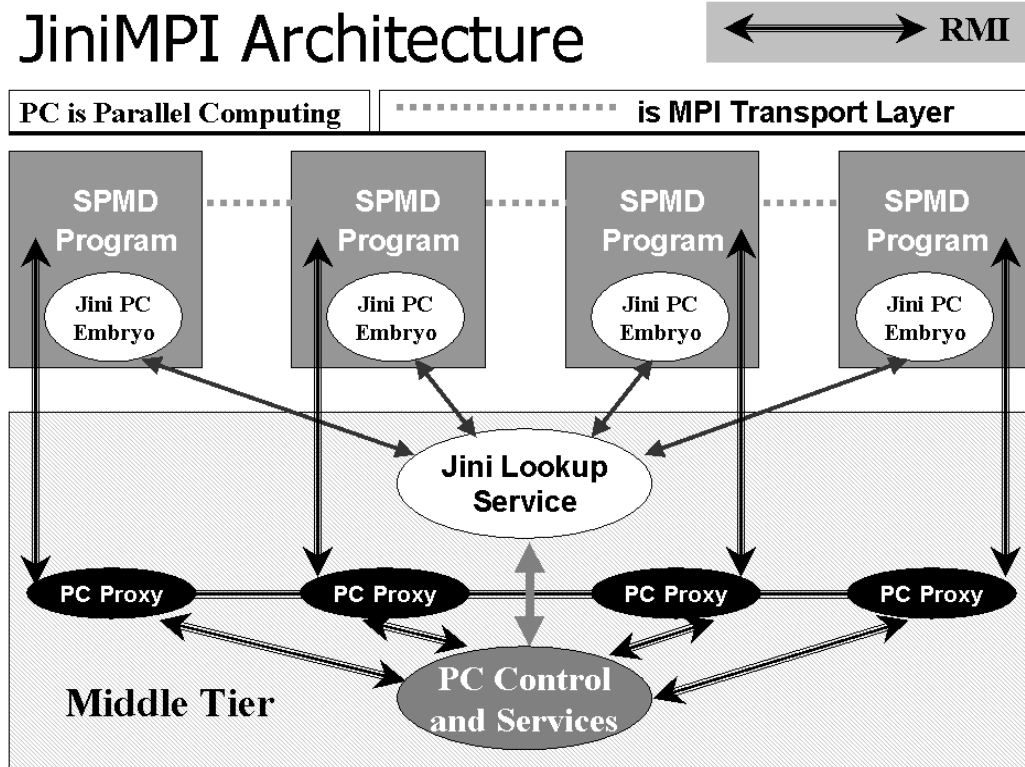


Figure C.1: JiniMPI Architecture

message-passing programs written *in* Java (and potentially other object-oriented network programming languages). It will study principles of reliably and efficiently implementing associated APIs in dynamic environments like Internet servers and networks. The concrete APIs will be derivatives of the *MPJ* specification from the Java Grande Message-Passing Working Group (section C.4.2).

In principal the *JiniMPI* and *MPJ* levels are independent, but they complement each other and are expected to be especially powerful when used together.

C.3.1 Jini-based middle tier for parallel programming

A possible JiniMPI architecture is illustrated in Figure C.1. The architecture supports MPI-based parallel computing and also includes ideas present in systems like Condor and Javelin. The diagram only shows the server layer (bottom) and the service layer (top). There would also a client layer that communicates directly with “Control and Services” module.

We assume that each workstation has a “Jini Parallel Computing Embryo”—a Jini service that registers the availability of the workstation to run either specific or generic applications. The Jini embryo can represent the machine—the ability to run general applications—or particular software. The Gateway, or Control and Services module [1], queries the Jini lookup services to find appropriate computers to run a particular MPI job. The mechanism could be used just to be run a single job, or to set up a farm of independent workers

The standard Jini mechanism is applied for each chosen embryo. This effectively establishes an RMI link from the Gateway to each SPMD node. It creates a Java proxy for the node program, which can itself be written any language (Java, Fortran, C++ etc). The Gateway-Embryo exchange should also supply to the Gateway with any needed data (such as specification of required parameters and how to input them) on behalf of the client layer. This strategy separates control from data transfer. It supports Jini (registration, lookup and invocation), services such as load balancing, and fault tolerance, in the control layer, and MPI style data messages in a fast transport layer. The Jini embryo is only used to initiate the process. It is not involved in the actual execution phase.

We will also investigate the implications of using a JavaSpace in the Control layer as the basis for a management environment (this is very different from using Linda or JavaSpaces at the execution level, so performance problems are not very relevant).

C.3.2 Research on high-performance message passing models for Java

In the early stages of the project we will complete a reference implementation of the *MPJ* specification, an MPI-inspired Java API from the Java Grande Message-Passing Working Group (section C.4.2). Our existing *mpiJava* software (section C.4.1) has been one basis for this work, but for further research a portable, *pure Java* version will be needed. Section C.4.3 describes a design. One of the conclusions of the design study was that difficult issues of reliability (and useability) in a network environment are naturally addressed in the framework of the Jini programming model. Our initial reference implementation will make extensive use of Jini for job initiation and handling failures. This implementation will be a foundation for subsequent research in the project.

The current MPJ specification supports essentially MPI-1.0 functionality, with some extensions specific to object-oriented languages (for example it has the facility to send and receive arbitrary serializable objects). With the reference implementation in place, the project will follow two directions

- Research into optimizations specific to the language and network context, to improve bandwidth and latency.
- Design and pilot implementation of extensions to the basic message-passing model, improving support for highly dynamic environments.

Associated tasks are detailed in the following two subsections.

Fast message-passing for Java

The initial reference implementation will use Java sockets as the basic transport. Later work will research different approaches to low-level transport—including calling native MPI by standard JNI or other methods, or using Java bindings to lower-level interfaces like VIA. It is also likely to involve work on improving the efficiency of object serialization, or exploiting the research of other groups on efficient object serialization.

Our earlier work on *mpiJava* already exploited the Java Native Interface (JNI) to call native MPI implementations. But there are concerns with the efficiency of crossing the JNI

barrier. For example, several of the current generation JITs have garbage collectors that do not support pinning of objects. A consequence is that arrays are copied every time they are passed between C and Java. More detailed criticisms of JNI can be found in [35] and [9]. The two groups involved have described ways to go beyond standard JNI, specifically to support efficient Java interfaces to VIA. Our research will attempt to leverage this work, either by adapting similar techniques to make low-overhead interfaces to native MPI, or (for suitable platforms) adopting Java VIA interfaces as a low-level transport in our Java implementation of the message-passing API (along lines comparable with [11]). Note that the new approaches typically assume limited changes to compiler or JVM, at least in the garbage collector.

Another area that needs further research is specific to object-oriented languages. To allow fast communication of *object graphs* between processors we need very efficient object serialization. Important work on improving Java serialization has been described in [29]. The authors report that their UKA-Serialization can save 76% to 96% of the time needed to serialize objects. Their work was done in the context of an optimized reimplementaion of RMI, but we can use the same software in fast MPJ implementations. We hope to combine ideas from UKA-Serialization with the MPI-specific ideas we started to explore in [7], to facilitate fast communication of objects in MPJ, especially for the important case of multidimensional Java arrays (if native methods are allowed, *one*-dimensional arrays of primitive elements can be communicated with no serialization at all, at least in the common case where sender and receiver have the same number representation and endianness).

Extending the MPJ message-passing model

As noted above, the initial MPJ draft specifies functionality similar to MPI 1.1, complemented by some object-oriented features.

One set of extensions to this draft will be inspired by features of the MPI 2 standard. This standard is not yet widely implemented even for traditional languages, but it includes features that are likely to be important in the volatile Java environments we target. Dynamic process creation was not part of MPI 1 but it is undoubtedly important for our environments. An early addition to the baseline MPJ model will be operations similar to `MPI_COMM_SPAWN` and `MPI_COMM_SPAWN_MULTIPLE`, which in MPI 2 start new groups of processes and return intercommunicators connecting them to the initial group. A related form of process creation is familiar from systems like PVM. A new feature in our research will be adoption of Jini or similar current technologies for discovery of the required computational resources. Another highly relevant feature of the MPI 2 specification is its introduction of a parallel client/server model, by which two running parallel programs (client and server) that do not initially share a communicator can establish a connection, and thus operate collectively as a single parallel program for some period. This kind of functionality is likely to be important in metacomputing applications. It introduces issues of how the initial rendezvous between the client and server occurs. MPI 2 specifies a simple mechanism based on entry points `MPI_PUBLISH_NAME`, `MPI_LOOKUP_NAME`, which are optional in MPI implementations. Clearly a Java environment will be able to offer much more flexible mechanisms modelled on Jini lookup.

Dynamic discovery of compute resources is one area where Jini-like ideas can help us.

A more difficult issue for scalable computing in networked environments is how to deal with dynamic *loss* of resources, due to a failure somewhere in the distributed platform. As emphasized in Section C.2.3, the general Jini framework incorporates various mechanisms designed to support programming in the presence of partial failure. One goal of our research will be to explore ways to incorporate similar concepts in the parallel message-passing context, enabling programs that are truly scalable even in the presence of partial failure. As a simple example, suppose we provided a collective checkpointing operation that dumped the current state of a parallel program to backing store, perhaps by serializing the replicated objects supporting the “main program” method (this scenario begs several questions of detail, but these can be addressed later). Invoking this collective operation at regular intervals would go some way to improving the reliability of a parallel program, by allowing restarts. But if a partial failure occurs during the checkpointing operation itself, the backed-up state will be corrupted. A scalable version of the Jini *transaction* model may be what is needed to rescue this kind of situation. This is one easy example, but it suggests that we should find roles for Jini-like ideas at the level of the SPMD program. Most likely these would be applied at the level of large, collective operations, such as global checkpointing, forking groups of slave processes for some subtask, and so on.

C.3.3 A three year workplan

Year one: In the first year we will complete ongoing work on a Jini-based pure Java implementation of MPJ—a message-passing environment with functionality similar to MPI 1.1. This implementation will be an important foundation for the subsequent research in the project. To support the later research, the transport layer must be easily replacable (similar to KaRMI, for example). This implies a layer analogous to the MPICH device level, but we will need new features to support Java and objects. Jini will be used for discovering compute hosts and (importantly) to ensure clean global termination in the event of failures. To prove the design, a native MPI-based implementation of the transport layer will also be implemented. This will complement the initial socket-based implementation, and will use standard JNI. (Our existing *mpiJava* software will probably be phased out.) In this year we will also be studying issues relating to concrete design of the JiniMPI Architecture, using Jini technologies as a gateway to parallel computing resources.

Year two: The basic message passing model will be extended with a version of dynamic process spawning using Jini to find resources, and a parallel client/server model, using Jini to establish connections between running parallel programs. Suitable Java-centric APIs will be designed. Relevant application codes from areas like parallel data-mining will be produced. We will study ideas following on from the Jaguar and JAVIA work at Berkeley and Cornell, and try to make use of those ideas to optimize our initial MPJ implementation, the goal being to develop genuine high-bandwidth, low-latency message-passing in Java. We will integrate UKA-serialization or successors, and further study MPI-specific improvements for important cases such as multidimensional arrays. Extensions needed to support efficient communication of the scientific Array classes supported by Java Grande will be a priority. Pilot implementations of JiniMPI architecture will be developed. We will investigate reliable checkpointing primitives for parallel programming, using Jini

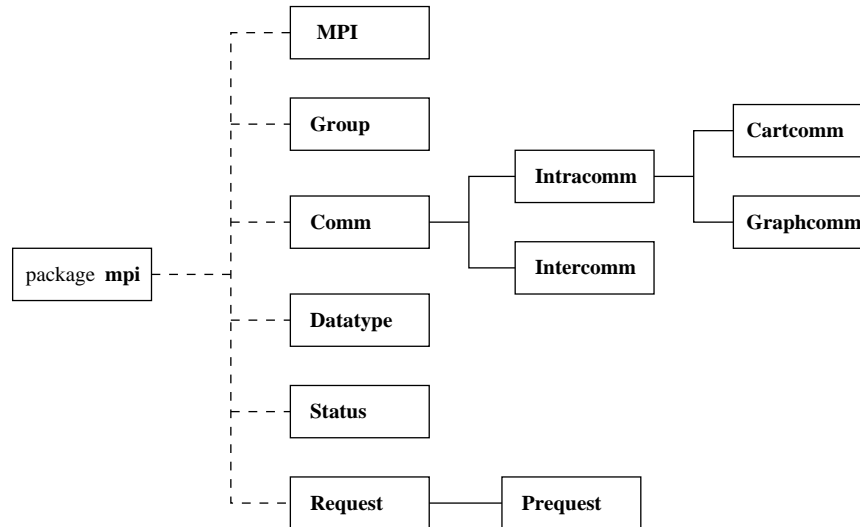


Figure C.2: Principal classes of mpiJava

transactions, or related mechanisms.

Year three: The experience of the first two years will feed into a practical model of scalable Internet computing, combining parallel-programming ideas from MPI with Jini ideas about fault tolerance. This model will be integrated with the three-tier JiniMPI model.

C.4 Related Work

C.4.1 Experience with *mpiJava*

mpiJava [4, 7, 27] is our object-oriented Java interface to MPI. The system provides a fully-featured Java binding of MPI 1.1 standard. The object-oriented API is modelled largely on the C++ binding that appeared in the MPI 2 standard. The implementation of *mpiJava* is through JNI (Java Native Interface) wrappers to a suitable native implementation of MPI. The software comes with a comprehensive test-suite translated from the IBM test-suite for the C version of MPI. Platforms currently supported include Solaris using MPICH or SunHPC-MPI, Linux using MPICH, and Windows NT using WMPI 1.1.

The MPI standard is explicitly object-based. The C and Fortran bindings rely on “opaque objects” that can be manipulated only by acquiring object handles from constructor functions, and passing the handles to suitable functions in the library. The C++ binding specified in the MPI 2 standard collects these objects into suitable class hierarchies and defines most of the library functions as class member functions. The *mpiJava* API follows this model, lifting the structure of its class hierarchy directly from the C++ binding. The major classes of *mpiJava* are illustrated in Figure C.2.

The benchmarks in Figure C.3 compare *mpiJava* (“J”) timings with native C timings for communication between a pair of PCs. The timings represent two different native MPI implementations (MPICH and WMPI), and also compare with with raw Windows sockets.

Bandwidth (Log) versus Message Length

(In Distributed Memory mode)

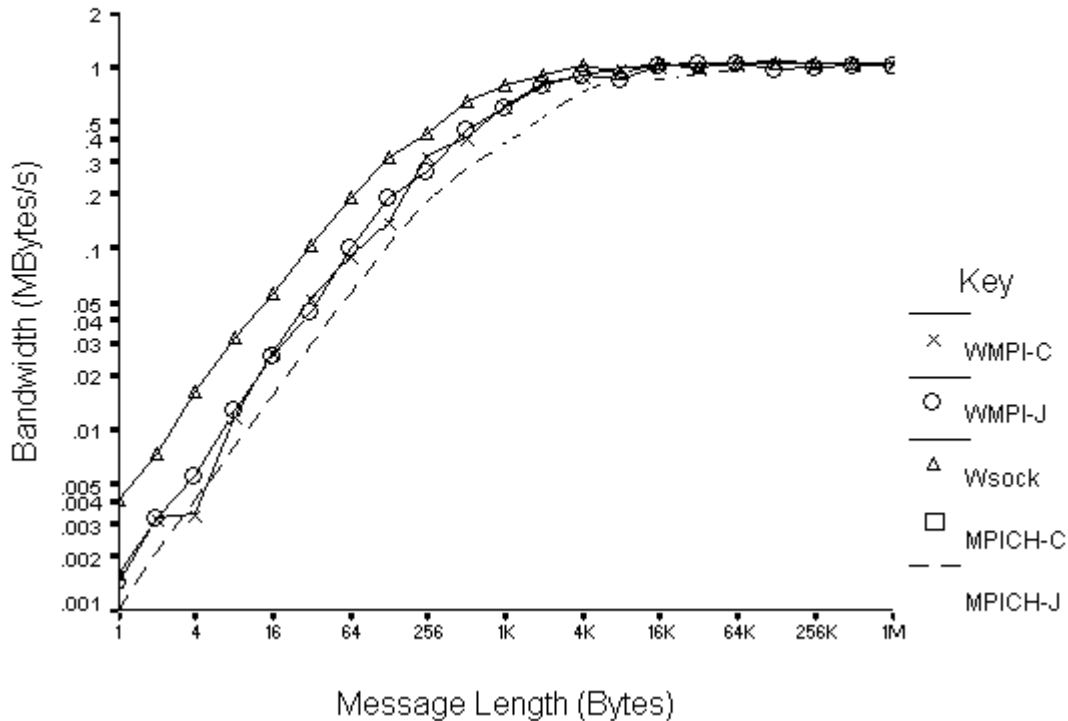


Figure C.3: PingPong Results in Distributed Memory mode

We see that the mpiJava JNI wrappers introduce a modest extra latency relative to native MPI, but for large messages bandwidth is not compromised. Although these results are encouraging, we should remark that these benchmarks were run using the Classic JVM. Current JIT compilers will degrade bandwidth because Java arrays are usually copied when they are passed to native methods. How best to avoid such overheads is an important research question [9, 35]. Also, these results are obtained for one-dimensional arrays with no serialization applied.

mpiJava is part of the *HPJava* environment [30]. We are actively developing and supporting the software. Downloads currently run at about 30 per month.

C.4.2 Java Grande Message-passing Working group

Java bindings to MPI were developed independently by several teams. One Java MPI interface was produced by Getov and Mintchev [15, 23]. In their work Java wrappers were automatically generated from the C MPI header. This eased the implementation work, but did not lead to a fully object-oriented API. A subset of MPI was implemented in the DOGMA system for Java-based parallel programming [21]. Dincer and Kadriy described an instrumented Java interface to MPI called *jmpi* [12]. Java implementations of the related PVM message-passing environment have been reported in [38] and [14].

The Message-Passing Working Group of the Java Grande Forum was formed just over

a year ago as a response to the appearance of these diverse APIs. An immediate goal was to discuss a common API for MPI-like Java libraries. An initial draft for a common API specification was distributed at Supercomputing '98 [8]. Since then the working group has met in San Francisco and Syracuse, and a Birds of a Feather meeting was held at Supercomputing '99. Minutes of meetings were published on the *java-mpi* mailing list and are available at [19, 20]. To avoid confusion with standards published by the original MPI Forum (which is not presently convening) the nascent API is now called *MPJ*.

C.4.3 Case study: reference implementation of MPJ

Presently there is no complete implementation of the draft MPJ specification. Our own Java message-passing interface, *mpiJava*, is moving towards the “standard”. The new version 1.2 of the software supports direct communication of objects via object serialization, which is an important step towards implementing the specification in [8].

The *mpiJava* wrappers rely on the availability of a platform-specific native MPI implementation for the target computer. While this is a reasonable basis in many cases, the approach has some disadvantages. For one thing the two-stage installation procedure—get and build a native MPI then install and match Java wrappers—can be tedious and discouraging to potential users. Secondly, in the development of *mpiJava* we sometimes saw conflicts between the JVM environment and the native MPI runtime behaviour. The situation has improved, and *mpiJava* now runs with several combinations of JVM and MPI implementation, but some problems remain. Finally, this strategy simply conflicts with the ethos of Java, where pure-Java, write-once-run-anywhere software is the order of the day.

Ideally, the first two problems would be addressed by the providers of the original native MPI package. We envisage that they could provide a Java interface bundled with their C and Fortran bindings. Ultimately, such packages would presumably be the best, industrial-strength implementations of systems like MPJ. Meanwhile, to address the last shortcoming listed above, we have outlined in [3] a design for a *pure-Java* reference implementation for MPJ. Design goals were that the system should be as easy to install on distributed systems as we can reasonably make it, and that it be sufficiently robust to be useable in an Internet environment. A particularly strong requirement is that in no circumstances should the software leave resource-wasting orphan processes lingering after an abrupt termination.

We are by no means the first people to consider implementing MPI-like functionality in pure Java. Working systems have already been reported in [12, 21], for example. Our goal was to build on some lessons learnt in those earlier systems, and produce software that is standalone, easy-to-use, robust, and fully implements the specification of [8].

We wish to simplify installation of message-passing software to a bare minimum. A user should download a jar-file of MPJ library classes to machines that may host parallel jobs, and run a parameterless installation script on each. Thereafter parallel java codes can be compiled on any host in the LAN (or subnet). An `mpjrun` program invoked on the development host transparently loads all the user’s class files to available compute hosts, and the parallel job starts. The only *required* parameters for the `mpjrun` program should be the class name for the application’s main program and the number of processors the application is to run on.

To be usable, an MPJ implementation should be fault-tolerant in at least the following

High Level MPI	Collective operations
	Process topologies
Base Level MPI	All point-to-point modes
	Groups
	Communicators
	Datatypes
MPJ Device Level	isend, irecv, waitany, . . .
	Physical process ids (no groups)
	Contexts and tags (no communicators)
	Byte vector data
Java Socket and Thread APIs	All-to-all TCP connections
	Input handler threads.
	Synchronized methods, wait, notify
Process Creation and Monitoring	MPJ service daemon
	Lookup, leasing, distributed events (Jini)
	exec java MPJSlave
	Serializable objects, RMIClassLoader

Figure C.4: Layers of proposed MPJ reference implementation

senses. If a remote host is lost during execution, either because a network connection breaks or the host system goes down, or for some other reason, *all* processes associated with affected MPJ jobs must shut down within some short interval of time. On the other hand, unless it is explicitly killed or its host system goes down altogether, the MPJ *daemon* on a remote host should survive unexpected termination of any particular MPJ job. Concurrent tasks associated with other MPJ jobs should be unaffected, even if they were initiated by the same daemon.

The paper design suggests that Jini is a natural foundation for meeting these requirements. The installation script can start a daemon on the local machine by registering a persistent activatable object with the `rmid` daemon. The MPJ daemons automatically advertise their presence through the Jini lookup services. The Jini paradigms of leasing and distributed events are used to detect failures and reclaim resources in the event of failure. These observations lead us to believe that an initial reference implementation of MPJ should probably use Jini technology [2, 13] to facilitate location of remote MPJ daemons and to provide a framework for the required fault-tolerance.

A possible architecture is sketched in Figure C.4. The base layer—process creation and monitoring—incorporates initial negotiation with the MPJ daemon, and low-level services provided by this daemon, including clean termination and routing of output streams (Figure C.5). The daemon invokes the `MPJSlave` class in a new JVM. `MPJSlave` is responsible for downloading the user’s application and starting that application. It may also directly invoke routines to initialize the message-passing layer. Overall, what this bottom layer provides to the next layer is a reliable group of processes with user code installed. It

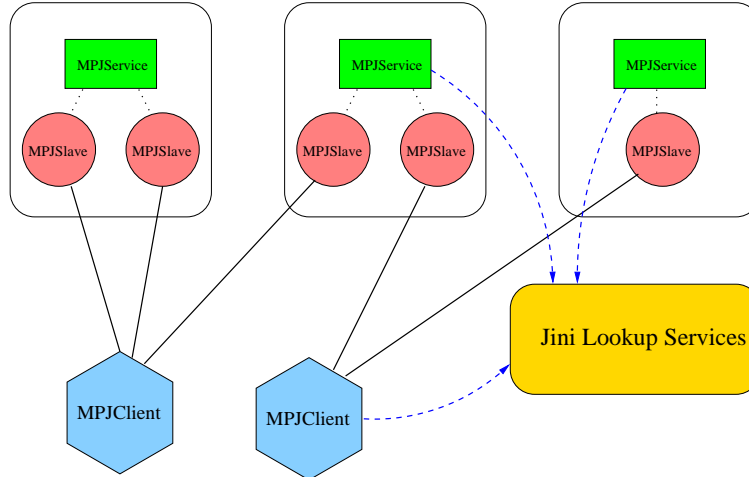


Figure C.5: Independent clients may find **MPJService** daemons through the Jini lookup service. Each daemon may spawn several slaves.

may also provide some mechanisms—presumably RMI-based (we assume that the whole of the bottom layer is built on RMI)—for global synchronization and broadcasting simple information like server port numbers.

Higher layers use Java sockets directly for efficient communication. The first manages low-level socket connections, establishing all-to-all TCP socket connections between the hosts. The idea of an “MPJ device” layer is inspired by the abstract device interface of MPICH. A minimal API includes non-blocking standard-mode send and receive operations. Other point-to-point communication modes are implemented with reasonable efficiency on top of this minimal set. The device level itself is meant to be implemented on socket `send` and `recv` operations, using standard Java threads and synchronization methods to achieve its richer semantics. The next layer above this is base-level MPJ, which includes point-to-point communications, communicators, groups, datatypes and environmental management. On top of this are higher-level MPJ operations including the collective operations. We anticipate that much of this code can be implemented by fairly direct transcription of the `src` subdirectories in the MPICH release—the parts of the MPICH implementation above the abstract device level.

C.4.4 Results from prior NSF support

Work on this proposal is related to previous NSF awards including the research grant described below from the Division of Advanced Computational Infrastructure and Research. This project is ongoing and will transfer from Syracuse to Florida State. Fox is PI.

Grant: 9872125, total award \$346,827 over period 09/01/98-08/31/01, “Data Parallel SPMD Programming Models from Fortran to Java”.

This involves senior personnel Fox and Carpenter, who with two students on the project will be moving from Syracuse to Florida State. The project focuses on the use of Java for *data parallel* programming but the methods are applicable to other languages. Collaborator Professor Xiaoming Li from Peking University is investigating applications to traditional scientific languages—especially Fortran. We have published several papers on this subject where the details are described. The HPJava model is less ambitious than systems like High Performance Fortran (HPF) and aims to support an SPMD model intermediate between basic message passing (MPI) and HPF. One can incorporate pure MPI code but also array based computation with automatic decomposition with a user specified mapping in the spirit of HPF. An essential capability is unified support of successful data parallel libraries like ScaLAPACK, PetSC, Kelp, Global Array Toolkit, PARTI/CHAOS and Adlib. So far we have developed an operational HPJava translator and linked to Global Arrays and Adlib. As part of our collaboration we have prepared and given in China a tutorial on HPJava and related approaches (<http://www.npac.syr.edu/projects/pcpc/HPJava/beijing.html>). To support MPI work from within HPJava programs we developed the MPI Java binding *mpi-Java* which is available for download from (<http://www.npac.syr.edu/projects/pcrc/HPJava/mpiJava.html>), and will be form one of the foundations of the work described in the current proposal. It is a reference implementation used by the Java Grande Message Passing Working Group. Fox organized the Java Grande Forum (<http://www.javagrande.org>) to address all the issues connected with the use of Java in scientific computing and both the working groups and associated conferences (now sponsored by ACM) have been quite successful. HPJava ideas have greatly benefitted from contacts in this arena. As well as publications given below, Sung Hoon Ko will complete his Ph.D. in this area during this semester.

Select publications

B. Carpenter, G. Zhang, G. Fox, X. Li, X. Li and Y. Wen “Towards a Java environment for SPMD programming”, *4th International Euromar Conference*, Springer, 1998.

G. Zhang, B. Carpenter, G. Fox, X. Li and Y. Wen, “The HPspmd Model and its Java Binding.”, chapter in book, R. Buyya ed, *High Performance Cluster Computing*, Vol 2, Prentice Hall 1999.

M. Baker, B. Carpenter, G. Fox, S.H. Ko and S. Lim, “mpiJava: An Object-oriented Java Interface to MPI”, *Intl. Workshop on Java for Parallel and Distributed Computing, IPSP/SPDP ’99*, San Juan, Puerto Rico, April 1999.

B. Carpenter, G. Fox, S.H. Ko and S. Lim, “Object Serialization for Marshalling Data in a Java Interface to MPI”, *ACM 1999 Java Grande Conference*, ACM Press 1999.

D. Bibliography

- [1] Erol Akarsu, Geoffrey Fox, Tomasz Haupt, Alexey Kalinichenko, Kang-Seok Kim, Praveen Sheethaalnath, and Choon-Han Youn. Using the Gateway system to provide desktop access to high performance computational resources. In *HPDC-8*. IEEE Press, August 1999.
- [2] Ken Arnold, Bryan O’Sullivan, Robert W. Scheifler, Jim Waldo, and Ann Wollrath. *The Jini Specification*. Addison Wesley, 1999.
- [3] Mark Baker and Bryan Carpenter. MPJ: A proposed Java message-passing API and environment for high performance computing. In *International Workshop on Java for Parallel and Distributed Computing*, Cancun, Mexico, May 2000. To be presented.
- [4] Mark Baker, Bryan Carpenter, Geoffrey Fox, Sung Hoon Ko, and Xinying Li. mpiJava: A Java interface to MPI. In *First UK Workshop on Java for High Performance Network Computing, Europar ’98*, September 1998. <http://www.cs.cf.ac.uk/hpjworkshop/>.
- [5] A. Baratloo, M. Karaul, H. Karl, and Z. M. Kedem. An infrastructure for network computing with Java applets. *Concurrency: Practice and Experience*, 10(11-13), 1998.
- [6] A. Baratloo, M. Karaul, Z. M. Kedem, and P. Wyckoff. Charlotte: Metacomputing on the Web. In *9th International Conference on Parallel and Distributed Computing Systems*, Dijon, France, September 1998.
- [7] Bryan Carpenter, Geoffrey Fox, Sung Hoon Ko, and Sang Lim. Object serialization for marshalling data in a Java interface to MPI. In *ACM 1999 Java Grande Conference*. ACM Press, June 1999.
- [8] Bryan Carpenter, Vladimir Getov, Glenn Judd, Tony Skjellum, and Geoffrey Fox. MPI for Java: Position document and draft API specification. Technical Report JGF-TR-3, Java Grande Forum, November 1998. <http://www.javagrande.org/>.
- [9] Chi-Chao Chang and Thorsten von Eiken. Interfacing Java to the Virtual Interface Architecture. In *ACM 1999 Java Grande Conference*. ACM Press, June 1999.
- [10] B.O. Christiansen, P. Cappello, M.F. Ionescu, M.O. Neary, K.E. Schauser, and D. Wu. Javelin: Internet-base parallel computing using Java. *Concurrency: Practice and Experience*, 9(11):1139–1160, 1997.
- [11] Rossen Dimitrov and Anthony Skjellum. An efficient MPI implementation for Virtual Interface (VI) Architecture-enabled cluster computing. In *Third MPI Developer’s and User’s Conference*. MPI Software Technology, Inc., March 1999.
- [12] Kivanc Dincer. *jmp*i and a performance instrumentation analysis and visualization tool for *jmp*i. In *First UK Workshop on Java for High Performance Network Computing, Europar ’98*, September 1998. <http://www.cs.cf.ac.uk/hpjworkshop/>.

- [13] W. Keith Edwards. *Core Jini*. Prentice Hall, 1999.
- [14] Adam J. Ferrari. JPVM: Network parallel computing in Java. *Concurrency: Practice and Experience*, 10(11-13), 1998.
- [15] Vladimir Getov, Susan Flynn-Hummel, and Sava Mintchev. High-performance parallel programming in Java: Exploiting native libraries. *Concurrency: Practice and Experience*, 10(11-13):873–877, 1998.
- [16] Steven D. Gribble, Matt Welsh, Eric A. Brewer, and David Culler. The MultiSpace: an evolutionary platform for infrastructural service. In *1999 Usenix Annual Technical Conference*, Monterey, June 1999.
- [17] Hewlett Packard Company. *E-speak Architecture Specification*, September 1999. <http://www.e-speak.hp.com/>.
- [18] IMPI Steering Committee. *IMPI—Interoperable Message-Passing Interface*, January 2000. <http://impi.nist.gov/IMPI/>.
- [19] Java Grande Message Passing Working Group. Minutes of Jun 14, 1999 meeting in San Francisco. <http://www.npac.syr.edu/projects/java-mpi/jul99/msg00000.html>.
- [20] Java Grande Message Passing Working Group. Minutes of Oct 1, 1999 meeting in Syracuse. <http://www.npac.syr.edu/projects/java-mpi/oct99/msg00000.html>.
- [21] Glenn Judd, Mark Clement, and Quinn Snell. DOGMA: Distributed object group management architecture. In *ACM 1998 Workshop on Java for High-Performance Network Computing. Palo Alto, February 1998*, volume 10(11-13) of *Concurrency: Practice and Experience*, 1998.
- [22] Message Passing Interface Forum. *MPI: A Message-Passing Interface Standard*. University of Tennessee, Knoxville, TN, June 1995. <http://www.mcs.anl.gov/mpi>.
- [23] Sava Mintchev and Vladimir Getov. Towards portable message passing in Java: Binding MPI. Technical Report TR-CSPE-07, University of Westminster, School of Computer Science, Harrow Campus, July 1997.
- [24] J. E. Moreira. Closing the performance gap between Java and Fortran in technical computing. In *First UK Workshop on Java for High Performance Network Computing*, September 1998. <http://www.cs.cf.ac.uk/hpjworkshop/>.
- [25] J. E. Moreira, S.P. Midkiff, and M. Gupta. From flop to MegaFlops: Java for technical computing. In *Languages and Compilers for Parallel Computing*, volume 1656 of *Lecture Notes in Computer Science*. Springer, 1998.
- [26] J. E. Moreira, S.P. Midkiff, M. Gupta, and R. Lawrence. High performance computing with the array package for Java: A case study using data mining. In *Supercomputing 99*, November 1999.

- [27] mpijava home page.
<http://www.npac.syr.edu/projects/pcrc/HPJava/mpiJava.html>.
- [28] Michael O. Neary, Sean P. Brydon, Paul Kmiec, Sami Rollins, and Peter Cappello. Javelin++: Scalability issues in global computing. In *ACM 1999 Java Grande Conference*. ACM Press, June 1999.
- [29] Christian Nester, Michael Philippsen, and Bernhard Haumacher. A more efficient RMI for Java. In *ACM 1999 Java Grande Conference*. ACM Press, June 1999.
- [30] Northeast Parallel Architectures Center. HPJava project home page.
<http://www.npac.syr.edu/projects/pcrc/HPJava/>.
- [31] M. Philippsen and M. Zenger. JavaParty—transparent remote objects in Java. *Concurrency: Practice and Experience*, 9(11):1225–1242, 1997.
- [32] UC Berkeley Computer Science Division. Ninja project home page.
<http://ninja.cs.berkeley.edu/>.
- [33] Rob van Nieuwpoort, Jason Maassen, Henri E. Bal, Thilo Kielmann, and Ronald Veldema. Wide-area parallel computing in Java. In *ACM 1999 Java Grande Conference*. ACM Press, June 1999.
- [34] Jim Waldo, Geoff Wyant, Ann Wollrath, and Sam Kendall. A note on distributed computing. Technical Report SMLI TR-94-29, Sun Microsystems Laboratories, 1994. Reprinted in [2].
- [35] Matt Welsh. Using Java to make servers scream. Invited talk at ACM 1999 Java Grande Conference, San Francisco, CA, June, 1999.
- [36] Matt Welsh. The Berkeley Ninja project. In *ISCOPE '99*, San Francisco, September 1999.
- [37] Peng Wu, Sam Midkiff, Jose Moreira, and Manish Gupta. Efficient support for complex numbers in Java. In *ACM 1999 Java Grande Conference*. ACM Press, June 1999.
- [38] Narendar Yalamanchilli and William Cohen. Communication performance of Java based Parallel Virtual Machines. *Concurrency: Practice and Experience*, 10(11-13), 1998.

David Bryan Carpenter

Northeast Parallel Architectures Center	Phone: 315 443 5068
Syracuse University	Fax: 315 443 1973
111 College Place	E-mail: dbc@npac.syr.edu
Syracuse, NY 13244-4100	URL: http://www.npac.syr.edu/users/dbc

Education

Ph D. in Physics, University of London (1979-1983).
B.Sc. in Physics, University of London (1976-1979).

Professional Experience

1996– Research Scientist, NPAC, Syracuse University **1994–1995** Programmer, High Performance Computing Centre, Southampton, UK. **1989–93** Research Fellow in Department of Electronics and Computer Science and Department of Physics, Southampton, UK. **1989** Employed at “Transputer Technology Solutions”, Southampton, UK. **1985–1988** Research Fellow in Department of Physics, Southampton, UK. **1985** Royal Society Overseas Fellowship at DESY, Hamburg. **1983–1984** Research Fellow in Theoretical Physics Department, Edinburgh University.

Select List of Publications:

- [1] Bryan Carpenter, Geoffrey Fox, Sung Hoon Ko and Sang Lim “Object Serialization for Marshalling Data in a Java Interface to MPI”, *ACM 1999 Java Grande Conference*, ACM Press 1999.
- [2] Mark Baker, Bryan Carpenter, Geoffrey Fox, Sung Hoon Ko and Sang Lim, “mpiJava: An Object-oriented Java Interface to MPI”, *Intl. Workshop on Java for Parallel and Distributed Computing, IPPS/SPDP '99*, San Juan, Puerto Rico, April 1999.
- [3] Bryan Carpenter, Guansong Zhang, Geoffrey Fox, Xiaoming Li, Xinying Li and Yuhong Wen, “Towards a Java environment for SPMD programming”, *4th International Euromicro Conference*, Springer, 1998.
- [4] Guansong Zhang, Bryan Carpenter and Geoffrey Fox and Xinying Li and Yuhong Wen, “The HPspmd Model and its Java Binding.”, chapter in book, R. Buyya ed, *High Performance Cluster Computing, Vol 2*, Prentice Hall 1999.
- [5] G. Zhang, B. Carpenter, G. Fox, X. Li, X. Li and Y. Wen, “PCRC-based HPF Compilation”, *10th Int’l Workshop on Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science, 1336, Springer 1997.
- [6] B. Carpenter, Y.-J. Chang, G. Fox, D. Leskiw and X. Li, “Experiments with *HP Java*”, *Concurrency: Practice and Experience*, Vol 9, num 9 (1997), p633.
- [7] J. Merlin, B. Carpenter and Tony Hey, “shpf: a Subset High Performance Fortran compilation system”, *Fortran Journal*, (1996), pp 2-6.
- [8] D.B. Carpenter, “Adlib: A Distributed Array Library to Support HPF Translation”, *5th Workshop on Compilers on Parallel Computer*, Malaga 1995.

- [9] D.B. Carpenter and H. Glaser, “Some Lattice-Based Scientific Problems, expressed in Haskell”, *Journal of Functional Programming*, 6(3):419-443, May 1996.
- [10] As. Abada, C.R. Allton, Ph. Boucard, D.B. Carpenter, M. Crisafulli, S. Güsken, P. Hernadez, V. Lubicz, G. Martinelli, O. Pène, C.T. Sachrajda, K. Schilling, G. Siegert and R. Sommer, “Semi-leptonic Decays of Heavy Flavours on a Fine-grained Lattice”, *Nuclear Physics* B416 (1994) p675.

Summary of Interests:

Carpenter has worked in parallel computing since 1985, when he was one of the first scientists to exploit the Inmos transputer for simulations of physical systems. Working at the University of Southampton, UK he became involved with design of libraries to support parallel computing. In a fruitful collaboration with John Merlin, then also at Southampton, he was codeveloper of the *shpf* system, an early implementation of subset HPF. This work produced the first implementation of the Adlib runtime library. After moving to Syracuse in 1996, Carpenter worked in the PCRC project. As part of this work the Adlib library was substantially redesigned to meet the requirements of a new HPF compiler, and reimplemented. Eventually it was delivered as the NPAC PCRC runtime kernel. Work on C++ interfaces to Adlib was a formative influence in the ongoing work on HPJava and HPspmd. Carpenter’s recent research has focussed around the HPspmd project, which is developing translators and other support for library-based data-parallel programming, using Java and other languages. The mpiJava software for message-passing parallel programming was a spin-off from this project.

Ph.D. Advisor

Elliot Leader

Geoffrey Charles Fox

gcf@cs.fsu.edu, <http://www.npac.syr.edu/DC>,

Phone: 315-443-2163 or 850-644-4587, Fax: (315) 443-4741

NPAC	Computational Science and Information Technology
Syracuse University	Florida State University
111 College Place	400 Dirac Science Library
Syracuse NY 13244-4100	Tallahassee Florida 32306-4130

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:

2000-	Professor of Computer Science, Florida State University
1990-	Professor of Computer Science, Syracuse University
1990-	Professor of Physics, Syracuse University
1990-2000	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, Brookhaven National Laboratory, Long Island
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 Editor: Concurrency: Practice and Experience (John Wiley, Inc.)

Selected List of Publications - 5 general ones preceded by 5 specific to proposal

- 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., Messina, P., Williams, R., Parallel Computing Works!, Morgan
Kaufmann, San Mateo Ca, 1994.

- 3) 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).
- 4) Fox, G "Internetics: Technologies, Applications and Academic Fields" Invited Chapter in Book :Feynman and Computation", edited by A.J.G. Hey, Perseus Books (1999)
- 5) Fox, G., Scavo T., Bernholdt D.,Markowski R.,McCracken N.,Podgorny M., Mitra D. and Malluhi Q., "Synchronous Learning at a Distance: Experiences with TangoInteractive", in *Proceedings of SC98*, Orlando, November 1998.
- 6) Fox G.C., Furmanski W., "Computing on the Web,New Approaches to Parallel Processing,Petaop and Exaop Performance in the Year 2007 IEEE Internet Computing 1:2,38-46, 1997
- 7) Akarsu E., Fox, G.C., Furmanski W., Haupt T., "WebFlow -- High-level Programming environment and Visual Authoring Toolkit for High Performance Distributed Computing" in *Proceedings of SC98*, Orlando, November 1998.
- 8) Erol Akarsu, Geoffrey Fox, Tomasz Haupt, Alexey Kalinichenko, Kang-Seok Kim, Praveen Sheethaalnath, and Choon-Han Youn , "Using Gateway System to Provide a Desktop Access to High Performance Computational Resources", *Proceedings of HPDC-8 Conference*, Redondo Beach Ca., Aug 3-6, 1999, IEEE Press.
- 9) Bryan Carpenter and Geoffrey Fox and Sung Hoon Ko and Sang Lim, "Object Serialization for Marshalling Data in a Java Interface to MPI", *ACM 1999 Java Grande Conference*, ACM Press 1999.)
- 10) Mark Baker and Bryan Carpenter and Geoffrey Fox and Sung Hoon Ko and Sang Lim, "mpiJava: An Object-Oriented Java interface to MPI", *International Workshop on Java for Parallel and Distributed Computing, IPPS/SPDP '99* San Juan, Puerto Rico, April 1999.

Summary of Interests

Fox has worked in a variety of applied computer science fields with his work on computational physics evolving into contributions to parallel computing initially involving the hypercube architecture. He has worked on the computing issues in several application areas – currently focussing on Earthquake Science. Over the last three years, his major activity has been the use of Object Web technologies to build collaboration systems and their application in an integrated approach to synchronous and asynchronous distance education. He has led activities to develop prototype high performance Java and Fortran compilers and their runtime support. His research group has pioneered use of CORBA and Java for both collaboration and distributed computing. He helped set up the Java Grande forum to encourage use of Java in large-scale computations. Fox is a proponent for the development of computational science and its follow on "Internetics" as an academic discipline and a scientific method.

Adviser: Dr. Richard Eden Cambridge University

SUMMARY PROPOSAL BUDGET

YEAR 1

ORGANIZATION Florida State University				FOR NSF USE ONLY			
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR David Carpenter				PROPOSAL NO.	DURATION (months)		
				AWARD NO.	Proposed	Granted	
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)				NSF Funded Person-mos.		Funds Requested By proposer	Funds granted by NSF (if different)
	CAL	ACAD	SUMR				
1. David Carpenter - PI	4.00	0.00	0.00	\$ 18,670			
2. Geoffrey C Fox - senior scientist	0.00	0.00	0.00	0			
3.							
4.							
5.							
6. (0) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	0.00	0.00	0.00	0			
7. (2) TOTAL SENIOR PERSONNEL (1 - 6)	4.00	0.00	0.00	18,670			
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)							
1. (0) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00	0			
2. (0) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00	0			
3. (2) GRADUATE STUDENTS				32,000			
4. (2) UNDERGRADUATE STUDENTS				4,000			
5. (0) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				0			
6. (0) OTHER				0			
TOTAL SALARIES AND WAGES (A + B)				54,670			
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				3,615			
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				58,285			
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)							
TOTAL EQUIPMENT				0			
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				3,000			
2. FOREIGN				0			
F. PARTICIPANT SUPPORT COSTS							
1. STIPENDS \$			0				
2. TRAVEL			0				
3. SUBSISTENCE			0				
4. OTHER			0				
(0) TOTAL PARTICIPANT COSTS				0			
G. OTHER DIRECT COSTS							
1. MATERIALS AND SUPPLIES				2,000			
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				0			
3. CONSULTANT SERVICES				0			
4. COMPUTER SERVICES				0			
5. SUBAWARDS				0			
6. OTHER				4,000			
TOTAL OTHER DIRECT COSTS				6,000			
H. TOTAL DIRECT COSTS (A THROUGH G)				67,285			
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) % of MTDC (Rate: 46.5000, Base: 63285)							
TOTAL INDIRECT COSTS (F&A)				29,427			
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				96,712			
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.)				0			
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				\$ 96,712			
M. COST SHARING PROPOSED LEVEL \$ 0				AGREED LEVEL IF DIFFERENT \$			
PI / PD TYPED NAME & SIGNATURE*			DATE	FOR NSF USE ONLY			
David Carpenter				INDIRECT COST RATE VERIFICATION			
ORG. REP. TYPED NAME & SIGNATURE*			DATE	Date Checked	Date Of Rate Sheet	Initials - ORG	

SUMMARY PROPOSAL BUDGET

YEAR 2

ORGANIZATION Florida State University				FOR NSF USE ONLY			
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR David Carpenter				PROPOSAL NO.	DURATION (months)		
				AWARD NO.	Proposed	Granted	
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)				NSF Funded Person-mos.		Funds Requested By proposer	Funds granted by NSF (if different)
	CAL	ACAD	SUMR				
1. David Carpenter - Principal Investigator	4.00	0.00	0.00	\$	19,230	\$	
2.							
3.							
4.							
5.							
6. (0) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	0.00	0.00	0.00		0		
7. (1) TOTAL SENIOR PERSONNEL (1 - 6)	4.00	0.00	0.00		19,230		
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)							
1. (0) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00		0		
2. (0) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00		0		
3. (2) GRADUATE STUDENTS					32,960		
4. (2) UNDERGRADUATE STUDENTS					4,000		
5. (0) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)					0		
6. (0) OTHER					0		
TOTAL SALARIES AND WAGES (A + B)					56,190		
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)					3,723		
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)					59,913		
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)							
TOTAL EQUIPMENT					0		
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)					3,000		
2. FOREIGN					0		
F. PARTICIPANT SUPPORT COSTS							
1. STIPENDS \$			0				
2. TRAVEL			0				
3. SUBSISTENCE			0				
4. OTHER			0				
(0) TOTAL PARTICIPANT COSTS					0		
G. OTHER DIRECT COSTS							
1. MATERIALS AND SUPPLIES					2,000		
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION					0		
3. CONSULTANT SERVICES					0		
4. COMPUTER SERVICES					0		
5. SUBAWARDS					0		
6. OTHER					4,000		
TOTAL OTHER DIRECT COSTS					6,000		
H. TOTAL DIRECT COSTS (A THROUGH G)					68,913		
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) %TDC-tuition (Rate: 46.5000, Base: 64913)							
TOTAL INDIRECT COSTS (F&A)					30,184		
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)					99,097		
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.)					0		
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				\$	99,097	\$	
M. COST SHARING PROPOSED LEVEL \$ 0				AGREED LEVEL IF DIFFERENT \$			
PI / PD TYPED NAME & SIGNATURE* David Carpenter				DATE	FOR NSF USE ONLY		
					INDIRECT COST RATE VERIFICATION		
ORG. REP. TYPED NAME & SIGNATURE*				DATE	Date Checked	Date Of Rate Sheet	Initials - ORG

SUMMARY PROPOSAL BUDGET

YEAR 3

ORGANIZATION Florida State University				FOR NSF USE ONLY			
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR David Carpenter				PROPOSAL NO.	DURATION (months)		
					Proposed	Granted	
				AWARD NO.			
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)				NSF Funded Person-mos.		Funds Requested By proposer	Funds granted by NSF (if different)
	CAL	ACAD	SUMR				
1. David Carpenter - Principal Investigator	4.00	0.00	0.00	\$ 19,807			
2.							
3.							
4.							
5.							
6. (0) OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)	0.00	0.00	0.00	0			
7. (1) TOTAL SENIOR PERSONNEL (1 - 6)	4.00	0.00	0.00	19,807			
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)							
1. (0) POST DOCTORAL ASSOCIATES	0.00	0.00	0.00	0			
2. (0) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)	0.00	0.00	0.00	0			
3. (2) GRADUATE STUDENTS				33,949			
4. (2) UNDERGRADUATE STUDENTS				4,000			
5. (0) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)				0			
6. (0) OTHER				0			
TOTAL SALARIES AND WAGES (A + B)				57,756			
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)				3,834			
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)				61,590			
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)							
TOTAL EQUIPMENT				0			
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)				3,000			
2. FOREIGN				0			
F. PARTICIPANT SUPPORT COSTS							
1. STIPENDS \$			0				
2. TRAVEL			0				
3. SUBSISTENCE			0				
4. OTHER			0				
(0) TOTAL PARTICIPANT COSTS				0			
G. OTHER DIRECT COSTS							
1. MATERIALS AND SUPPLIES				2,000			
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION				0			
3. CONSULTANT SERVICES				0			
4. COMPUTER SERVICES				0			
5. SUBAWARDS				0			
6. OTHER				4,000			
TOTAL OTHER DIRECT COSTS				6,000			
H. TOTAL DIRECT COSTS (A THROUGH G)				70,590			
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE) %TDC less tuition (Rate: 46.5000, Base: 66590)							
TOTAL INDIRECT COSTS (F&A)				30,964			
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)				101,554			
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.)				0			
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)				\$ 101,554			
M. COST SHARING PROPOSED LEVEL \$ 0				AGREED LEVEL IF DIFFERENT \$			
PI / PD TYPED NAME & SIGNATURE*			DATE	FOR NSF USE ONLY			
David Carpenter				INDIRECT COST RATE VERIFICATION			
ORG. REP. TYPED NAME & SIGNATURE*			DATE	Date Checked	Date Of Rate Sheet	Initials - ORG	

SUMMARY PROPOSAL BUDGET

Cumulative

ORGANIZATION Florida State University				FOR NSF USE ONLY			
PRINCIPAL INVESTIGATOR / PROJECT DIRECTOR David Carpenter				PROPOSAL NO.	DURATION (months)		
				AWARD NO.	Proposed	Granted	
A. SENIOR PERSONNEL: PI/PD, Co-PI's, Faculty and Other Senior Associates (List each separately with title, A.7. show number in brackets)				NSF Funded Person-mos.		Funds Requested By proposer	Funds granted by NSF (if different)
				CAL	ACAD	SUMR	
1. David Carpenter - PI				12.00	0.00	0.00	\$ 57,707
2. Geoffrey C Fox - senior scientist				0.00	0.00	0.00	0
3.							
4.							
5.							
6. () OTHERS (LIST INDIVIDUALLY ON BUDGET JUSTIFICATION PAGE)				0.00	0.00	0.00	0
7. (2) TOTAL SENIOR PERSONNEL (1 - 6)				12.00	0.00	0.00	57,707
B. OTHER PERSONNEL (SHOW NUMBERS IN BRACKETS)							
1. (0) POST DOCTORAL ASSOCIATES				0.00	0.00	0.00	0
2. (0) OTHER PROFESSIONALS (TECHNICIAN, PROGRAMMER, ETC.)				0.00	0.00	0.00	0
3. (6) GRADUATE STUDENTS							98,909
4. (6) UNDERGRADUATE STUDENTS							12,000
5. (0) SECRETARIAL - CLERICAL (IF CHARGED DIRECTLY)							0
6. (0) OTHER							0
TOTAL SALARIES AND WAGES (A + B)							168,616
C. FRINGE BENEFITS (IF CHARGED AS DIRECT COSTS)							11,172
TOTAL SALARIES, WAGES AND FRINGE BENEFITS (A + B + C)							179,788
D. EQUIPMENT (LIST ITEM AND DOLLAR AMOUNT FOR EACH ITEM EXCEEDING \$5,000.)							
TOTAL EQUIPMENT							0
E. TRAVEL 1. DOMESTIC (INCL. CANADA, MEXICO AND U.S. POSSESSIONS)							9,000
2. FOREIGN							0
F. PARTICIPANT SUPPORT COSTS							
1. STIPENDS \$ <u>0</u>							
2. TRAVEL <u>0</u>							
3. SUBSISTENCE <u>0</u>							
4. OTHER <u>0</u>							
(0) TOTAL PARTICIPANT COSTS							0
G. OTHER DIRECT COSTS							
1. MATERIALS AND SUPPLIES							6,000
2. PUBLICATION COSTS/DOCUMENTATION/DISSEMINATION							0
3. CONSULTANT SERVICES							0
4. COMPUTER SERVICES							0
5. SUBAWARDS							0
6. OTHER							12,000
TOTAL OTHER DIRECT COSTS							18,000
H. TOTAL DIRECT COSTS (A THROUGH G)							206,788
I. INDIRECT COSTS (F&A)(SPECIFY RATE AND BASE)							
TOTAL INDIRECT COSTS (F&A)							90,576
J. TOTAL DIRECT AND INDIRECT COSTS (H + I)							297,364
K. RESIDUAL FUNDS (IF FOR FURTHER SUPPORT OF CURRENT PROJECTS SEE GPG II.D.7.j.)							0
L. AMOUNT OF THIS REQUEST (J) OR (J MINUS K)							\$ 297,364 \$
M. COST SHARING PROPOSED LEVEL \$ 0				AGREED LEVEL IF DIFFERENT \$			
PI / PD TYPED NAME & SIGNATURE* David Carpenter				DATE	FOR NSF USE ONLY		
					INDIRECT COST RATE VERIFICATION		
ORG. REP. TYPED NAME & SIGNATURE*				DATE	Date Checked	Date Of Rate Sheet	Initials - ORG

Budget Justification

Personnel

Salary costs are for 4 months per annum of the Principal Investigator, Carpenter, 2 Graduate Research Assistants, and 2 Undergraduate Research Students, including summer research.

Rates are based on actual rates where there are incumbents and approved rates for unfilled positions. Rates for senior staff and graduate research assistants are increased by 3% in year two and in year three.

There is no salary requested for Dr Fox.

Fringe Benefits

Fringe Benefits are 18.4% for senior staff and 0.6% for students.

Travel

Funds requested are for attending conferences to help keep abreast of the latest developments in the field. Estimates are based on previous conference attending costs.

Tuition

Tuition of \$2000 per year for each graduate student is recorded in item G6.

Indirect costs

Florida State University's indirect cost is $46.5\% \times \text{MTDC}$ (total direct costs minus tuition).

Current and Pending Support

(See GPG Section II.D.8 for guidance on information to include on this form.)

The following information should be provided for each investigator and other senior personnel. Failure to provide this information may delay consideration of this proposal.

Investigator: **David Carpenter**

Other agencies (including NSF) to which this proposal has been/will be submitted.

Support: ☒ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title: **Programming Models from Fortran to Java**

Source of Support: **National Science Foundation**

Total Award Amount: \$ **346,827** Total Award Period Covered: **11/01/98 - 10/31/01**

Location of Project: **Syracuse University and Florida State University**

Person-Months Per Year Committed to the Project. Cal: **3.00** Acad: **0.00** Sumr: **0.00**

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: \$ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: \$ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: \$ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

Support: ☐ Current ☐ Pending ☐ Submission Planned in Near Future ☐ *Transfer of Support

Project/Proposal Title:

Source of Support:

Total Award Amount: \$ Total Award Period Covered:

Location of Project:

Person-Months Per Year Committed to the Project. Cal: Acad: Sumr:

*If this project has previously been funded by another agency, please list and furnish information for immediately preceding funding period.

H. Facilities, Equipment, and Other Resources

H.1 Local Computer Facilities

The School of Computational Science and Information Technology (CSIT) at FSU has a substantial computer infrastructure, including 5 four-processor Origin 200s, a cluster of 30 Pentium Pros with 100 Mbit/sec ethernet network, a 16 processor IBM SP2, two 4 processor ES40 Compaq machines, and an assortment of older heterogeneous clusters. These will be supplemented by about 10 modern Sun servers from NPAC. Florida State University has committed to the allocation of 5 to 8 million dollars to the purchase of a massively parallel supercomputer that will achieve over a Teraflop of processing power over the next two years.

CSIT presently has 3.5 system administrators who are responsible for the daily upkeep of CSIT resources, software upgrades, troubleshooting, helping users, and performing backups.

H.2 Office

CSIT provides adequate office facilities for all participants.

H.3 Major Equipment

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

H.4 Other Resources

CSIT provides basic secretarial services, duplicating facilities, and the like.

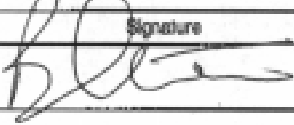
CERTIFICATION PAGE

Certification for Principal Investigators and Co-Principal Investigators:

I certify to the best of my knowledge that:

- (1) the statements herein (excluding scientific hypotheses and scientific opinions) are true and complete, and
 (2) the text and graphics herein as well as any accompanying publications or other documents, unless otherwise indicated, are the original work of the signatories or individuals working under their supervision. I agree to accept responsibility for the scientific conduct of the project and to provide the required progress reports if an award is made as a result of this application.

I understand that the willful provision of false information or concealing a material fact in this proposal or any other communication submitted to NSF is a criminal offense (U.S. Code, Title 18, Section 1001).

Name (Typed)	Signature	Social Security No.*	Date
P/CPD Bryan Carpenter		SSNs are confidential and are not displayed on FASTlane submissions	2/15/00
Co-P/CPD			
Co-P/CPD			
Co-P/CPD			
Co-P/CPD			

Certification for Authorized Organizational Representative or Individual Applicant:

By signing and submitting this proposal, the individual applicant or the authorized official of the applicant institution is: (1) certifying that statements made herein are true and complete to the best of his/her knowledge; and (2) agreeing to accept the obligation to comply with NSF award terms and conditions if an award is made as a result of this application. Further, the applicant is hereby providing certifications regarding Federal debt status, debarment and suspension, drug-free workplace, and lobbying activities (see below), as set forth in Grant Proposal Guide (GPG), NSF 00-2. Willful provision of false information in this application and its supporting documents or in reports required under an ensuing award is a criminal offense (U. S. Code, Title 18, Section 1001).

In addition, if the applicant institution employs more than fifty persons, the authorized official of the applicant institution is certifying that the institution has implemented a written and enforced conflict of interest policy that is consistent with the provisions of Grant Policy Manual Section 510; that to the best of his/her knowledge, all financial disclosures required by that conflict of interest policy have been made; and that all identified conflicts of interest will have been satisfactorily managed, reduced or eliminated prior to the institution's expenditure of any funds under the award, in accordance with the institution's conflict of interest policy. Conflict which cannot be satisfactorily managed, reduced or eliminated must be disclosed to NSF.

Debt and Debarment Certifications

(If answer "yes" to either, please provide explanation.)

Is the organization delinquent on any Federal debt?

Yes ☐

No ☒

Is the organization or its principals presently debarred, suspended, proposed for debarment, declared ineligible, or voluntarily excluded from covered transactions by any Federal department or agency?

Yes ☐

No ☒

Certification Regarding Lobbying

This certification is required for an award of a Federal contract, grant, or cooperative agreement exceeding \$100,000 and for an award of a Federal loan or a commitment providing for the United States to insure or guarantee a loan exceeding \$150,000.

Certification for Contracts, Grants, Loans and Cooperative Agreements


The undersigned certifies, to the best of his or her knowledge and belief, that:

(1) No federal appropriated funds have been paid or will be paid, by or on behalf of the undersigned, to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with the awarding of any federal contract, the making of any Federal grant, the making of any Federal loan, the entering into of any cooperative agreement, and the extension, continuation, renewal, amendment, or modification of any Federal contract, grant, loan, or cooperative agreement.

(2) If any funds other than Federal appropriated funds have been paid or will be paid to any person for influencing or attempting to influence an officer or employee of any agency, a Member of Congress, an officer or employee of Congress, or an employee of a Member of Congress in connection with this Federal contract, grant, loan, or cooperative agreement, the undersigned shall complete and submit Standard Form-LLL, "Disclosure Form to Report Lobbying," in accordance with its instructions.

(3) The undersigned shall require that the language of this certification be included in the award documents for all subawards at all levels including subcontracts, subgrants, and contracts under grants, loans, and cooperative agreements and that all subrecipients shall certify and disclose accordingly.

This certification is a material representation of fact upon which reliance was placed when this transaction was made or entered into. Submission of this certification is a prerequisite for making or entering into this transaction imposed by section 1352, title 31, U.S. Code. Any person who fails to file the required certification shall be subject to a civil penalty of not less than \$10,000 and not more than \$100,000 for each such failure.

AUTHORIZED ORGANIZATIONAL REPRESENTATIVE		SIGNATURE	DATE
NAME/TITLE (TYPED) Raymond E. Byc, Jr., Interim VP Rsrch			02/15/00
TELEPHONE NUMBER 850-644-5260	ELECTRONIC MAIL ADDRESS nsfaward@res.fsu.edu	FAX NUMBER 850-644-1464	

*SUBMISSION OF SOCIAL SECURITY NUMBERS IS VOLUNTARY AND WILL NOT AFFECT THE ORGANIZATION'S ELIGIBILITY FOR AN AWARD. HOWEVER, THEY ARE AN INTEGRAL PART OF THE INFORMATION SYSTEM AND ASSIST IN PROCESSING THE PROPOSAL. SSN SOLICITED UNDER NSF ACT OF 1950, AS AMENDED.

Dr David Bryan Carpenter
NPAC, 111 College Place
Syracuse University
Syracuse, NY 13244-4100
U.S.A.

February 15, 2000

Dear Bryan,

Re: Collaborative research in Message Passing Environments for Java

Herewith I confirm our willingness to collaborate closely with the project proposed by you, entitled, "Integration of High Performance Message-based Programming Environments with Distributed Objects and the Web". As previously discussed and agreed, particular areas of joint research interest include the development of a message passing API specification for Java, a pilot implementation, as well as performance evaluation experiments and multi-language application software development.

We are also looking forward to collectively participating in the forthcoming International Java Grande forum meetings, which would be an ideal environment for discussion and presentation of joint ideas and results, within the target research field. Your project proposal would also be a natural extension to our current collaboration in the area of MPI-like Java API specification within the Message Passing Working Group of the Java Grande forum.

Best regards,



Dr Vladimir Getov
Reader in High-Performance Computing
Director, Distributed and High-Performance Computing Group
Chairman, Java Grande Message-Passing Working Group

HARROW SCHOOL OF COMPUTER SCIENCE

Artificial Intelligence Head of School Vassilis Konstantinou BSc MSc
Computer Systems & Teleinformatics Harrow Campus Watford Road Northwick Park Harrow HA1 3TP
Information & Software Systems Telephone 0171 911 5000 Fax 0171 911 5906
<http://www.wmin.ac.uk/CS> Email csadvice@wmin.ac.uk