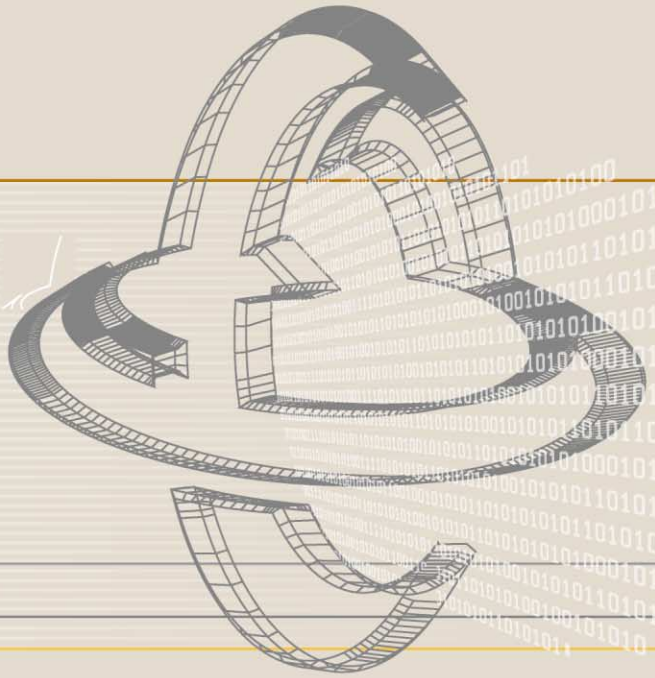
# 2000 REPORT


Innovative Computing Laboratory
COMPUTER SCIENCE DEPARTMENT
UNIVERSITY OF TENNESSEE

# CONTENTS



# INTRODUCTION

## Mission Statement

The Innovative Computing Laboratory's (ICL) mission is to accelerate the pace of scientific discovery in the community by providing leadership, technology, and software.

ICL aspires to be a world leader in enabling technologies and software for scientific computing. Our vision is to provide high performance tools to tackle science's most challenging problems and to play a major role in the development of standards for scientific computing in general. The result will be a rate of scientific progress previously unknown.

## Background

ICL was established in the fall of 1989 when Dr. Jack Dongarra came to the University of Tennessee (UT) from Argonne National Laboratory (ANL). Dr. Dongarra was given a dual appointment as Distinguished Professor in the Computer Science Department at the university and as Distinguished Scientist at Oak Ridge National Laboratory (ORNL). This dual position was established by the UT/ORNL Science Alliance, Tennessee's oldest and largest Center of Excellence, as a means for attracting top research scientists from around the country and the world to visit the university and collaborate. Subsequently, many post-doctoral researchers and professors from various research backgrounds such as mathematics, geology, chemistry, etc. visited the university. Many of these scientists, such as Zhaojun Bai of the University of California Davis, Adam Beguelin from Inktomi Corporation, Jaeyoung Choi of Soongsil University (Korea), Andy Cleary from Data Digest, Frederic Desprez of ÉNS Lyon, Robert van de Geijn of the University of Texas Austin, Roldan Pozo from NIST, Françoise Tisseur of Manchester University, and Bernard Tourancheau of Université Claude Bernard de Lyon passed through UT as post-doctoral researchers and worked with Dr. Dongarra. Such scientists were instrumental in helping him attract additional researchers as well as top graduate students.

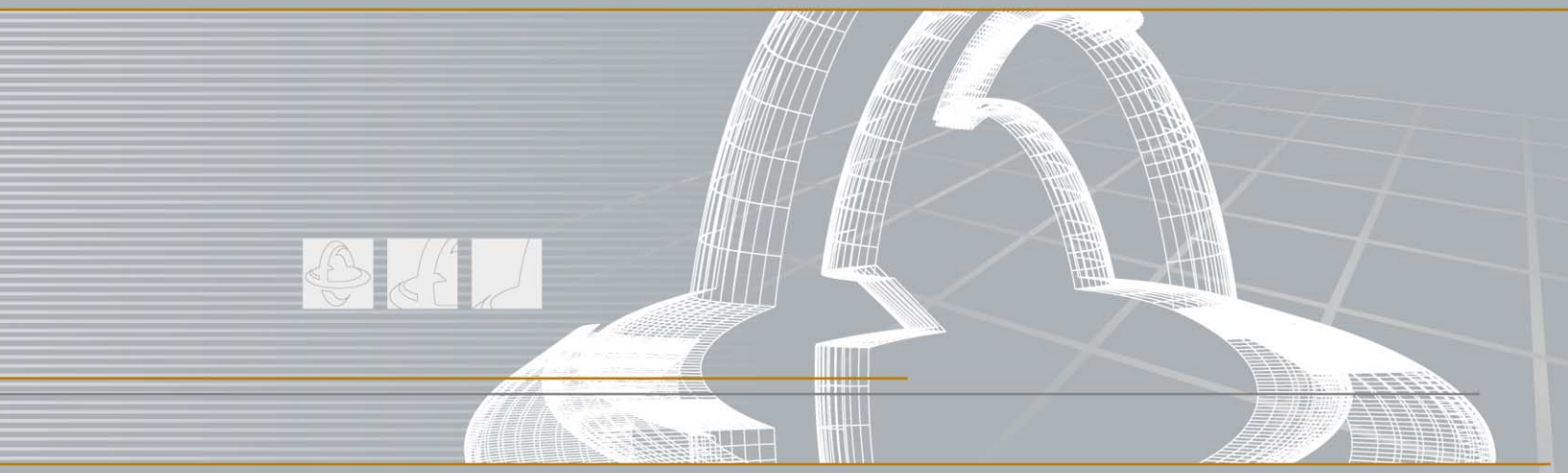
Through interactions with colleagues at Rice University, ICL became an integral part of the Center for Research on Parallel Computation (CRPC), a National Science Foundation (NSF) Science and Technology Center established in 1989 and lead by Rice University. CRPC worked to make parallel computation accessible to industry, government, and academia and to educate a new generation of technical professionals. Through the 1990s, ICL worked on a number of efforts that since have become part of the basic fabric of scientific computing in the world. Our enabling technology efforts include the BLAS, LAPACK, ScaLAPACK, PVM, MPI, Netlib, NHSE, and the TOP500. These successes are continuing along with current ICL efforts such as ATLAS, PAPI, HARNESS, DSI, IBP, RIB, and NetSolve.

Having linear algebra as its original focus, the group has evolved and expanded to focus on many progressive areas of high performance computing research such as distributed computing, software repositories, and performance evaluation. Currently a University Distinguished Professor, Dr. Dongarra continues as director of ICL. As such, he not only serves as principal investigator (PI) for many of the projects, but he also maintains some level of participation in all projects.

## Profile

Located at the heart of the UT campus in Knoxville, ICL is an internationally recognized academic leader in high performance computing (HPC) research. In 1999, two of our projects were awarded R&D 100 awards: ATLAS and NetSolve. In addition, we have recently moved into new facilities at the university, due in large part to our incredible growth over the last few years.

Operating under grants totaling nearly $5 million, ICL is also recognized by senior UT administration as one of UT's top research centers. According to Dr. T. Dwayne McCay, Vice President of Research and Information Technology, "Jack Dongarra and the students and staff of the Innovative Computing Laboratory have been leading our University and our nation in high performance computing and information technology research for more than a decade now. Looking out at the decade to come, we're really excited by the prospect of what their creativity and their new discoveries will bring us. It is the work of Dr. Dongarra's team that exemplifies why we are so determined to be a top 25 research university, the efficient solution to modern problems requires the most modern tools and the best trained minds and they are attracted to the great research universities by the opportunity to work with someone like Jack Dongarra. These universities in turn fuel the engine that drives the economy which raises the standard of living for all our citizens."



# RESEARCH

As ICL has grown over the years, the range and diversity of the research and development carried out by our staff and students has increased in parallel. In the past year alone, we supported or participated in more than 20 significant projects. Our large and wide-ranging portfolio of research projects has evolved over the course of a decade, beginning from a narrow but solid foundation.

The original focus of ICL was Dr. Dongarra's work in numerical linear algebra and the numerical libraries that encode its operations in software. But driven by the relentless demand for higher performance in the computational science community, ICL built upon its successes in the area of numerical libraries and the growing strength of its personnel to break new ground in the areas of high performance and distributed computing. Similarly, our work with numerical libraries created a strong area of expertise in performance evaluation and benchmarking for high-end computers. The enormous investments by both government and private industry in high performance computing have made our ability to do research in this area correspondingly important. Finally, as a by-product of a long tradition of delivering high quality software produced from our research, we have helped to lead the movement to build robust, comprehensive, and well-organized software repositories.

On the following pages, many of the projects in our main research areas - numerical libraries, high performance distributed computing, performance evaluation and benchmarking, and software repositories - are described in detail. Evidence of the perceived value of this work is apparent in the range of agencies and organizations that have funded these efforts. The main source of support has been federal agencies who are charged with investing the nation's research funding: the National Science Foundation (NSF), Department of Energy (DOE), Department of Defense (DoD), the Defense Advanced Research Projects Agency (DARPA), and the National Aeronautics and Space Administration (NASA). However, strong support from private industry has also played a significant role. Some companies have targeted specific ICL projects. But others, most notably IBM, Microsoft, Sun Microsystems, Intel, Hewlett-Packard, and MathWorks, have made contributions to our work that are more general and open-ended. We gratefully acknowledge the significance of their generosity to our success.

## NUMERICAL LINEAR ALGEBRA

PROJECTS

ATLAS
BLAST
F2J
LAPACK
ScaLAPACK

ICL's original area of research and one of our strongest areas today is linear algebra algorithms and software. Linear algebra operations form the core of an overwhelming number of scientific applications; thus, having efficient algorithms and implementations for these operations is of utmost importance in achieving good performance for these applications. In collaboration with other researchers and with industry, we have led successful efforts to standardize library interfaces for the Basic Linear Algebra Subroutines (BLAS) as well as for higher level dense linear algebra routines such as those for solving linear systems. Moreover, our groundbreaking research in efficient algorithms and implementations for these routines has been adopted and refined by industry to produce highly efficient linear algebra implementations for most architectures, with the result that applications that rely on the standard libraries can achieve excellent performance while remaining portable across multiple platforms. Dense linear algebra software produced via our research has included LINPACK for vector supercomputers, LAPACK for shared memory multiprocessors, and ScaLAPACK for distributed memory multiprocessors. More recently, the ATLAS system has been developed for the automatic generation and optimization of linear algebra software. Recent ICL linear algebra research has also focused on sparse linear algebra in the areas of iterative methods and parallel preconditioning. In addition, we have developed a Fortran to Java (F2J) translator to translate, initially, the BLAS and LAPACK numerical libraries from their Fortran77 reference source code to Java class files.



## DISTRIBUTED COMPUTING

PROJECTS

FT-MPI
GrADS
HARNESS
I2-DSI
IBP
MPI_Connect
NetSolve
SInRG
TORC

Distributed computing is another major area of research for ICL, and our researchers have been involved in a number of distributed computing projects over the past decade. To allow scientific applications to run in parallel across networks of workstations, our research staff, in collaboration with Oak Ridge National Laboratory (ORNL) and Emory University, developed the highly successful Parallel Virtual Machine (PVM) system. Although originally intended for cluster computing, PVM was adopted by most major high performance computing (HPC) vendors as a de facto standard for message passing on distributed memory multiprocessors. The PVM system transparently handles message routing, data conversion for incompatible architectures, and other tasks necessary for operation in a heterogeneous, network environment. Although largely surpassed by the industry-standard Message Passing Interface (MPI), PVM remains widely used and is particularly effective for heterogeneous applications that exploit specific strengths of individual machines on a network.

ICL helped lead the effort to standardize message passing in the form of the Message Passing Interface (MPI), Jack Dongarra served as chairman for the MPI1 Standards Activity. In addition, the MPI_Connect and FT-MPI projects have addressed the needs for MPI interoperability and fault tolerance, respectively, in a networked, heterogeneous environment. FT-MPI is part of a broader project called HARNESS, which is based on the concept of a distributed virtual machine (DVM) and provides a plug-in interface for dynamically customizing, adapting, and extending a heterogeneous network computing environment.

The NetSolve client-server system is another of our distributed computing projects. The goal of NetSolve is to provide users with easy access to remote hardware and software computational resources in a network environment. NetSolve consists of computational servers, agents that handle scheduling decisions, and a variety of client interfaces.

We are also involved in projects such as the Internet2 Distributed Storage Infrastructure (I2-DSI) project, which is designing and deploying a reliable, scalable, high-performance storage services infrastructure for wide-area networks.

## SOFTWARE REPOSITORIES

PROJECTS

NA-Digest/NA-Net

NHSE

Netlib

RIB

ICL and other academic researchers have produced a wealth of numerical and high performance computing software. To preserve this software past the lifetime of the projects that produced it and to provide a central location containing a moderated collection of high-quality software, a software repository effort was begun in the 1980s. The original Netlib collection of mathematical software has grown to include software and technical reports in a variety of areas, including dense and sparse linear algebra, differential equations, optimization, and parallel tools. The Netlib collection is widely used by academic and government researchers as well as by industry and has received over 90,000,000 requests over its 15-year lifetime.

The National High-performance Software Exchange (NHSE) project, which started in 1994, sought to enable other organizations and high performance computing areas to replicate Netlib's success in sharing software resources. The Repository in a Box (RIB) toolkit developed by the NHSE has been used by a number of government agencies to set up repositories in the areas of parallel tools, computational chemistry, signal image processing, and grand challenge applications. RIB's interoperation capabilities allow these repositories to share software information. Netlib and the NHSE and RIB repositories make a vast amount of high quality software easily available to users via the Web, thus accomplishing technology transfer on a large scale.



## PERFORMANCE EVALUATION

PROJECTS

LINPACK Benchmark

PAPI

TOP100 Clusters/
TOP500 Supercomputers

In addition to producing software that helps achieve high performance on parallel computers, ICL has been a leader in benchmarking and performance evaluation efforts that measure and report performance of these machines. Our researchers have developed a number of benchmark codes. The LINPACK Benchmark is a numerically intensive test that has been used for years to measure the floating point performance of computers. Performance on the high performance version of this benchmark, called HPL, is the basis of the semi-annual TOP500 list that ranks the fastest 500 computers in the world as well as the TOP100 Clusters. HPL is a portable, high-performance implementation of the LINPACK Benchmark for distributed memory computers.

Other benchmarks that we have developed or have helped develop include LLCBench, which is a suite of low-level benchmarks that measure the floating point hardware, memory subsystem, and MPI implementation for a given platform. Another benchmark is SparseBench, which uses common iterative methods, preconditioners, and storage schemes to evaluate machine performance on typical sparse operations.

In addition to benchmarks, ICL researchers have developed a portable library interface, called PAPI, for access to hardware performance counters. PAPI facilitates accurate performance measurements that provide information needed by application developers to detect and diagnose performance problems.

# ATLAS

## AUTOMATICALLY TUNED LINEAR ALGEBRA SOFTWARE

ICL TEAM
Clint Whaley
Antoine Petitet
Jack Dongarra

ATLAS (Automatically Tuned Linear Algebra Software) is an instantiation of a new paradigm in high performance library production and maintenance, which we term AEOS (Automated Empirical Optimization of Software). This style of library management has been created in order to allow software to keep pace with the incredible rate of hardware advancement inherent in Moore's Law. ATLAS is the application of this new paradigm to linear algebra software, with the present emphasis on the Basic Linear Algebra Subprograms (BLAS), a widely used, performance-critical, linear algebra kernel library.

Linear algebra routines are widely used in the computational sciences in general, and scientific modeling in particular. In many of these applications, the performance of the linear algebra operations is the main constraint preventing the scientist from modeling more complex problems, which would then more closely match reality. This then dictates an ongoing need for highly efficient routines; as more compute power becomes available the scientist typically increases the complexity/accuracy of the model until the limits of the computational power is reached. Therefore, since many applications have no practical limit of enough accuracy, it is important that each generation of increasingly powerful computers have optimized linear algebra routines available.

Linear algebra is rich in operations, which are highly optimizable, in the sense that a highly tuned code may run multiple orders of magnitude faster than a naively coded routine. However, these optimizations are platform specific, such that an optimization for a given computer architecture will actually cause a slow-down on another architecture. The traditional method of handling this problem has been to produce hand-optimized routines for a given machine. This is a painstaking process, typically requiring many man-months of highly trained (both in linear algebra and computational optimization) personnel. The incredible pace of hardware evolution makes this technique untenable in the long run, particularly so when considering that there are many software layers (e.g., operating systems, compilers, etc), which also effect these kinds of optimizations, that are changing at similar, but independent rates.

Therefore a new paradigm is needed for the production of highly efficient routines in the modern age of computing, and ATLAS represents an implementation of such a set of new techniques. In an AEOS-enabled package such as ATLAS, the package provides many ways of doing the required operations, and uses empirical timings in order to choose the best method for a given architecture. Thus, if written generally enough, an AEOS-aware package can automatically adapt to a new computer architecture in a matter of hours, rather than requiring months or even years of highly-trained professionals' time, as dictated by traditional methods.

ATLAS typically uses code generators (i.e., programs that write other programs) in order to provide the many different ways of doing a given operation, and has sophisticated search scripts and robust timing mechanisms in order to find the best ways of performing the operation for a given architecture.

The BLAS application programming interface (API) is supported by hand-tuned efforts of many hardware vendors, and thus provides a good first target for ATLAS since there is both a large audience for this API, and on those platforms where vendor-supplied BLAS exist, an easy way to determine if ATLAS can provide the required level of performance. Figure 1 shows a performance comparison chart of ATLAS, the vendor-supplied BLAS, and the Fortran77 reference BLAS across various architectures.

What does ATLAS currently provide?

- Complete, optimized, and portable BLAS
- Optimized LU and Cholesky factorization and solve routines that are compatible with LAPACK
- Standard C and Fortran77 APIs for all routines
- Testers and timers for above

What are the requirements for using ATLAS?

Excepting the Fortran77 interface routines, ATLAS is written purely in strict ISO/ANSI C. Compiling the package requires access to a Unix-like command environment (e.g., Unix make, /bin/sh, etc.). ATLAS is optimized for hierarchical memory systems, and will perform best on machines with registers and at least one level of cache. ATLAS runs

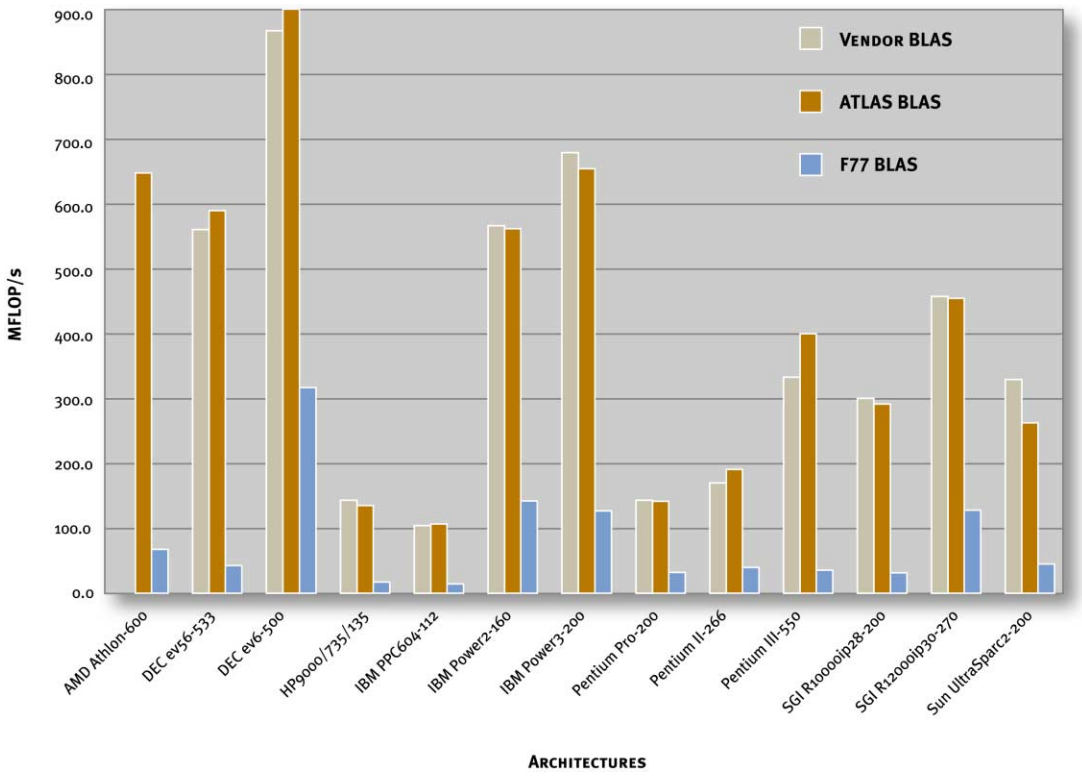



FIGURE 1.

CHART DEPICTING THE PERFORMANCE OF 500 X 500 DGEMM ACROSS MULTIPLE ARCHITECTURES

on any Unix OS possessing an ANSI/ISO C compiler, as well as Windows 9x/NT/2000.

Anyone needing optimized BLAS/LAPACK benefits from ATLAS. There are a large number ATLAS users who call the BLAS directly from their applications. ATLAS is also used by several computer vendors in creating architecture-specific versions of the BLAS. Finally, ATLAS is being considered for inclusion in a number of linear algebra-related packages, such as Matlab, Octave, Maple and Mathematica.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/atlas/

**E-mail**
atlas@cs.utk.edu

**Publications**
Dongarra, J., Petitet, A., Whaley, C. "Automated Empirical Optimization of Software and the ATLAS Project," *Parallel Computing* (to appear in 2001). Also available as University of Tennessee LAPACK Working Note No. 147 http://www.netlib.org/lapack/lawns/lawn147.ps

**Awards**
R&D 100 winner, 1999

**Agency Funding**
Department of Energy (DOE)
National Science Foundation (NSF)

**Related URLs**
BLAS - http://www.netlib.org/blas/
Maple - http://www.maplesoft.com/
Mathematica - http://www.mathematica.com/
Matlab - http://www.mathworks.com/
Octave - http://www.che.wisc.edu/octave/

# BLAST

## BLAS TECHNICAL FORUM

**ICL TEAM (ALPHABETICAL)**

Susan Blackford
Jack Dongarra
Antoine Petitet
Clint Whaley

**COLLABORATORS**

Compaq
Cornell Theory Center
Cray
Florida Institute of Technology
Hewlett-Packard
Hitachi
IBM
Intel
Lucent Technologies
Mississippi State University
National Institute of Standards and Technology (NIST)
NEC
Numerical Algorithms Group, Ltd.
RAL/CERFACS
Rice University
SGI
Texas Instruments
University of California, Berkeley
University of Houston
University of Illinois Urbana - Champaigne
University of Minnesota
University of Notre Dame
University of Tennessee
University of Texas at Austin
Visual Numerics, Inc.

The BLAS Technical Forum was established to consider expanding the specification of a set of kernel routines for linear algebra (historically called the Basic Linear Algebra Subprograms and commonly known as the BLAS) in light of modern software, language, and hardware developments. The BLAS Technical Forum meetings were conducted in the spirit of the earlier BLAS meetings and the standardization efforts of the MPI and High Performance Fortran (HPF) forums, and began with a workshop in November 1995 at the University of Tennessee. Meetings were hosted by universities, government institutions, and software and hardware vendors. The final meeting was held in March 1999.

Various working groups were established at the meetings to consider issues such as the overall functionality, language interfaces, sparse BLAS, distributed-memory dense BLAS, extended and mixed precision BLAS, interval BLAS, and extensions to the existing BLAS. The rules of the forum were adopted from those used for the MPI and HPF forums. The efforts of these working groups are summarized in the BLAST document available on the BLAST Forum web page. Most of the discussions resulted in definitive proposals, which led to the specifications given in chapters two through four of the document. Not all of the discussions resulted in definitive proposals, and such discussions are summarized in the *Journal of Development* in the hope that they may encourage future efforts to take those discussions to a successful conclusion. The minutes from all of the meetings are also contained on the BLAST Forum web page. Virtual meetings and voting on chapters were also conducted via the web page.

A major aim of the standards defined in the document is to enable linear algebra libraries (both public domain and commercial) to interoperate efficiently, reliably, and easily. We believe that hardware and software vendors, higher level library writers, and application programmers all benefit from the efforts of this forum and are the intended end users of these standards.

Numerical linear algebra, particularly the solution of linear systems of equations, linear least squares problems, eigenvalue problems and singular value problems, is fundamental to most calculations in scientific computing and is often the most computationally intensive part of such calculations. Designers of computer programs involving linear algebraic operations have frequently chosen to implement certain low level operations, such as the dot product or the matrix vector product, as separate subprograms. This may be observed both in many published codes and in codes written for specific applications at many computer installations.

Such an approach encourages structured programming and improves the self-documenting quality of the software by specifying basic building blocks and identifying these operations with unique mnemonic names. Since a significant amount of execution time in complicated linear algebraic programs may be spent in a few low level operations, reducing the execution time spent in these operations leads to an overall reduction in the execution time of the program. The programming of some of these low level operations involves algorithmic and implementation subtleties that require care and can be easily overlooked. If there is general agreement on standard names and parameter lists for some of these basic operations, then portability and efficiency can also be achieved.

The first major concerted effort to achieve agreement on the specification of a set of linear algebra kernels resulted in the Level 1 BLAS and associated test suite. The Level 1 BLAS are the specification and implementation in Fortran of subprograms for scalar and vector operations. This was the result of a collaborative project in 1973-77. Following the distribution of the initial version of the specifications to people active in the development of numerical linear algebra software, a series of open meetings was held at conferences, and as a result, extensive modifications were made in an effort to improve the design and make the subprograms more robust. The Level 1 BLAS were extensively and successfully exploited by LINPACK, a software package for the solution of dense and banded linear equation and linear least squares problems.

With the advent of vector machines, hierarchical memory machines and shared memory parallel machines, specifications for the Level 2 and 3 BLAS

(matrix-vector and matrix-matrix operations, respectively) were drawn up in 1984-86 and 1987-88. These specifications made it possible to construct new software to more effectively utilize the memory hierarchy of modern computers. In particular, the Level 3 BLAS allowed the construction of software based upon block-partitioned algorithms, typified by the linear algebra software package, LAPACK. LAPACK is state of the art software for the solution of dense and banded linear equation, linear least squares, eigenvalue, and singular value problems that makes extensive use of all levels of BLAS and particularly utilizes the Level 2 and 3 BLAS for portable performance. LAPACK is widely used in application software and is supported by a number of hardware and software vendors.

To a great extent, the user community embraced the BLAS, not only for performance reasons, but also because developing software around a core of common routines like the BLAS is good software engineering practice. Highly efficient machine-specific implementations of the BLAS are available for most modern high-performance computers. The BLAS have enabled software to achieve high performance with portable code.

The original BLAS concentrated on dense and banded operations. However, many applications require the solution of problems involving sparse matrices. There have also been efforts to specify computational kernels for sparse vector and matrix operations.

The motivation for the kernel operations in the BLAST document is proven functionality. No operation was to be added because it just neatly fit into a framework, and no operation was to be discarded because it did not fit into a particular framework.

Many of the new operations are based upon auxiliary routines in LAPACK (e.g., SUMSQ, GEN_GROT, GEN_HOUSE, SORT, GE_NORM, GE_COPY). Only after the LAPACK project was begun was it realized that there were operations like the matrix copy routine (GE_COPY), the computation of a norm of a matrix (GE_NORM), and the generation of Householder transformations (GEN_HOUSE) that occurred so often that it was wise to make separate routines for them.

A second group of these operations extended the functionality of some of the existing BLAS (e.g., AXPBY, WAXPBY, GER, SYR/HER, SPR/HPR, SYR2/HER2, SPR2/HPR2). For example, the Level 3 BLAS for the rank k update of a symmetric matrix only allows a positive update, which means that it cannot be used for the reduction of a symmetric matrix to tridiagonal form (to facilitate the computation of the eigensystem of a symmetric matrix), or for the factorization of a symmetric indefinite matrix, or for a quasi-Newton update in an optimization routine.

Other extensions (e.g., AXPY_DOT, GE_SUM_MV, GEMVT, TRMVT, GEMVER) perform two Level 1 BLAS (or Level 2 BLAS) routine calls simultaneously to increase performance by reducing memory traffic.

The original efforts to specify sparse Level 2 and 3 BLAS took considerably longer than the corresponding efforts for the dense and banded BLAS, principally because of the need to obtain agreement on the way to represent sparse matrices. The lessons learned from those efforts have provided a vital background to the specifications given in the BLAST document.

The original Level 2 BLAS included, as an appendix, the specification of extended precision subprograms. With both the widespread adoption of hardware supporting the IEEE extended arithmetic format (and other forms of extended precision arithmetic) and the increased understanding of algorithms to successfully exploit such arithmetic, it was felt that to be timely a complete specification for a set of extra precise BLAS should be included.

The specification of the original BLAS was given in the form of Fortran66 and subsequently Fortran77 subprograms. In the BLAST document, we provide specifications for Fortran95, Fortran77, and C. Reference implementations are also provided. Alternative language bindings for C++ and Java were also discussed during the meetings of the forum, but the specifications for these bindings were postponed for a future series of meetings.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/blast/

**E-mail**
blast@cs.utk.edu

**Related URLs**
LAPACK - http://www.netlib.org/lapack/index.html
BLAS - http://www.netlib.org/blas/

# F2J/ JLAPACK

**ICL TEAM** (ALPHABETICAL)
Jack Dongarra
Andrew Downey (G)
Keith Seymour
(G) = GRADUATE STUDENT

The goal of the JLAPACK project is to provide application programming interfaces (APIs) to numerical libraries from Java programs. The numerical libraries will be distributed as class files that are produced by a Fortran-to-Java translator called F2J. The F2J translator is a formal compiler that translates programs that are written using a subset of Fortran77 into a form that may be executed on Java virtual machines. The first priority for F2J is to translate the BLAS and LAPACK numerical libraries from their Fortran77 reference source code to Java class files. The subset of Fortran77 translated by F2J roughly matches the Fortran source used by BLAS and LAPACK. These libraries are established, reliable, and widely used linear algebra packages, which makes them a reasonable first testbed for F2J. Many other libraries of interest are expected to use a very similar subset of Fortran77.

The current release of JLAPACK consists of 288 double-precision routines translated from LAPACK and 32 double-precision routines from BLAS, totaling 109,438 lines of Java source code. Translated versions of the BLAS and LAPACK testers, totaling roughly 50,000 lines of Java source code, are also available for download.

With a goal of producing a Java implementation of JLAPACK, we have developed a tool to automate the translation. Doing so allows us to generate pure Java code in a consistent and reliable way from the original Fortran source. In addition, there is now a tool that can be applied successfully to other numerical libraries and eventually to a wide range of Fortran code.

The F2J compiler system was written in ANSI C, using a C parser generated by the Bison parser generator. Original code was written after determining that existing Fortran tools such as F2C and G77 would be difficult to modify. Similarly, the Bison grammar was derived from the Fortran77 standard since available parse files would have needed extensive rewriting to produce an abstract syntax tree (AST). The BLAS and LAPACK source code are assumed to be syntactically correct Fortran. Comments from the Fortran source are preserved in the translation as Java comments. The F2J executable generates both Java source code and Java Virtual Machine (JVM) bytecode (class files).

The F2J compiler operates in four stages:

### LEXING/PARSING

In this stage, the lexer separates the Fortran source code into tokens and the parser builds a complete AST as well as symbol tables for each program unit. Subsequent compilation stages obtain all information about the program structure from the AST built during parsing.

### OPTIMIZING THE USE OF SCALAR WRAPPERS

In Fortran, values are passed to functions and subroutines by reference. This implies that if a Fortran subroutine modifies one of its parameters, then that modification also takes effect in the calling routine. However, Java uses pass-by-value, which implies that modifications would not take effect in the caller. In order to simulate pass-by-reference in Java, the scalar must be wrapped in an object. Then, instead of passing the integer value, the object wrapper, whose scalar field may be modified in the subroutine, is passed.

The scalar "optimization" phase determines which parameters of each subroutine absolutely need to be wrapped. The rest are passed as Java primitive data types (int, double, etc.) in order to improve access times and save memory. Determination is made via three rules:

A variable must be wrapped if

1. The variable is an argument to this function and it is on the left side of an assignment statement in this program unit
2. The variable is an argument to this function and it is an argument to a READ statement
3. The variable is passed to a function/ sub-routine that modifies it

Rule 3 implies that every function/subroutine that the current program unit depends on must be checked before this unit can be completely checked. F2J resolves the dependencies before continuing to check the current unit.

### TYPE ASSIGNMENT

This stage is not "typechecking" in the semantic analysis sense because F2J is not attempting to determine the correctness of the Fortran code. In

this stage, F2J performs a traversal of the AST and assigns type information to each node, propagating information up the tree. For example, F2J looks at both sides of an addition operation and assigns the widest type to the addition node and so on up the tree. This information helps the code generator emit the appropriate type-specific opcodes and type casts when necessary.

### Code Generation

Code generation is by far the largest and most complicated stage in the translator. In this stage, F2J traverses the AST and generates code as it traverses down the tree. The code generator depends on the information determined in all the prior steps to generate correct code. Currently, Java source code and JVM bytecode are generated during the same pass because using separate passes would have resulted in a lot of duplicated code and made maintenance more difficult.

Each Fortran program unit is generated as a separate Java class containing a single static method. For example, the Fortran subroutine DGEMM would be translated to a Java class named Dgemm containing only a single method named dgemm. All arrays are arranged in column-major fashion, with 2D Fortran arrays being translated as linearized 1D Java arrays.

Fortran GOTO statements are easily translated to JVM bytecode since there exists a GOTO opcode. However, Java source code does not provide a GOTO statement. As a result some post-processing must be performed on the class files that were generated from Java source in order to correctly generate the GOTO statements.

For the next release, we are planning to generate all the routines directly as Java class files. Handling GOTO statements is easier when generating JVM bytecode as opposed to Java source because the resulting bytecode requires no post-processing to eliminate the GOTOs. In addition, translating directly from Fortran to bytecode allows generating operations on complex numbers as stack arithmetic rather than method calls. The bytecode generator is in the final stages of development and has already successfully translated the BLAS library and testers.

Since the nature of numerical computing demands fast execution, we also plan to implement an optimization phase in the F2J compiler to produce faster bytecode.

JLAPACK benefits Java application developers who require a wide range of linear algebra routines in a pure Java implementation. The availability of a Java implementation of LAPACK facilitates the creation of web-based numerical and engineering applications.

Thousands of users have downloaded the JLAPACK library for use in engineering, research, and teaching. Some users have also reported success in translating their own libraries using the F2J source code.

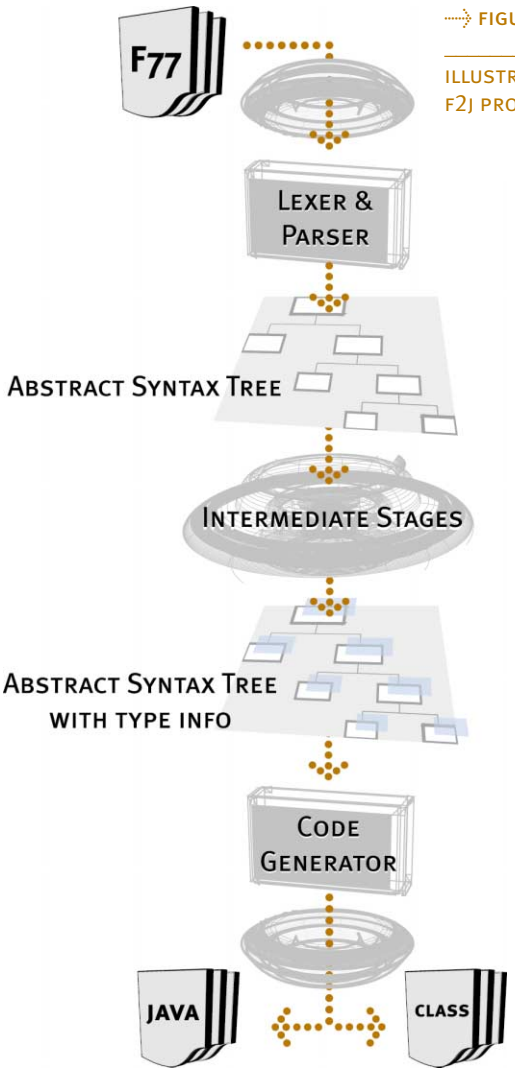



FIGURE 1.
ILLUSTRATION OF THE F2J PROCESS

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/f2j/

**E-mail**
f2j@cs.utk.edu

**Publications**
Dongarra, J., Doolin, D., Seymour, K. "JLAPACK: Compiling LAPACK FORTRAN to Java," *Scientific Programming*, 7 Number 2, (1999): 111-138.
http://www.cs.utk.edu/f2j/f2jreport/f2jreport.html

**Related URLs**
Bison - http://www.gnu.org/software/bison/bison.html
BLAS - http://www.netlib.org/blas/
LAPACK - http://www.netlib.org/lapack/

# LAPACK

## LINEAR ALGEBRA PACKAGE

**ICL TEAM (ALPHABETICAL)**

Susan Blackford

Jack Dongarra

**COLLABORATORS**

Numerical Algorithms Group, Ltd.

University of California, Berkeley

University of California, Davis

Rice University

LAPACK is a library of Fortran77 subroutines for solving the most commonly occurring problems in numerical linear algebra. It has been designed to provide high efficiency on vector processors, high-performance "super-scalar" workstations, and shared memory multiprocessors. It can also be used satisfactorily on all types of scalar machines (PCs, workstations, mainframes). A distributed-memory version of LAPACK, ScaLAPACK, has been developed for other types of parallel architectures (for example, massively parallel SIMD machines or distributed memory machines). The name LAPACK is an acronym for Linear Algebra PACKage.

LAPACK can solve systems of linear equations, linear least squares problems, eigenvalue problems, and singular value problems. LAPACK can also handle many associated computations such as matrix factorizations or estimating condition numbers. Dense and band matrices are provided for, but not general sparse matrices. In all areas, similar functionality is provided for real and complex matrices.

The original goal of the LAPACK project was to make the widely used LINPACK and EISPACK libraries run efficiently on shared-memory vector and parallel processors. On these machines, LINPACK and EISPACK are inefficient because their memory access patterns disregard the multi-layered memory hierarchies of the machines, thereby spending too much time moving data instead of doing useful floating-point operations. LAPACK addresses this problem by reorganizing the algorithms to use block matrix operations, such as matrix multiplication, in the innermost loops. These block operations can be optimized for each architecture to account for the memory hierarchy, and thus provide a portable way to achieve high efficiency on diverse modern machines. LAPACK also adds extra functionality that was not available in LINPACK or EISPACK, uses some new or improved algorithms, and integrates the two sets of algorithms (LINPACK and EISPACK) into a unified package.

The Basic Linear Algebra Subprograms (BLAS) are a set of such kernel routines for linear algebra. The Level 1 BLAS were proposed in 1973-77 to specify scalar and vector operations, and were extensively and successfully exploited in LINPACK. The Level 2 and 3 BLAS specify matrix-vector and matrix-matrix operations, respectively, and were presented in 1984-86 and 1987-88. LAPACK routines are written so that as much as possible of the computation is performed by calls to the BLAS. Highly efficient machine-specific implementations of the BLAS are available for many modern high-performance computers. If a machine-specific implementation is not available, ATLAS can be used to generate an optimized BLAS library for your computer.

The first public release (version 1.0) of the LAPACK software library occurred on February 29, 1992. Version 2.0 was released in September 1994, and Version 3.0 was released in June 1999. LAPACK continues to expand into an even wider community effort.

Version 3.0 of LAPACK introduced new routines and extended the functionality of existing routines. The most significant new routines and functions were

1. a faster singular value decomposition (SVD), computed by divide-and-conquer (xGESDD)
2. faster routines for solving rank-deficient least squares problems
   - using QR with column pivoting (xGELSY, based on xGEQP3)
   - using the SVD based on divide-and-conquer (xGELSD)
3. new routines for the generalized symmetric eigenproblem
   - xHEGVD/xSYGVD, xHPGVD/xSPGVD, xHBGVD/xSBGVD: faster routines based on divide-and-conquer
   - xHEGVX/xSYGVX, xHPGVX/xSPGVX, xHBGVX/xSBGVX: routines based on bisection/inverse iteration to more efficiently compute a subset of the eigenvalues and/or eigenvectors
4. faster routines for the symmetric eigenproblem using the "relative robust representation" algorithm (xSYEVR/xHEEVR, xSTEVR, xSTEGR)
5. new simple and expert drivers for the generalized nonsymmetric eigenproblem (xGGES, xGGEV, xGGESX, xGGEVX), including error bounds
6. a solver for the generalized Sylvester equation

(xTGSYL), used in 5.

7. computational routines (xTGEXC, xTGSEN, xTGSNA) used in 5
8. a blocked version of xTZRQF (xTZRZF), and associated xORMRZ/xUNMRZ

This release of LAPACK introduced routines that exploit IEEE arithmetic. We have a prototype running of a new algorithm (xSTEGR), which may be the ultimate solution for the symmetric eigenproblem on both parallel and serial machines. This algorithm has been incorporated into the drivers xSYEVR, xHEEVR, and xSTEVR for the symmetric eigenproblem, and will be propagated into the generalized symmetric definite eigenvalue problems, the SVD, the generalized SVD and the SVD-based least squares solver. Refer to the LAPACK Users' Guide for further information. We expect to also propagate this algorithm into ScaLAPACK.

The original LAPACK project was funded by the National Science Foundation (NSF). Since its completion, four follow-up projects, LAPACK 2, ScaLAPACK, ScaLAPACK 2, and LAPACK 3 have been funded in the U.S. by the NSF and the Defense Advanced Research Project Agency (DARPA).

The complete LAPACK package is freely available on Netlib and can be obtained via the World Wide Web or anonymous FTP. General information about LAPACK (and the BLAS) can be obtained from the LAPACK web site.

One of the primary design features of the LAPACK library is that all releases are backward compatible. A user's program calling LAPACK will never fail because of a new release of the library. The complete package, including test code and timing programs in four different Fortran data types, constitutes some 805,000 lines of Fortran source and comments.

Alternative language interfaces to LAPACK (or translations/conversions of LAPACK) are available in Fortran95, C, and Java.

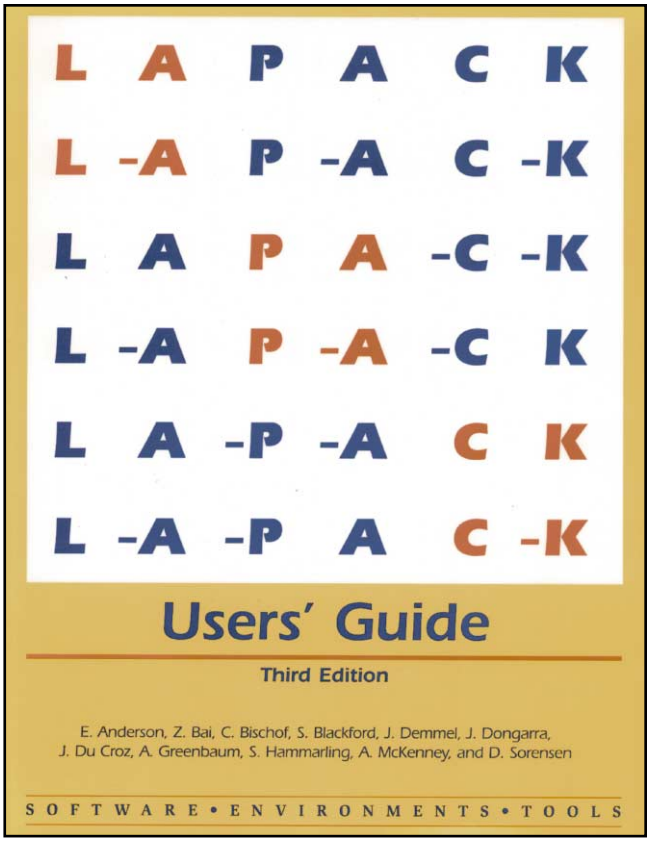



FIGURE 1.

COVER OF LAPACK USERS' GUIDE
IMAGE COURTESY OF SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS (SIAM)

The LAPACK Users' Guide provides an informal introduction to the design of the package, a detailed description of its contents, and a reference manual for the leading comments of the source code.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/lapack/

**E-mail**
lapack@cs.utk.edu

**Publications**
Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D., *LAPACK Users' Guide*, 3rd ed., Philadelphia: Society for Industrial and Applied Mathematics, 1999.

**Agency Funding**
Defense Advanced Research Projects Agency (DARPA)
Department of Energy (DOE)
National Science Foundation (NSF)

**Related URLs**
ATLAS - http://www.netlib.org/atlas/
BLAS - http://www.netlib.org/blas/faq.html
CLAPACK - http://www.netlib.org/clapack/
JavaLAPACK - http://www.netlib.org/java/f2j/
LAPACK 95 - http://www.netlib.org/lapack95/
Users' Guide - http://www.netlib.org/lapack/lug/

**FAQ**
http://www.netlib.org/lapack/faq.html

# ScaLAPACK

## SCALABLE LINEAR ALGEBRA PACKAGE

**ICL TEAM (ALPHABETICAL)**

Susan Blackford

Jack Dongarra

Antoine Petitet

Clint Whaley

**COLLABORATORS**

Intel Corporation

Numerical Algorithms Group, Ltd.

Oak Ridge National Laboratory

Soongsil University - Seoul, South Korea

University of California, Berkeley

The name ScaLAPACK is an acronym for Scalable Linear Algebra PACKage, or Scalable LAPACK. ScaLAPACK is a library of high-performance linear algebra routines for distributed-memory, message-passing MIMD computers and networks of workstations supporting MPI and/or PVM. It is a continuation of the LAPACK project, which designed and produced analogous software for workstations, vector supercomputers, and shared-memory parallel computers.

Both LAPACK and ScaLAPACK contain routines for solving systems of linear equations, least squares problems, and eigenvalue problems. The goals of both projects are efficiency (to run as fast as possible), scalability (as the problem size and number of processors grow), reliability (including error bounds), portability (across all important parallel machines), flexibility (so users can construct new routines from well-designed parts), and ease of use (by making the interface to LAPACK and ScaLAPACK look as similar as possible). Many of these goals, particularly portability, are aided by developing and promoting standards, especially for low-level communication and computation routines. We have been successful in attaining these goals, limiting most machine dependencies to two standard libraries called the BLAS, or Basic Linear Algebra Subprograms, and BLACS, or Basic Linear Algebra Communication Subprograms.

LAPACK will run on any machine where the BLAS are available, and ScaLAPACK will run on any machine where both the BLAS and the BLACS are available.

The ScaLAPACK library is currently written in Fortran77 (with the exception of a few symmetric eigenproblem auxiliary routines written in C to exploit IEEE arithmetic) in a Single Program Multiple Data (SPMD) style using explicit message passing for interprocessor communication.

ScaLAPACK routines are available in four types; single precision real, double precision real, single precision complex, and double precision complex.

The complete ScaLAPACK package, including test code and timing programs in the four different data types, constitutes some 500,000 lines of Fortran and C source and comments.

ScaLAPACK includes routines for solving the following:

- linear systems of equations
- general and symmetric positive definite band linear systems of equations
- general and symmetric positive definite tridiagonal linear systems of equations
- condition estimation and iterative refinement for LU and Cholesky factorization
- matrix inversion
- full-rank linear least squares problems
- orthogonal and generalized orthogonal factorizations
- orthogonal transformation routines
- reductions to upper Hessenberg, bidiagonal and tridiagonal form
- reduction of a symmetric-definite generalized eigenproblem to standard form
- the symmetric/Hermitian eigenproblem
- the generalized symmetric/Hermitian eigenproblem
- the nonsymmetric eigenproblem
- the singular value decomposition

Like LAPACK, the ScaLAPACK routines are based on block-partitioned algorithms in order to minimize the frequency of data movement between different levels of the memory hierarchy. The fundamental building blocks of the ScaLAPACK library are distributed memory versions of the Level 1, 2, and 3 BLAS, called the Parallel BLAS or PBLAS, and a set of BLACS for communication tasks that frequently occur in parallel linear algebra computations. In the ScaLAPACK routines, the majority of interprocessor communication occurs within the PBLAS, so the source code of the top software layer of ScaLAPACK looks similar to that of LAPACK.

The ScaLAPACK library for dense linear algebra computations is in the process of transition to the commercial marketplace. ScaLAPACK has been incorporated into several commercial packages, including the NAG Parallel Library, IBM Parallel ESSL, and SGI Cray Scientific Software Library, and is being integrated into the VNI IMSL Numerical Library, as well as software libraries for Fujitsu, Hewlett-Packard/Convex, Hitachi, and NEC.

The ScaLAPACK library has become an official

release for DOE's ASCI Red's operating system. Each build of the operating system will be validated against ScaLAPACK and each new compiler and operating system drop will contain an automatically generated fresh ScaLAPACK build. It will be in /usr/lib, as standard as the BLAS.

The ScaLAPACK generalized Hermitian eigensolver has been enhanced and has been incorporated into the electron structure MP-Quest project at Sandia National Laboratory. It is approximately ten times faster than the previously existing eigensolver in MP-Quest, and will result in an approximate 90% savings in cycles on the ASCI Red computer. Performance enhancements to the eigencode are being propagated into the next ScaLAPACK release.

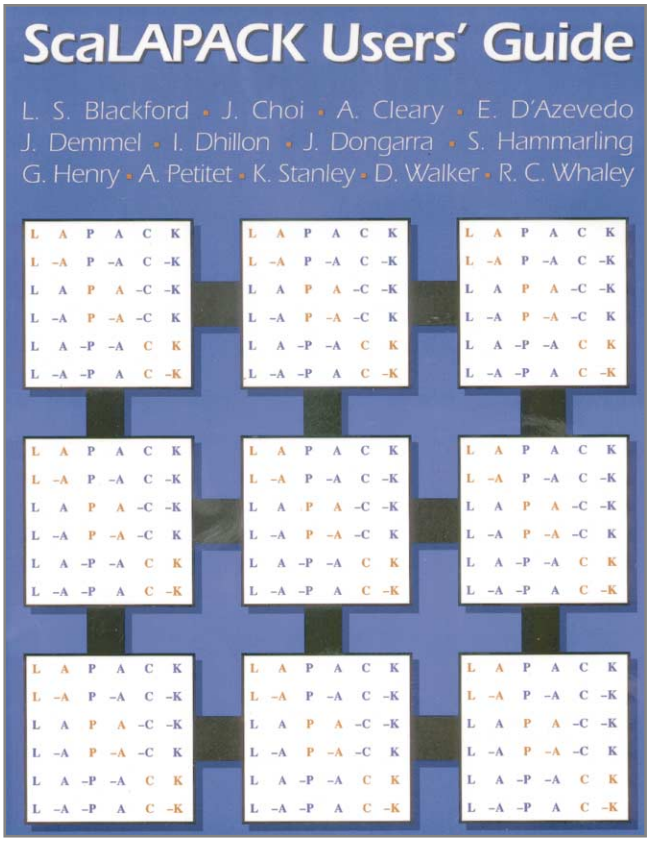



FIGURE1.

COVER OF
SCALAPACK USERS' GUIDE
IMAGE COURTESY OF SOCIETY FOR INDUSTRIAL AND APPLIED MATHEMATICS (SIAM)

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/scalapack/

**E-mail**
scalapack@cs.utk.edu

**Publications**
Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D. *LAPACK Users' Guide*, 3rd ed., Philadelphia: Society for Industrial and Applied Mathematics, 1999.

**Agency Funding**
Defense Advanced Research Projects Agency (DARPA)
Department of Energy (DOE)
National Science Foundation (NSF)

**Related URLs**
DOE ASCI Red - http://www.sandia.gov/ASCI/Red/
IBM Parallel ESSL -
http://www.rs6000.ibm.com/software/sp_products/esslpara.html
NAG Parallel Library -
http://www.nag.co.uk/numeric/fd/FDdescription.asp
SGI Cray Scientific Software Library -
http://www.sgi.com/software/scsl.html
VNI IMSL Numerical Library -
http://www.vni.com/products/imsl/index.html

**FAQ**
http://www.netlib.org/scalapack/faq.html

# FT-MPI

## FAULT-TOLERANT MESSAGE PASSING INTERFACE

ICL TEAM (ALPHABETICAL)
Jack Dongarra
Graham Fagg

The initial version of the Message Passing Interface (MPI) standard was designed to work efficiently on Massively Parallel multi-Processors (MPPs), which had very little job control and thus a static process model. Later versions of the MPI standard incorporated some dynamic process control, but did not allow for node/process failures to be tolerated. As high performance computing (HPC) systems increase in size with higher potential levels of individual node failure, the need for new fault tolerant applications to be developed increases. Currently, fault tolerant applications cannot be built with MPI because MPI is unable to handle failures gracefully. FT-MPI is a developmental implementation of MPI that allows fault tolerant applications to be built. Under FT-MPI, applications control how failures are handled by either the message passing layer or the application itself.

The initial version of the MPI standard specified that if a process failed the communicator (of which the process belonged) became invalid and could no longer be used for communication. The only recovery method from such a failure was to abort the rest of the application regardless of how long it had been running. Check-pointing and regular saving of the application's state could alleviate these problems, and for some application classes, it still can.

Such semantics for failure are suitable for highly stable small to medium sized systems where failures are infrequent. Beyond these sizes, the mean time between failures (MTBF) of nodes becomes a factor. As attempts to build the next generation Petaflop systems advance, this situation is likely to become worse. Individual node reliability decreases with an increase in node numbers, which leads to higher possibilities of node failures.

Although FT-MPI can allow for automatic restart of applications in the case of failures by co-operating with checkpoint libraries, it is really meant to allow development of algorithm level, fault tolerant applications. In other words, some applications adapt to failures by changing their algorithms, such as applications that allow for reduced grids in numeric solvers. Building such applications using present implementations of MPI is not currently possible. FT-MPI supports the execution of current MPI applications without modification by providing a subset of the MPI-1 and MPI-2 application programming interfaces (APIs).

Current semantics of MPI indicate that a failure of an MPI process or communication causes all communicators associated with them to become invalid. Since the standard provides no method to reinstate the communicators (it is even unclear if they can be freed), MPI_COMM_WORLD itself becomes invalid, thus causing the entire MPI application to shut down.

FT-MPI extends the MPI communicator states from {valid, invalid} to a range {FT_OK, FT_DETECTED, FT_RECOVER, FT_RECOVERED, FT_FAILED}. In effect, this becomes {OK, PROBLEM, FAILED} with the other states mainly of interest to the internal fault recovery algorithm of FT_MPI. Processes also have typical states of {OK, FAILED}, which FT-MPI replaces with {OK, UNAVAILABLE, JOINING, FAILED}. The UNAVAILABLE state includes unknown, unreachable, or "we have not voted to remove it yet" substates. A communicator changes its state when either an MPI process changes its state or a communication within that communicator fails for some reason. The typical MPI semantics are from OK to FAILED, which then causes an application to abort. By allowing the communicator to be in an intermediate state, we allow the application the ability to decide how to alter the communicator and its state, as well as how communication within the intermediate state behaves.

When a failure occurs, the user application can tell the message-passing layer how to handle it or it can handle the failure itself. The application can specify failure modes that determine whether communicators are reformed or destroyed in the event of a failure, and whether communications can continue until the application reaches a state in which it can more fully address the failure.

Failure modes for communicators are as follows:

- SHRINK: The communicator is shrunk so that there are no holes in its data structures. The ranks of the processes are changed, forcing the application to recall MPI_COMM_RANK.
- BLANK: This is the same as SHRINK, except that the communicator can now contain gaps to be

filled in later. Communicating with a gap will cause an invalid rank error. Note also that calling MPI_COMM_SIZE will return the size of the communicator and not the number of valid processes within it.

- REBUILD: It is the most complex because it forces the creation of new processes to fill any gaps. The new processes can either be placed into the empty ranks or the communicator can be shrunk and the processes appended to the end. This is used by applications that require a certain size to execute, i.e., power of two FFT solvers.
- ABORT: This is a mode where the application (on error detection) forces a graceful abort. The user can not trap this, and the only option is to change the communicator mode to one of the above modes.

Communications within the communicator are controlled by a message mode for the communicator, which can be either of the following:

- NOP: No operations allowed on error, i.e., no user level message operation is allowed and all simply return an error code. This is used to allow an application to return from any point in the code to a state where it can take appropriate action as soon as possible.
- CONT: All communication that is NOT directed to the effected/failed node can continue as normal. Attempts to communicate with a failed node will return errors until the communicator state is reset.

Typical usage of FT-MPI would be in the form of an error check and then some corrective action such as a communicator rebuild. A typical code fragment is shown below, where in the event of an error the communicator is simply rebuilt and reused:

```
rc= MPI_Send (----, com);
If (rc==MPI_ERR_OTHER)
    MPI_Comm_dup (com, newcom);
    com = newcom;       /* continue.. */
```

Some types of computations such as SPMD master-slave codes only need the error checking in the master code if the user is willing to accept the master as the only point of failure.

The FT-MPI system was developed from scratch as a high performance MPI implementation that is designed to execute on top of the HARNESS MetaComputing environment. The system includes an advanced buffer management scheme for handling complex, user derived data types (DDTs), a self-tuning collective communications library, and a complex, multi-threaded message passing subsystem that supports TCP/UDP, Myricom GM, and SystemV shared memory concurrently.

FT-MPI allows message passing applications to be built that can survive node failures without the need to continuously make expensive state and checkpoint dumps to disk. It also allows for new classes of algorithms to be developed that can adapt to failures, which is currently not possible with other implementations of MPI.

FT-MPI has been used by the developers of the Parallel Spectral Transformation Shallow Water Model (PSTSWM) at Oak Ridge National Laboratory (ORNL) to implement a new version of this complex parallel code that automatically repairs itself during failures without external intervention. Developers of the widely used Parallel Climate Community Model (PCCM) are considering modifying their code to use FT-MPI to implement a reduced grid algorithm for handling failures.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/ftmpi/

**E-mail**
ftmpi@cs.utk.edu

**Agency Funding**
Department of Energy (DOE)

**Publications**
Dongarra, J., Fagg, G. "FT-MPI: Fault Tolerant MPI, Supporting Dynamic Applications in a Dynamic World", *Lecture Notes in Computer Science: Proceedings of the EuroPVM-MPI 2000*, (Hungary: Springer Verlag, 2000), Vol. 1908, 346-353.
http://www.netlib.org/utk/people/JackDongarra/PAPERS/ft-mpi.pdf

**Related URLs**
HARNESS - http://www.epm.ornl.gov/harness/
Parallel Climate Community Model (PCCM) - http://www.epm.ornl.gov/chammp/pccm2.1/
Parallel Spectral Transformation Shallow Water Model (PSTSWM) - http://www.epm.ornl.gov/chammp/pstswm/

# GrADS

## GRID APPLICATION DEVELOPMENT SYSTEM

**ICL TEAM (ALPHABETICAL)**

Susan Blackford
Jack Dongarra
Brett Ellis
Antoine Petitet

**COLLABORATORS**

Information Sciences Institute
University of California, San Diego
University of Chicago
University of Houston
University of Illinois at Urbana-Champaign
University of Indiana
University of Southern California
University of Tennessee
Rice University

Advances in networking technologies will soon make it possible to use the global information infrastructure in a qualitatively different way—as a computational resource as well as an information resource. This idea for an integrated computation and information resource called the Computational Power Grid has been described in the recent book entitled *The Grid: Blueprint for a New Computing Infrastructure*. The Grid will connect the nation's computers, databases, instruments, and people in a seamless web of computing and distributed intelligence that can be used as a problem-solving resource in many fields of human endeavor, particularly for science and engineering. To realize this vision, we must overcome significant scientific and technical obstacles. Principal among these is usability. Because the Grid will be inherently more complex than existing compute systems, programs that execute on the Grid will reflect some of this complexity. Hence, making Grid resources useful and accessible to scientists and engineers will require new software tools that embody major advances in both the theory and practice of building Grid applications.

The goal of the Grid Application Development System (GrADS) is to simplify distributed heterogeneous computing in the same way that the World Wide Web simplified information sharing over the Internet. GrADS will exploit the scientific and technical problems that must be solved to make it relatively easy to develop Grid applications for real problems and to tune their performance. This will require research in five key areas, each validated in a prototype infrastructure:

- Grid software architectures that facilitate information flow and resource negotiation among applications, libraries, compilers, linkers, and runtime systems
- Base software technologies, such as scheduling, resource discovery, and communication, to support development and execution of performance-efficient Grid applications
- Languages, compilers, environments, and tools to support the creation of applications and problem-solving environments for the Grid
- Mathematical and data structure libraries for Grid applications, including numerical methods for control of accuracy and latency tolerance
- Innovative new science and engineering applications that can take advantage of these new technologies to run effectively in a Grid environment

A team of thirteen principal investigators from the following universities is conducting the research: RiceUniversity, University of California San Diego, University of Chicago, University of Houston, University of Illinois at Urbana-Champaign, University of Indiana, University of Southern California, and Univeristy of Tennessee. Technology transfer in GrADS will be via two principal mechanisms. First, we are working closely with a group of industrial collaborators to encourage the adoption and standardization of system software technologies that arise from the research. Second, we are working directly with application developers through the two NSF PACIs: NPACI and the Alliance, and through the NASA IPG and ASCI ASAP programs. GrADS is fostering research, education, and technology transfer programs that are contributing to evolutionary new ways of utilizing the global information infrastructure as a platform for computation, changing the way scientists and engineers solve their everyday problems.

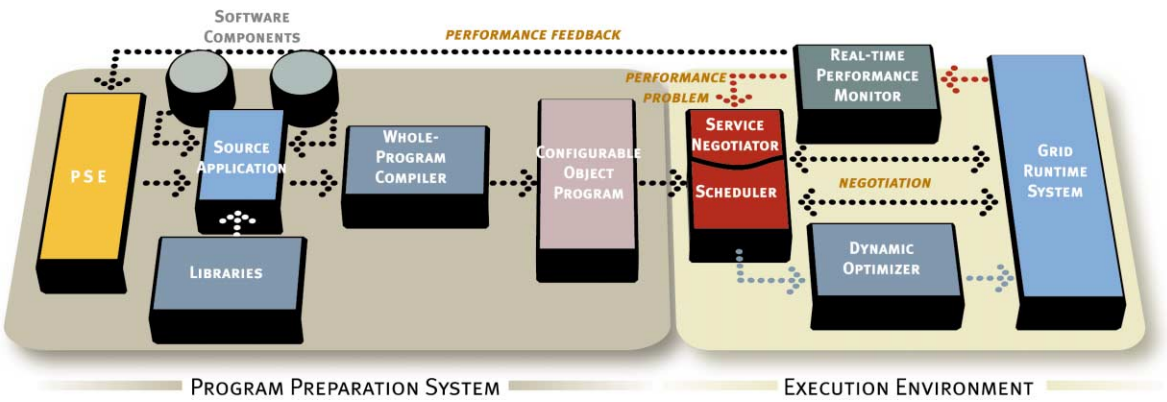



FIGURE 1.

ILLUSTRATION OF THE GrADS GRID COMPILATION ARCHITECTURE

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/grads/

**E-mail**
grads@cs.utk.edu

**Publications**
Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., Johnnson, L., Kennedy, K., Kesselman, C., Reed, D., Torczon, L., Wolski, R. "The GrADS Project: Software Support for High-Level Grid Application Development," *Technical Report*, February 2000.
http://www.hipersoft.rice.edu/grads/tr/grads_project.html

**Agency Funding**
National Science Foundation (NSF)

**Related URLs**
NSF Alliance - http://www.ncsa.edu/
NSF NPACI - http://www.npaci.edu/

# HARNESS

**HETEROGENEOUS ADAPTABLE RECONFIGURABLE NETWORKED SYSTEMS**

**ICL TEAM (ALPHABETICAL)**

Jack Dongarra

Graham Fagg

Keith Moore

**COLLABORATORS**

Emory University

Oak Ridge National Laboratory

HARNESS (Heterogeneous Adaptable Reconfigurable Networked SystemS) is an experimental metacomputing framework built around the services of a highly customizable and reconfigurable distributed virtual machine (DVM). A DVM is a tightly coupled computation and resource grid that provides a flexible environment to manage and coordinate parallel application execution.

The system is designed to support a wide range of DVM sizes, from users building personal DVMs to enterprise and widely distributed DVMs. Collaboration and resource sharing between different entities is performed by the temporary merging and splitting of different DVMs.

HARNESS seeks to remove the limitations discovered in the Parallel Virtual Machine (PVM) system and create a completely different approach to building, modifying, and using multiple DVMs to facilitate a new generation of collaborative and computation environments.

Virtual machine (VM) terminology, borrowed from PVM, refers to a system where the computing resources on that system can be viewed as a single, large, distributed memory computing resource. This abstraction was once very powerful and widely accepted, but the PVM system itself only allowed for a single VM, which limited collaboration, and the code was monolithic, which made it difficult to add new functionality. PVM's single VM also had a single point of failure through which it maintained a master database that inevitably limited its scalability.

HARNESS was designed to support

- multiple DVMs
- no single point of failure
- scalable infrastructure though replicated state
- flexible/reconfigurable functionality by allowing new components or plug-ins to be added and removed dynamically
- Legacy message passing code

The architecture is built on kernels, daemons, DVMs, and services provided by standard components. The kernel is implemented as a set of core functions for loading and running components either locally or via remote requests. A HARNESS daemon is composed of a kernel (the HARNESS Core or Hcore) and a minimal set of required components to provide basic services. These services include maintaining state, the ability to communicate between components, and remote invocation of components and new daemons. A HARNESS DVM is composed of a set of co-operating daemons that together present the basic services of communication, process control, resource management, and fault detection. Important goals of HARNESS are that it should be robust and reliable. All information and control within HARNESS are to be built on a Symmetric Peer-to-Peer Distributed Control (SPDC) algorithm; thus, HARNESS will not have a single point of failure, but instead a user configurable level of fault tolerance.

Figures 1 and 2 show the design of a daemon built on a kernel (Hcore), then how the daemons interconnect to form a DVM. A HARNESS DVM is made from multiple HARNESS daemons executing on multiple hosts.

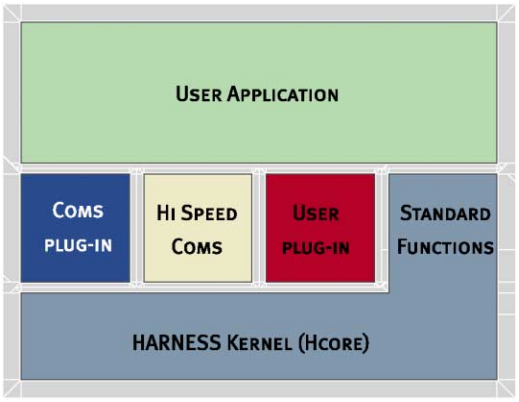



FIGURE 1.

ILLUSTRATION OF A DAEMON BUILT ON A KERNEL

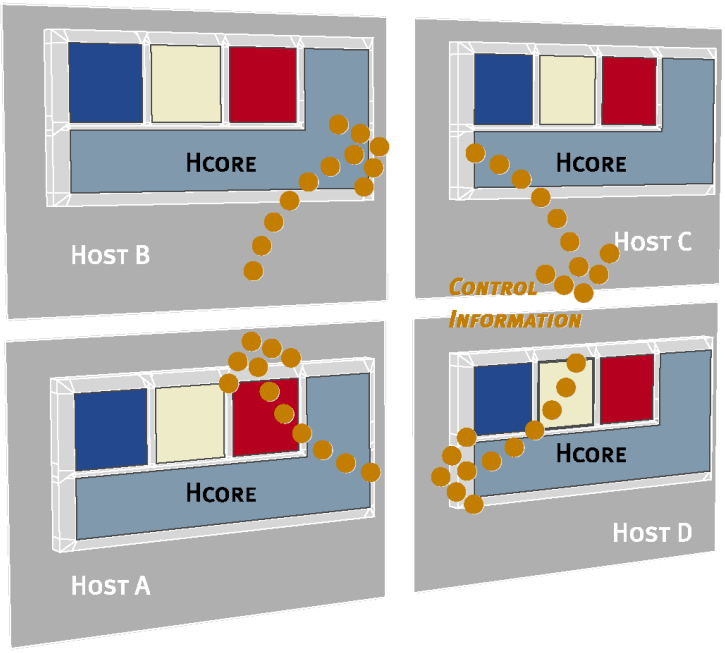



FIGURE 2.

ILLUSTRATION OF DAEMONS INTERCONNECTING TO FORM A DVM

Emory University has produced a Java-based implementation using JNI/RMI and dynamic loading. Both UT/ICL and Oak Ridge National Laboratory (ORNL) have produced C-based HARNESS cores that support dynamic libraries and shared objects as component plug-ins. Support for inter-operation between the Java and C-based systems has also been developed.

Two plug-ins have been developed to allow standard message passing codes to be used directly, and they support both the PVM 3.X and a subset of MPI-2 APIs. These plug-ins allow users of existing applications to run on HARNESS without any code modification. The MPI plug-in, known as FT-MPI, provides additional functionality to support fault-tolerant applications.

Using the PVM and MPI plug-ins allows for thousands of existing message passing applications to execute under the HARNESS system. The HARNESS system itself is expected to be used as a natural upgrade path for existing PVM users as well as users wishing to build their own personal computational grids without having to buy into some global grid framework.

The HARNESS system provides a number of benefits over existing VM-based systems, such as reliability, scalability, ease of adding new functionality, and the sharing of resources by merging DVMs.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/harness/

**E-mail**
harness@cs.utk.edu

**Publications**
Dongarra, J., Geist, A., Kohl, J., Papadopoulos, P., Scott, S., Sunderam, V. "HARNESS: Heterogeneous Adaptable Reconfigurable Networked Systems," March 1998. http://www.epm.ornl.gov/harness/hpdc.ps

**Agency Funding**
Department of Energy (DOE)

# I2-DSI

## INTERNET2 DISTRIBUTED STORAGE INFRASTRUCTURE

**ICL TEAM (ALPHABETICAL)**

Micah Beck

Chaoyang Liu (G)

Terry Moore

(G) = GRADUATE STUDENT

**COLLABORATORS**

EROS Data Center

IBM

International Center for Advanced Internet Research

Lokomo Systems

National University of Singapore

North Carolina Supercomputing Center

University of Hawaii at Manoa

University of Indiana

University of North Carolina at Chapel Hill

Texas A & M University

The Internet2 Distributed Storage Infrastructure (I2-DSI) project is a research effort designed to explore innovative uses of network storage in next generation applications and wide area information systems. The project began in the fall of 1998 with the cooperation of the University Corporation for Advanced Internet Development (UCAID) and the Internet2 community. Its ultimate goal is to develop a next generation service platform that can help both the research and education communites fulfill their traditional missions during the next decade. To this end, I2-DSI aims to develop a reliable, scalable, high performance storage infrastructure that advanced applications can use to exploit the power of the Internet2 campus and backbone networks and overcome limitations inherent in the current World Wide Web architecture.

The underlying idea of I2-DSI's storage-based approach is as follows: By combining intelligent replication of content and services with the transparent resolution of client requests to the "nearest" virtual host, I2-DSI can give the I2 community easy, high-performance access to types of content and modes of service delivery that are now available only in "proof of concept" form. In some important respects, I2-DSI's approach is similar to recent commercial efforts from companies like Akamai and Digital Island, which have capitalized on the idea of using advanced forms of traditional web caching to reduce bandwidth consumption, distribute server load, and improve performance in the distribution of typical Web content. But I2-DSI seeks to generalize content distribution beyond the current cache-based model, using replication in a way that allows objects like databases and executable content to be staged on and served from a variety of different platforms.

With sponsorship from Internet2 corporate partners and the cooperation of several universities in the Internet2 community, I2-DSI has established a national testbed to experiment with this new approach. Building on a major donation of large storage servers from IBM, along with contributions from StorageTek, Novell, Cisco Systems, Ellemtel (now Ericsson), Starburst (acquired by Adero), and Sun Microsystems, this testbed now has servers in six different locations in the USA, as shown in figure 1. There is also strong international interest in such innovative uses of network storage. This year both Singapore and Australia will add the first international nodes to I2-DSI.

Digital content and services are distributed on I2-DSI in the form of channels. Channels are I2-DSI's units of replication. A channel in this sense is a collection of content and associated services that can be transparently delivered to end user communities at a chosen cost/performance point through a flexible, policy-based application of resources. What such a channel can contain is a superset of what can be accessed via Web protocols. This includes not only text, graphics, Java applets, and multi-media, but also services that utilize non-Web protocols. Experimental channels already deployed on the current testbed include

- CPAN – Comprehensive Pearl Archive Network, which aims to contain all the Perl material that a member of the community will ever need (http://cpan.dsi.internet2.edu)
- Open Video – An expanding collection of public domain video content for the information retrieval and digital library research communities (http://openvideo.dsi.internet2.edu)
- Docsouth – Documenting the American South (DAS) is a collection of sources on Southern history, literature and culture from the colonial period through the first decades of the 20th century (http://docsouth.dsi.internet2.edu)
- MetaLab Linux – The comprehensive Linux Archives maintained by the UNC MetaLab (formerly the original SunSITE) (http://linux.dsi.internet2.edu)
- High MPEG – High bandwidth MPEG-1 videos for local streaming delivery (http://highmpeg.dsi.internet2.edu)

To make it possible to replicate such channels in a scalable way, the project has had to address the technical challenge of automating the replication process even when the underlying hardware/software platforms on the different system nodes vary. Today, such push-button replication of Web sites is nearly impossible. The current lack of standards for Web servers has limited the use of replication either

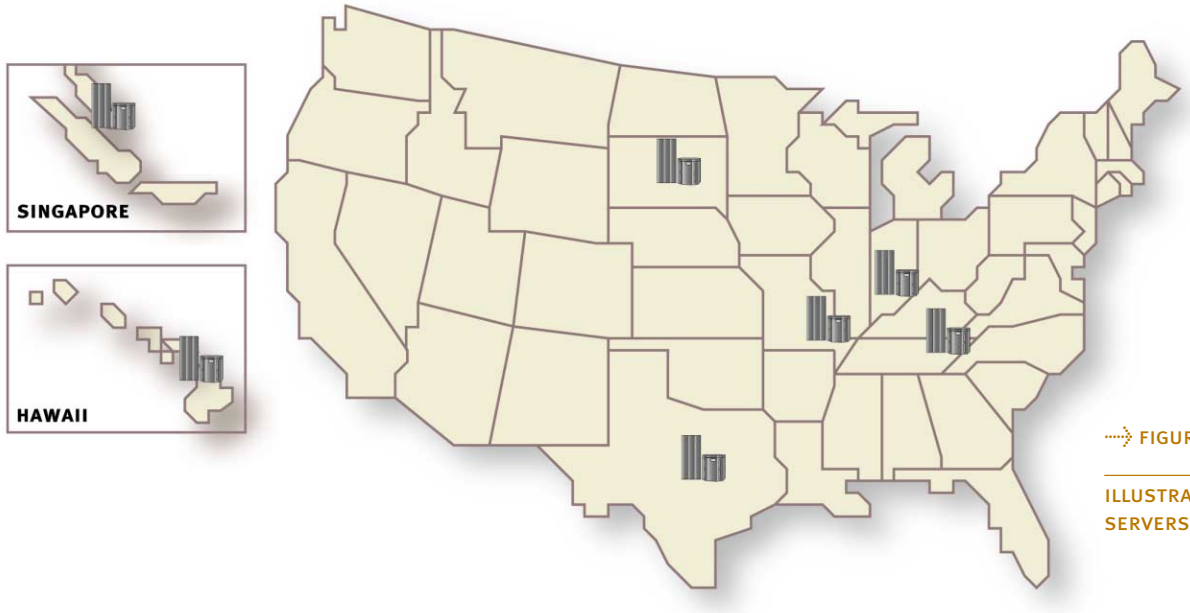



FIGURE 1.

ILLUSTRATION OF SERVERS' LOCATIONS

to cases that support only a common subset of server features or to those that can guarantee identical server platforms on all the nodes. In response to this problem, we have developed a data model for channels, called the Portable Channel Representation (PCR), which explicitly encodes all the technical metadata necessary to automate the process of installing and configuring a given channel on a given type of hardware/software platform. PCR will be promulgated as an open standard for creating replicable content channels, and I2-DSI is collaborating with private industry to develop tools that implement it and make it easy to use.

The design and development of PCR is just one example of the kind of collaboration among academic organizations and between academia and industry that have marked I2-DSI from its inception. I2-DSI's corporate sponsors have contributed not only equipment and funding, but also developers who have participated directly in the work. The initial academic collaboration between ICL here at the University of Tennessee and the School of Information Science at UNC Chapel Hill grew to include the EROS Data Center, the North Carolina Supercomputing Center, the University of Hawaii at Manoa, Indiana University, and several other people and organizations in the Internet2 community. In the summer of 2000, the National Science Foundation's Advanced Network Infrastructure program added its support to this collective effort, awarding a three year grant for almost one million dollars to the I2-DSI team.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/dsi/

**E-mail**
dsi@cs.utk.edu

**Publications**
Beck, M., Chawla, R., Dempsey, B., Moore. T. "Portable Representation of Internet Content Channels in I2-DSI," *4th Intl. Web Caching Workshop*, San Diego, March 31-April 2, 1999.

Beck, M., Moore, T. "The Internet2 Distributed Storage Infrastructure Project: An Architecture for Internet Content Channels," *Computer Networking and ISDN Systems*, 1998, 30 (22-23): pp. 2141-2148.

Dempsey, B., Weiss, D. "Towards An Efficient, Scalable Replication Mechanism for the I2-DSI Project," University of North Carolina School of Library and Information Science *Technical Report*, TR-1999-01, April 1999.

**Agency Funding**
National Science Foundation (NSF)

**Industry Support**
Cisco Systems, Inc.
Ericcson
IBM
Internet2
Novell
Starburst Multicast

**Related URLs**
Akamai - http://www.akamai.com/
Digital Island - http://www.digitalisland.com/
UCAID - http://www.ucaid.edu/ucaid/

# IBP

## INTERNET BACKPLANE PROTOCOL

**ICL TEAM (ALPHABETICAL)**

Micah Beck

Alex Bassi

Balajee Kannan (G)

Susan Ling (G)

Terry Moore

(G) = GRADUATE STUDENT

**COLLABORATORS**

Profs. Jim Plank and Rich Wolski,
UT CS Department
University of California, San Diego

The Internet Backplane Protocol (IBP) is middleware for managing and using remote storage. It was invented to support an approach to building large scale, distributed applications called *logistical networking*. *Logistical networking* is the global scheduling and optimization of data movement, storage, and computation, using a model that takes into account all the network's underlying physical resources. It contrasts with more traditional approaches to networking, which do not explicitly model storage or computation as shared network resources. We call this approach "logistical" because of the analogy it bears with the systems of warehouses, depots, and distribution channels commonly used in the logistics of military and industrial activities. IBP provides a mechanism for using distributed storage for logistical purposes.

IBP was designed to enable applications to treat the Internet as if it were a processor backplane. Whereas on a typical computer backplane, the programmer has access to memory and peripherals and can direct communication between them, IBP gives the programmer access to remote storage and standard Internet resources (e.g., content servers implemented with standard network sockets) and can direct communication between them with the IBP application programming interface (API).

IBP represents the kind of middleware needed to overcome the current balkanization of storage management capabilities on the Internet. By providing a uniform, application-independent interface to storage in the network, IBP makes it possible for applications of all kinds to use *logistical networking* to exploit data locality and more effectively manage buffer resources. This allows any application that needs to manage distributed state to benefit from the kind of standardization, interoperability, and scalability that have made the Internet such a powerful communication tool.

IBP by itself is a very primitive enhancement to the functionality of the network, playing a role for data storage analogous to the role played by Internet Protocol (IP) for data transmission. As a primitive layer of storage functionality, it provides no additional features, such as fault tolerance or location-independent naming, which are not already provided by the server platform on which it is implemented. In order to support such useful features, it is necessary to layer additional protocols on top of IBP. These higher-level protocols can be implemented at the application level, or in more general high-level protocols, analogous to the way Transmission Control Protocol (TCP) is layered on top of IP. While various application efforts are underway that make direct use of IBP as a primitive service, other development efforts are exploring the definition and implementation of higher-level storage protocols that make indirect use of it possible.

One example of an application that makes native use of IBP is IBP-Mail, and it illustrates the power of this approach. IBP-Mail uses IBP to transmit and deliver mail attachments that require storage resources beyond the capacity of standard mail servers. While the use of e-mail attachments to move digital objects has become ubiquitous over the past few years, the impact of this technology has been curtailed by the fact that the size of attachments that can be sent is typically limited to 10s of megabytes of data. For example, if you were to attempt to send someone a new digital video, which might be several gigabytes in size, the mail servers in between would be certain to reject such a huge enclosure. Moreover, even if such a file could be sent, there is no guarantee that the recipient would have sufficient resources to download it.

When using IBP-Mail, the file to be sent is never stored on a mail server, so typical restrictions on the size of enclosures do not apply as illustrated in figure 1. The file is instead stored in the network at an IBP storage depot and only a pointer to this file is mailed. Upon receiving the mail, the recipient can either download the file from the network or direct it to a nearby streaming media server, accessing the result with the appropriate media player. Moving the file between IBP depots allows the file to be transferred to a location where high-bandwidth service can be provided, thus eliminating the need for an actual download.

Research with IBP includes experimentation with routing and forwarding IBP-Mail attachments, and with the use of processing power at the IBP depots for such things as compression/decompres-

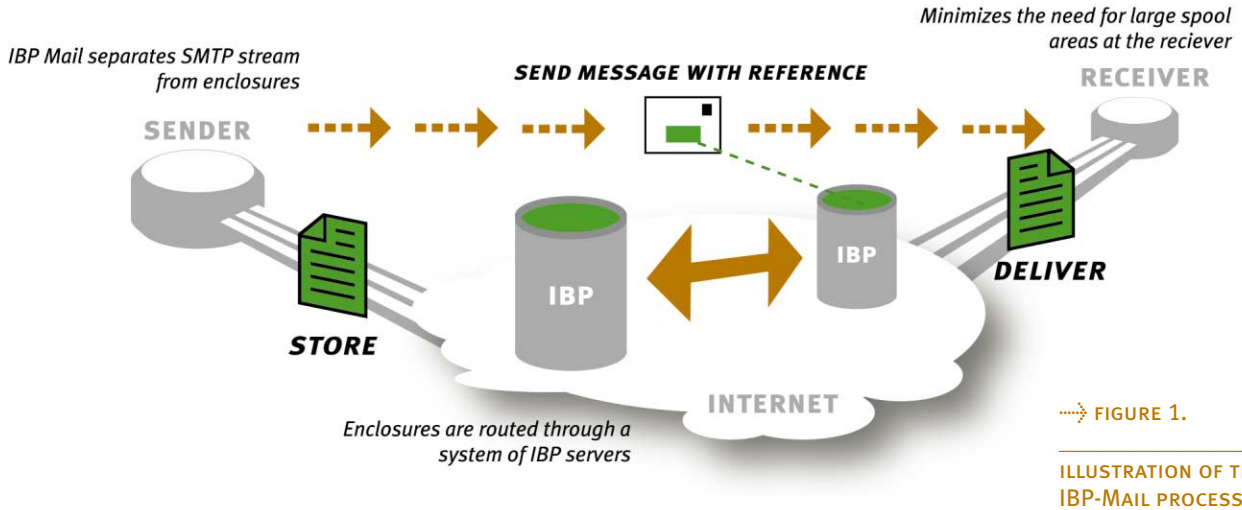



FIGURE 1.

ILLUSTRATION OF THE IBP-MAIL PROCESS

sion of attachments. Since the need to move ever larger digital objects around within the science and engineering community is near universal, we expect that IBP-Mail's power to manage extremely large attachments will make it a popular tool.

The IBP project is closely related to two other ICL projects – Internet2 Distributed Storage Infrastructure (I2-DSI) and Scalable Intracampus Research Grid (SInRG) – both of which involve close collaborations with faculty in the UT Computer Science Department, especially Rich Wolski and Jim Plank. We have already implemented a proof of concept prototype of IBP, which we are deploying on the I2-DSI. IBP is also now being integrated into distributed computing environments (e.g., NetSolve and Ninf) in order to support the management of distributed state for such things as caching, process migration, and fault tolerance.

To promulgate the use of IBP for *logistical networking* by the research community, we are using the I2-DSI to deploy IBP nodes in an experimental, wide-area test bed we call the Logistical Backbone (L-Bone). The L-Bone consists of a set of IBP servers and some basic associated resources and protocols, such as directory service and network proximity measurement, which will offer higher level services to the research community.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/ibp/

**E-mail**
ibp@cs.utk.edu

**Publications**
Plank, J., Beck, M., Elwasif, W., Moore, T., Swany, M., Wolski, R. "The Internet Backplane Protocol: Storage in the Network," *Proceedings of Network Storage Symposium*, October, 1999. http://dsi.internet2.edu/netstore99/docs/papers/plank.pdf

**Agency Funding**
Department of Energy (DOE)
National Science Foundation (NSF)

**Related URLs**
I2-DSI - http://icl.cs.utk.edu/projects/I2DSI/
NetSolve - http://icl.cs.utk.edu/projects/netsolve/
Ninf - http://ninf.etl.go.jp/
SInRG - http://icl.cs.utk.edu/projects/SInRG/

# MPI_Connect

**ICL TEAM** (ALPHABETICAL)
Jack Dongarra
Graham Fagg
Kevin London

**COLLABORATORS**
Oak Ridge National Laboratory
University of Stuttgart, Germany

MPI_Connect is a software package that allows heterogeneous parallel applications running on different Massively Parallel Processors (MPPs) to interoperate and share data thus creating Meta-applications.

The software package is designed to support applications running under vendor message passing interface (MPI) implementations and allow interconnection between these implementations by using the same MPI send and receive calls as developers already use within their own applications. This precludes users from learning a new method for interoperating, as the syntax for sending data between applications is identical to what they already use. The support of vendor MPI implementations means that internal communications occur via the optimized vendor versions and thus incur no performance degradation as opposed to using only TCP/IP, the only other option previously available.

The need for MPI_Connect came from the development of vendor versions of MPI for MPPs. These special machines did not support inter-machine interaction, and the MPI standard itself did not allow for dynamic process management. In short, once the parallel job had been started, the number of nodes (and hence processes) was fixed or static. In MPI terms, all processes formed a communicator "MPI_COMM_WORLD," and all communication could only be within this communicator. The naming of all processes is based on this communicator, and known as a rank, from 0 to N-1 where N is the number of processes started. This meant that these processes could not talk to any other processes because MPI did not provide a means for addressing these processes in terms of a rank.

This had three drawbacks:

1. If a single machine was not big enough for a computation, then multiple machines could not be connected together to form a bigger system.
2. If additional computational nodes were required after the application was started, they could not be added to an already executing application.
3. If a node that a process was executing on failed, then the whole application would be aborted since MPI had no means of handling a failure or restarting an application.

The first problem could be handled by a public domain version of MPI, called MPICH, from Argonne National Laboratory, by utilizing the CH-P4 device that uses TCP/IP. Another public implementation known as LAM-MPI also supported multi-machine execution. Unfortunately, this would slow down internal machine communication by a factor of 2 or more compared to the native MPI performance.

The second problem was addressed in version two of the MPI standard, known as MPI-2, by the addition of dynamic process functions. Unfortunately, few vendors supported these dynamic process management features, and none supported heterogeneous dynamic process management. To support fault tolerance applications would mean changing the MPI semantics. Without this, the only option is checkpoint, rollback, and restart. Several systems support MPI applications over the Condor checkpoint library. FT-MPI supports native fault tolerant applications without check-pointing and is the default MPI implementation for the DOE/ICL HARNESS Project. The project closest to MPI_Connect in functionality is the PArallel eXtentions (PACX) project from the University of Stuttgart Computer Center. Although the PACX package is a complete MPI implementation in its own right, it currently only supports the Cray T3D/E and Intel Paragon XPS architectures.

MPI_Connect is unique in that it is

1. Light weight: it is not a complete MPI implementation, but a run-time
2. Efficient: it uses Vendor MPI implementations for all internal communications
3. Dynamic: applications can connect and disconnect multiple times to multiple applications
4. Configurable: users can control the number of nodes that perform intercommunication

MPI_Connect is used by adding as few as three extra function calls to existing applications. The extra calls are used to name processes in order for external applications to locate and then communicate with these processes. In simple terms, the extra calls are used to

1. Name applications (or application parts)
2. Find other named applications and recreate a

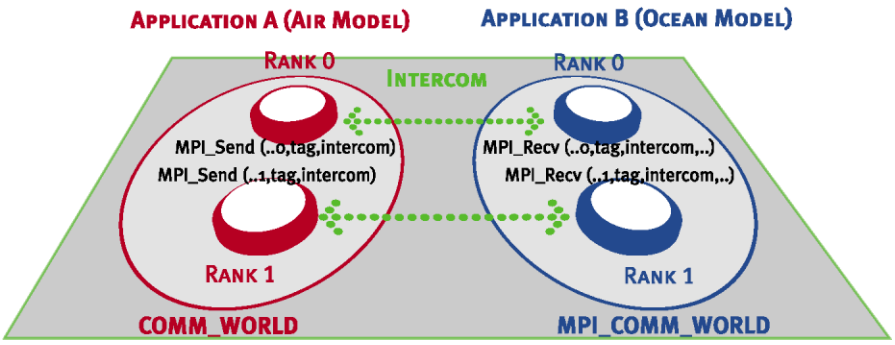



FIGURE 1.

ILLUSTRATION OF THE COUPLING OF TWO HYPOTHETICAL APPLICATIONS

connection to them

3. Communicate with them using normal MPI point-to-point communications
4. Remove the application names after use
5. Repeat the above as many times as needed

MPI_Connect usage is best illustrated with a simple example such as the coupling of hypothetical applications A and B. The coupling could be a complex combination of Wave Height and Atmospheric models, for example.

As illustrated in figure 1, we connect application A to application B. The first step is for both applications to register their names. Then each application looks up the other application (both must do this to enable bi-directional communication). Once the applications have looked each other up, they are given inter-communicators that they can use for point-to-point communications. When the process is complete, the applications remove their names, free the inter-communicators, and then exit.

MPI_Connect uses the MPI profiling interface to intercept MPI calls and then decides whether they are internal or external to a MPP system. In the case of external communication, the data is packaged correctly using binary, byte swapping, or XDR and then sent via the SNIPE Lite communications library.

MPI_Connect uses the MPI profiling interface to intercept all MPI calls. Once a call has been intercepted, MPI_Connect checks the communicator and data structures used and decides whether the thread of control should be passed to the PMPI native library or an alternative communications library (usually SNIPE_Lite). Generally, if the MPI call involves a communication within the local MPI_COMM_WORLD, then the thread of control is passed to the native library. Otherwise, it is an external communication. In the case of an external communication, some type of data manipulation may have to be performed either before or after the external communications library is invoked. In addition, the translation of external return codes to matching return codes defined by the local MPI implementation must also be performed.

Currently, the main users of MPI_Connect have been the DoD Major Shared Resource Centers (MSRCs) as part of the Programming Environment and Training (PET), especially the Climate Weather Oceanography (CWO) Computational Technology Area (CTA). Such use resulted in an Engineering Research and Development Center (ERDC-formally CEWES) team winning an award at SuperComputing '98 for using MPI_Connect to couple a CG Wave application between multiple SGI Origin2000s at multiples sites.

Another noteworthy user has been the DOE Accelerated Strategic Computing Initiative (ASCI) program via their benchmarking of the Blue-Pacific IBM SP system at Lawrence Livermore National Laboratory (LLNL). This involved coupling the three separate SP systems using MPI_Connect and running the LINPACK Benchmark from UT. The 5800 CPU systems sustained 2.144 Teraflops of performance.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/mpiconnect/

**E-mail**
mpiconnect@cs.utk.edu

**Publications**
Fagg, G., London, K. "MPI Inter-connection and Control," DoD ERDC PET *Technical Report*, 1998.

**Awards**
SC98 HPC Challenge Award for Most Effective Engineering Methodology

**Agency Funding**
Department of Energy (DOE)

**Related URLs**
DoD MSRCs - http://www.hpcmo.hpc.mil/Htdocs/MSRC/
DOE ASCI Blue-Pacific - http://www.llnl.gov/asci/platforms/bluepac/
MPI - http://www.netlib.org/mpi/
PACX - http://www.hlrs.de/organization/pds/projects/pacx-mpi/
PVM - http://www.epm.ornl.gov/pvm/pvm_home.html

# NetSolve

**ICL TEAM (ALPHABETICAL)**
Sudesh Agrawal (G)
Dorian Arnold
Susan Blackford
Jack Dongarra
Yan Huang (G)
Michelle Miller
Sathish Vadhiyar (G)
(G) = GRADUATE STUDENT

**COLLABORATORS**
Univerisity of California, San Diego

NetSolve is a project that investigates the usage of distributed computational resources connected by computer networks to solve complex scientific problems efficiently. It is a remote procedure call (RPC)-based client/agent/server system that allows users to discover, access, and utilize remotely housed software modules, as well as the computational hardware needed to run these modules. The resources to be leveraged can be distributed by geographic location and/or ownership, and heterogeneous operating environments are supported.

The motivation for NetSolve is to create a grid-based software computing environment used routinely by a large user base to enhance scientific computing capabilities. Fundamental characteristics include

- Ease-of-use for both the user and administrator
- Efficient utilization of resources
- Ease-of-integration of new software modules
- High levels of quality assurance (in the accuracy and performance of both the NetSolve system and the underlying software services)

The NetSolve framework is based on the premise that distributed computations involve resources, processes, data, and users, and that secure yet flexible mechanisms for cooperation and communication between these entities is the key to metacomputing infrastructures. Although other research groups and organizations are investigating Distributed and Grid computing concepts, NetSolve's niche is providing access to complex collections of high-performance software that run on clusters of commodity components or supercomputers. Such access reduces the effort scientists normally exert to use these software resources. At the same time, the system leverages aggregate hardware resources to vastly improve processing times.

To a NetSolve client user, the system is interfaced by a convenient and intuitive application programming interface (API). The API has been implemented in a variety of languages and environments, including C, FORTRAN, Matlab, and Mathematica. These interfaces allow client users to invoke the NetSolve system with requests for software services (the user also defines the input and output parameters for the required service). Once invoked, the NetSolve client automatically contacts the NetSolve information service and resource scheduler, the NetSolve agent. The agent uses both static and dynamic information collected from NetSolve computational servers to determine which server can service the client user's request most efficiently. This information is transmitted to the client that, once again, automatically negotiates for this service to be carried out with the specified computational server. Input data is then transferred from the client host to the server host, which executes the requested service and transfers output data back to the client host. At this point in the process, control is returned to the client user's program with the results from the service now available.

From a NetSolve administrator's perspective, the system has two key components: an agent and a server. The agent represents the gateway to the NetSolve system. It maintains a database of NetSolve servers along with their capabilities (hardware performance and allocated software) and dynamic usage statistics. It then uses this information to allocate server resources for client requests, as mentioned above. The agent, via its resource allocation mechanisms, attempts to find the server that will service the request the quickest, balance the load amongst its servers, and keep track of failed servers. Requests are directed away from failed servers. The agent also adds fault-tolerant heuristics that attempt to use every likely server until it finds one that successfully services the request.

The NetSolve server, the computational backbone of the system, is a daemon process that awaits client requests. The server can run on single workstations, workstation clusters, symmetric multiprocessors or machines with massively parallel processors (MPPs). One key component of the server is a source code generator that parses a NetSolve problem description file (PDF). This PDF contains information that allows the system to create new modules and incorporate new software functionalities. Basically, the PDF defines a wrapper that the server uses to call the function being incorporated.

There are many advantages to using NetSolve. NetSolve can provide access to otherwise unavailable software and, in cases where the software is

readily available, it can make the power of supercomputers accessible from low-end machines such as laptop computers. NetSolve is designed to suit the needs of the domain scientist who is not necessarily a mathematics or computer science expert. For such a user, the system provides convenient, intuitive, and uniform interfaces to a wide variety of numerical functions. NetSolve is also designed to increase the accessibility of larger software systems like simulators and modeling software. For users of these systems, NetSolve can make software available to platforms that it has not been ported to or improve execution time by running the software on higher capacity computational resources. NetSolve can also be used to extend the capabilities of problem solving environments (PSE), such as Matlab, by increasing the number and types of implemented algorithms available. The system also provides these environments with the ability to distribute NetSolve's computational tasks among multiple processors – a feat, for example, that is not possible with Matlab alone.

NetSolve has been employed by users in a variety of scientific domains ranging from image processing to nuclear engineering, microbiology, and sub-surface fluid modeling. Since the software system is freely available for download and use from the project Web site, there are many undocumented cases of NetSolve use. Our experience, in addition to requests for technical support, tells us that the majority of these cases are of users who have embedded calls to NetSolve within their computational science applications to access lower-level services (i.e., numerical linear algebra routines like linear system solvers, eigensolvers, differential equations, etc.). The NetSolve project has also been part of larger collaborations with research groups from universities, government laboratories, and private research organizations. The aims of these collaborations have primarily been to integrate NetSolve with large scientific applications and simulations to improve the performance of these applications through resource aggregation, as well as to make such software more readily accessible to larger, interested communities.

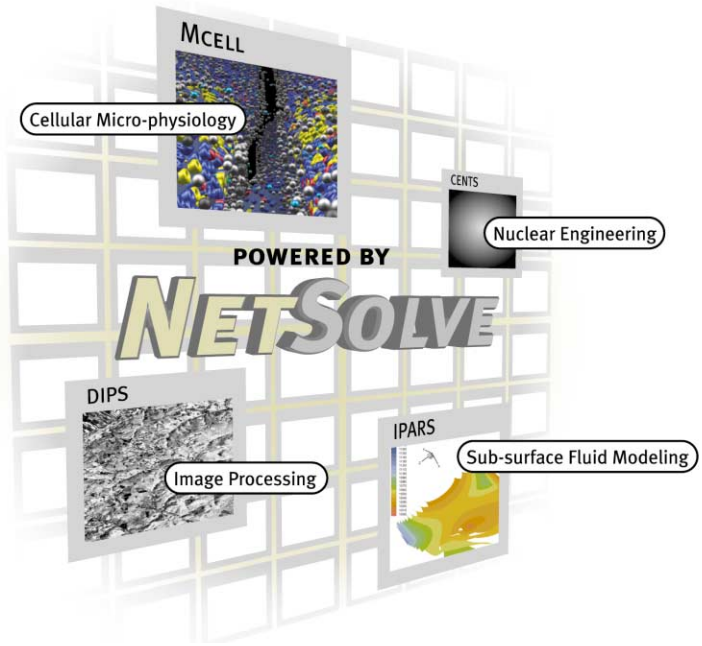



FIGURE 1.

ILLUSTRATION DEPICTING SOME OF THE APPLICATIONS AND RESEARCH FIELDS THAT UTILIZE NETSOLVE

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/netsolve/

**E-mail**
netsolve@cs.utk.edu

**Publications**
See http://www.cs.utk.edu/netsolve/documentation.html

**Awards**
R&D 100 winner, 1999

**Agency Funding**
Department of Defense (DoD)
Department of Energy (DOE)
National Science Foundation (NSF)

**Related URLs**
IPARS - http://www.ticam.utexas.edu/CSM/ACTI/ipars.html
MCell - http://www.mcell.cnl.salk.edu/
ORNL - http://www.ornl.gov/
Salk Institute for Biological Studies- http://www.salk.edu/

# SInRG

## SCALABLE INTRACAMPUS RESEARCH GRID

**ICL TEAM (ALPHABETICAL)**

Micah Beck
Jack Dongarra
Brett Ellis
Terry Moore

**COLLABORATORS**

UNIVERSITY OF TENNESSEE

Don Bouldin, Electrical & Computer Engineering
Peter Cummings, Chemical Engineering
Jens Gregor, Computer Science
Louis Gross, Ecology & Evolutionary Biology
Michael Langston, Computer Science
James Plank, Computer Science
Gary Smith, Medical Arts
Michael Thomason, Computer Science
Robert Ward, Computer Science
Rich Wolski, Computer Science

The Innovative Computing Laboratory is leading a large collaboration of faculty members from Computer Science and other UT departments in the creation of an experimental technology grid on the Knoxville campus. The purpose of this infrastructure, which is called the Scalable Intracampus Research Grid (SInRG), is to support leading-edge research on technologies and applications for a new approach to high performance distributed computing and information systems.

The vast majority of SInRG's funding will go to purchase special Grid Service Clusters (GSCs), which are hardware ensembles specifically designed and configured to fit SInRG's multifaceted research agenda. As shown in figure 1, each GSC will consist of a compute engine (e.g., a large commodity cluster), a mass storage device, and a fast data switch integrating them all and connecting them to the campus' high performance network. Essentially, each GSC is a next generation workgroup cluster in an advanced local area network. Each of them will be assigned to one of the collaborating teams and will be customized to meet their special needs. But they are also designed from the ground up to be nodes on the Grid, so that computational power can be moved around as needed and resources can be flexibly shared by the whole SInRG community.

As SInRG is deployed over the next few years, it will mirror within the boundaries of the Knoxville campus both the underlying technologies and the interdisciplinary research collaborations that are characteristic of the National Technology Grid currently being developed by the US research community. A “computational power grid” like SInRG uses special system software, sometimes known as network middleware, to integrate high performance networks, computers, and storage systems into a unified system that can provide advanced computing and information services (e.g., data staging, remote instrument control, and resource aggregation) in a pervasive and dependable way for an entire community.

The National Technology Grid is now growing out of the convergent efforts of the National Science Foundation’s (NSF) Partnerships for Advanced Computational Infrastructure (PACI) and several other government agencies, including NASA, DoD, and DOE. While the SInRG infrastructure will in time become a prominent node on this national Grid, it’s primary purpose is to provide a technological and organizational microcosm in which key research challenges of grid-based computing can be addressed by leveraging the advantages of local communication and local control. SInRG is supported by a $2 million, five year grant from the Research Infrastructure Program of the Computer and Information Science and Engineering directorate of the NSF.

As illustrated in figure 1, the project involves a large team of computer scientists and research partners from other disciplines that will build and use SInRG. The project team is made up of two basic groups. One is focused on research for SInRG’s middleware, and the other is engaged in interdisciplinary research leading to applications that will leverage SInRG’s power.

The UT CS researchers who make up the middleware group (Jack Dongarra, Jim Plank, Rich Wolski, and Micah Beck) bring complementary research interests and component software to the task at hand, including software for remote scientific computing (NetSolve), distributed scheduling (AppLeS), resource monitoring and performance prediction (Network Weather Service), and flexible management of distributed storage (IBP). To create SInRG’s system software, this group is not only building on these different components, they are also leveraging the work of the PACIs and other parts of the national grid community. This work is therefore part of a much larger story about the evolution of modern information power grids.

SInRG’s research applications reflect another aspect of the national effort, via the fact that it is based on interdisciplinary collaboration between computer scientists and researchers from other domains with extremely challenging computational problems to solve. The grid community recognizes that, to make rapid progress, the requirements of advanced applications must drive the development of grid technology. The initial set of SInRG applications from UT will build on long running partnerships between a group of computer science co-PIs and

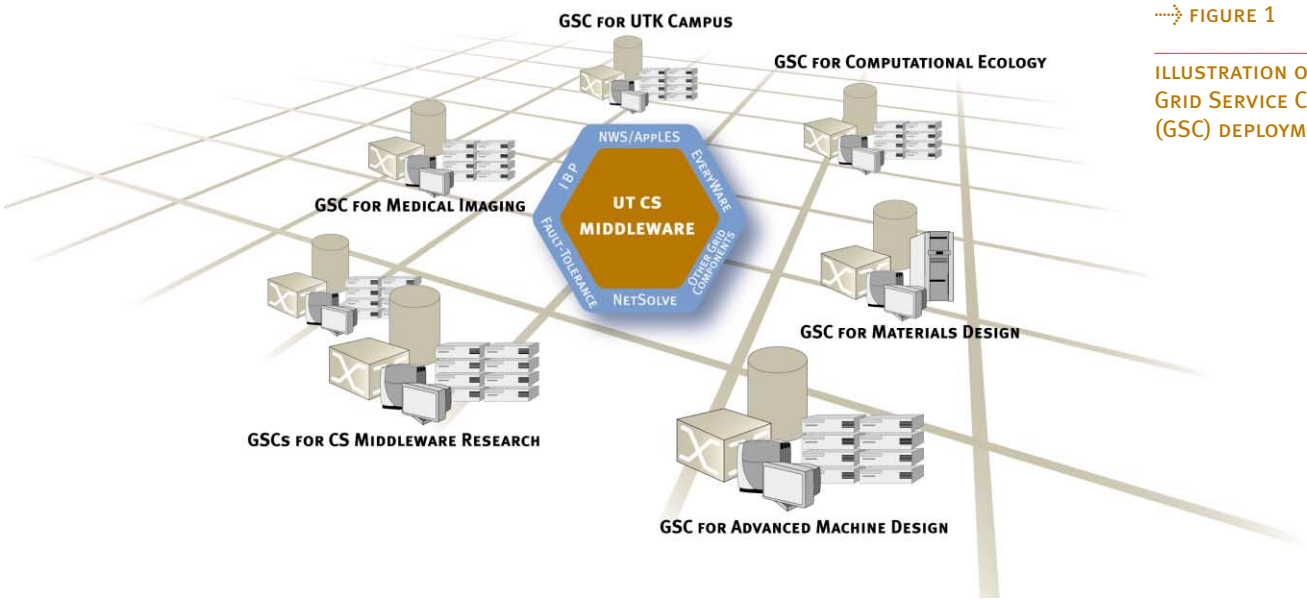



FIGURE 1

ILLUSTRATION OF THE GRID SERVICE CLUSTER (GSC) DEPLOYMENT

leading researchers from other UT departments, including chemical engineering, medical imaging, electrical engineering and computational ecology. The fact that SInRG has such well-established research collaborations to build on and that all the collaborators are on the same campus is a major advantage for the project.

Like the national Grid, SInRG will be a geographically distributed system. By the end of the five years, there will be at least seven GSCs spread among six different locations around the campus, including one across the Tennessee River at the UT Medical Center. Each will be managed with some degree of autonomy by these groups, with oversight and coordination from the CS co-PIs who collaborate with them. The campus network will provide the underlying fabric that makes it possible to use all this distributed hardware as a single collective resource, a unified computational power grid.

Motivated to find a new approach to supporting high performance computing for the campus research community, UTK's Division of Information Infrastructure is also participating in SInRG and will have a GSC that is partially funded by the project. Since central university computing facilities are often increasingly pressured to fund large-scale supercomputers, SInRG's concept represents a new model of resource sharing that permits large-scale computing within more modest institutional budgets. Using this model, project leaders anticipate that over time the success of SInRG will encourage other UT researchers (including researchers on other UT campuses) to join SInRG, adding their own GSCs and other resources to the Grid.

Finally, though the SInRG project is primarily supported by the National Science Foundation (NSF), it also represents an excellent example of mutually beneficial collaboration between academia and private industry. Eager to support leading edge research on grid-based computing and to test out their technology in the environment SInRG provides, Microsoft and Sun Microsystems have made significant contributions of funding and/or technology to the project. Project leaders also expect this kind of collaboration with industry to continue throughout the life of the project.

## RESOURCES

**Web Site**
http://www.cs.utk.edu/sinrg/

**E-mail**
sinrg@cs.utk.edu

**Publications**
See http://www.cs.utk.edu/sinrg/docs/

**Agency Funding**
National Science Foundation (NSF)

**Industrial Support**
Dell Computer Corporation
Microsoft Research
Sun Microsystems

# TORC

## TENNESSEE OAK RIDGE CLUSTER

**ICL TEAM (ALPHABETICAL)**

Jack Dongarra

Brett Ellis

Paul Peltz

**COLLABORATORS**

Grid Application Development System (GrADS)

Microsoft Research

Myricom

Oak Ridge National Laboratory

UT Computer Science Department

The Tennessee Oak Ridge Cluster (TORC) is a two-part cluster environment with high-speed, low latency interconnects. It is primarily comprised of commodity hardware and software, and its purpose is to provide a local, easily accessible computer resource for parallel computing, grid-based meta-computing, different network technologies, and research/development of software. The two-part cluster consists of a production portion and an experimental portion. TORC is a cluster that resides at both the University of Tennessee (UT) and Oak Ridge National Lab (ORNL), but only the UT portion is described below.

Our close working relationship with companies like Giganet, Microsoft, Intel, Packet Engines, and Foundry networks, make the cluster possible. The commodity part of the name is very important in its design. We wanted proof that with small amounts of funding, when compared to the price of a modern supercomputer, we could attain decent performance and stability measured by cost per flop.

The production cluster consists of homogeneous, highly stable machines for long runs as well as modeling and is accessible by ICL staff at all times. The production cluster consists of eight machines and is designed solely for computational purposes. It has the following configuration:

- Dual PIII 550 MHz Processors
- 512MB memory
- 9GB UltraWide SCSI
- RedHat Linux 6.2

The production cluster employs three interconnect types for a rich variety of networking experiences: Myrinet, Giganet, and 100 Mbit Ethernet. There is also a head node called Torc0 that maintains a consistent state of software across the cluster. Common research software installed include MPICH, Globus, ScaLAPACK, Legion, ATLAS, NWS, Matlab, NetSolve, PVM, BLACS, LAPACK, TotalView, and various compilers. The Repository in a Box (RIB) software toolkit is used to maintain an easily accessible software catalog for our users, and the production cluster is available to all members of the UT Computer Science (CS) Department as well as researchers from other universities and institutions.

The experimental portion of the cluster exists as a configurable group of machines that can be booted into multiple operating systems. This cluster includes Pentium II's and III's in both single and SMP processor arrangements. It consists of 12 additional machines that can be (by request) subdivided in any way to run any OS that runs on the I86 architecture. Currently, the default OS on each machine is Linux. Table 1 at right shows the various configurations of the machines that comprise the experimental cluster. The interconnects available consist of all of the ones included in the production clusters as well as several other kinds of Gigabit ether technology.

The experimental cluster is used to perform many of the same tasks as the production cluster. It also serves as a roll out area for new OSs. On the experimental cluster, we test both Windows and Unix operating systems for usability, stability, and development. It is very important that software produced at ICL, and within the CS department as a whole, has the widest possible testbed to ensure functionality. The experimental cluster provides this on a small scale. If unreserved, the experimental cluster is available to the same users of the production cluster. There are times, depending on need, that much of the experimental cluster is reserved for private use.

An additional benefit of the experimental cluster is the ability to test new machines. The architectures have included Compaq Alpha, Alpha Clones, and Sun Sparcs. Such testing has been extremely valuable from the viewpoint of both the software developer and systems administrator.

TORC is currently used in a variety of projects within ICL and the CS department. Ongoing research projects currently utilizing TORC include: ATLAS, NetSolve, ScaLAPACK, and RIB. We have also used it to test LSF, Globus, Legion, and PBS for several funding institutions. The most obvious use for the cluster has been to test clustering technologies. It has helped us, and others, to understand the intricacies and problems inherent in putting together a cluster of machines that need to be very flexible and stable for users.

FIGURE 1.

PHOTO OF THE TORC PRODUCTION CLUSTER

| # | PROCESSOR | RAM | STORAGE | OPERATING SYSTEM |
|---|---|---|---|---|
| 3 | PENTIUM III 600MHZ | 256 MB | 9 GB EIDE DRIVE | WINDOWSNT/REDHAT LINUX 6.1 |
| 2 | PENTIUM II 450MHZ | 256 MB | 8 GB SCSI DRIVE | WINDOWSNT/REDHAT LINUX 6.1 |
| 5 | DUAL PENTIUM II 200 MHZ | 256 MB | 6 GB EIDE DRIVE | WINDOWSNT/REDHAT LINUX 6.1 |
| 1 | DUAL PENTIUM III 550 MHZ | 256 MB | 9 GB ULTRAWIDE SCSI | WINDOWSNT/REDHAT LINUX 6.1 |
| 1 | PC164 ALPHA CLONE 533 MHZ EV56 | 256 MB | 2- 2 GB EIDE DISK DRIVES | REDHAT LINUX 6.2 |

TABLE 1.

THE CONFIGURATION OF THE EXPERIMENTAL TORC CLUSTER

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/torc/

**E-mail**
torc@cs.utk.edu

**Agency Funding**
National Science Foundation (NSF)

**Related URLs**
Beowulf - http://www.beowulf.org/
Berkley NOW Project - http://now.cs.berkeley.edu/
High Performance Virtual Machines Project - http://www-csag.ucsd.edu/projects/clusters.html
High Speed Networks and Parallel Applications Project - http://lhpca.univ-lyon1.fr/
Jazznet - http://math.nist.gov/jazznet/

# NA-Net

# NA-Digest

**ICL TEAM (ALPHABETICAL)**

Jack Dongarra

Jeremy Millar

Keith Moore

**COLLABORATORS**

Lucent Technologies

Stanford University

The Mathworks

The NA-Net is a system developed to serve the numerical analyst community and other researchers of similar interest. The NA-Net provides two independent databases and a weekly digest to its members. One of these independent databases contains e-mail information and it knows the electronic mail address of each of its members and is capable of forwarding mail to them. In addition, this database serves as the distribution list for the NA-Digest described below. Also contained in NA-Net is a white pages database. The white pages database is basically a directory service that provides a way to exchange personal information among its members. Contained in the white pages database are phone numbers, postal mailing addresses, research interests, affiliations, etc. In addition to the two databases, the NA-Net also provides its members with the NA-Digest, a weekly collection of articles on topics related to numerical analysis and those who practice it. Figure 1 and table 1 provide the latest demographic information of NA-Net memberships.

| DOMAIN NAME | COUNTRY CODE | MEMBERS |
|---|---|---|
| ae | United Arab Emirates | 1 |
| ar | Argentina | 26 |
| at | Austria | 44 |
| au | Australia | 140 |
| be | Belgium | 94 |
| bg | Bulgaria | 20 |
| bo | Bolivia | 2 |
| br | Brazil | 77 |
| bw | Botswana | 1 |
| by | Belarus | 7 |
| ca | Canada | 212 |
| ch | Switzerland | 66 |
| cl | Chile | 13 |
| cn | China | 53 |
| co | Colombia | 5 |
| com | US Commercial | 1077 |
| cr | Costa Rica | 3 |
| cu | Cuba | 5 |
| cy | Cyprus | 3 |
| cz | Czech Republic | 32 |
| de | Germany | 523 |
| dk | Denmark | 46 |
| dz | Algeria | 1 |
| ec | Ecuador | 1 |
| edu | US Educational | 1800 |
| ee | Estonia | 2 |
| eg | Egypt | 1 |
| es | Spain | 156 |
| et | Ethiopia | 1 |

| DOMAIN NAME | COUNTRY CODE | MEMBERS |
|---|---|---|
| fi | Finland | 33 |
| fr | France | 376 |
| gb | Great Britain | 1 |
| ge | Georgia | 1 |
| gov | US Government | 292 |
| gr | Greece | 70 |
| hk | Hong Kong | 32 |
| hr | Croatia | 13 |
| hu | Hungary | 21 |
| id | Indonesia | 11 |
| ie | Ireland | 17 |
| il | Israel | 71 |
| in | India | 100 |
| int | International | 1 |
| ir | Iran | 30 |
| is | Iceland | 3 |
| it | Italy | 223 |
| jo | Jordan | 1 |
| jp | Japan | 173 |
| kr | South Korea | 69 |
| kw | Kuwait | 2 |
| kz | Kazakhstan | 1 |
| lb | Lebanon | 39 |
| lt | Lithuania | 4 |
| lv | Latvia | 1 |
| ma | Morocco | 6 |
| mil | US Military | 56 |
| mk | Macedonia | 1 |
| mo | Macau | 2 |

| DOMAIN NAME | COUNTRY CODE | MEMBERS |
|---|---|---|
| mx | Mexico | 22 |
| my | Malaysia | 8 |
| na | Namibia | 1 |
| net | Network | 180 |
| nl | Netherlands | 137 |
| no | Norway | 69 |
| nz | New Zealand | 42 |
| om | Oman | 1 |
| org | Non-Profit Organization | 84 |
| pa | Panama | 2 |
| pe | Peru | 4 |
| ph | Philippines | 4 |
| pk | Pakistan | 4 |
| pl | Poland | 51 |
| pt | Portugal | 46 |
| py | Paraguay | 1 |
| ro | Romania | 39 |
| ru | Russia | 107 |
| sa | Saudi Arabia | 13 |
| se | Sweden | 133 |
| sg | Singapore | 16 |
| si | Slovenia | 10 |
| sk | Slovak Republic | 9 |
| sn | Senegal | 1 |
| su | USSR (former) | 28 |
| sz | Swaziland | 1 |
| th | Thailand | 8 |
| tr | Turkey | 37 |
| tw | Taiwan | 43 |
| ua | Ukraine | 13 |

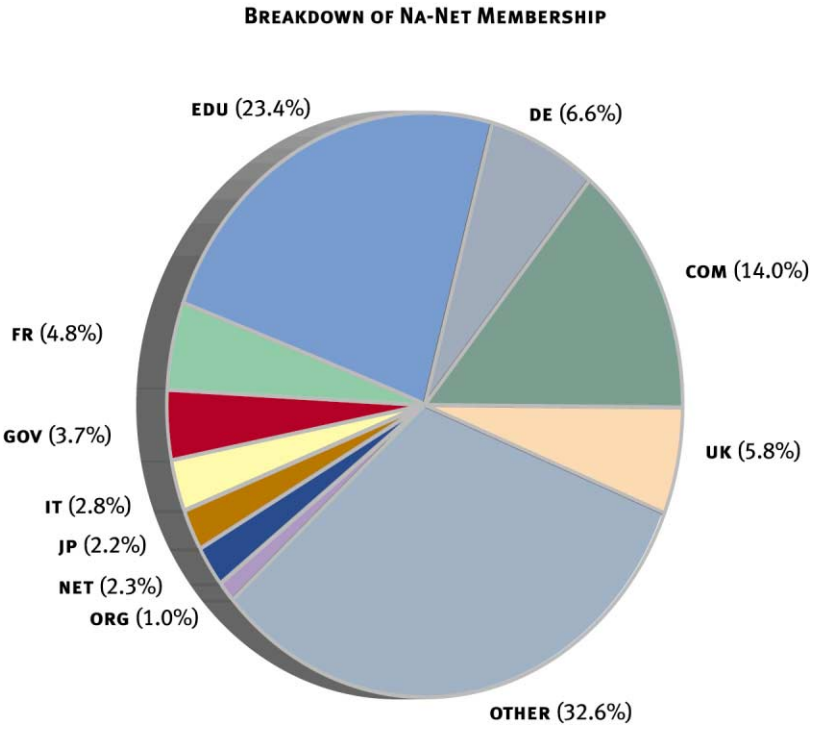



FIGURE 1.

CHART DEPICTING THE DEMOGRAPHIC INFORMATION OF NA-NET MEMBERSHIPS

| DOMAIN NAME | COUNTRY CODE | MEMBERS |
|---|---|---|
| uk | United Kingdom | 449 |
| us | United States | 6 |
| uy | Uruguay | 2 |
| ve | Venezuela | 11 |
| vn | Vietnam | 2 |
| yu | Yugoslavia | 13 |
| za | South Africa | 31 |
| | TOTAL | **7690** |

TABLE 1.

DETAILED DEMOGRAPHIC INFORMATION OF NA-NET MEMBERSHIPS

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/nadigest/

**E-mail**
nadigest@cs.utk.edu

**Related URLs**
NA-Net - http://www.netlib.org/na-net/na_home.html

# Netlib

**ICL TEAM (ALPHABETICAL)**
Jack Dongarra
Jeremy Millar

**COLLABORATORS**
Lucent Technologies

Netlib is an online repository of freely available software, documents, and databases. Created in 1985, it is currently maintained by multiple institutions. Originally consisting of only two servers, the repository has been replicated in Britain, Greece, Russia, Japan, and Korea and now consists of four main servers. These servers are located here at ICL, Bell Labs, the UK, and Norway. Each server performs as a master for particular subsets of the entire collection that are then replicated and automatically synchronized to ensure efficient service is provided throughout the global community.

Netlib is unique compared to other software repositories in that it is moderated by an editorial board. Moderation ensures that the quality of software remains high; however, no support or guarantees are offered by the Netlib maintainers.

While most of the software in Netlib is mathematical in nature, the repository also contains software for use by the entire scientific computing community such as networking and visualization tools. Software on Netlib may be retrieved via FTP, HTTP, and XNetlib or by sending an e-mail request to the Netlib maintainers. Netlib has the ability to send individual files or entire libraries since it performs what is termed "dependency checking." This feature allows for the delivery of all files a software package may depend on, which eliminates the need to make multiple requests thus reducing time and effort on the part of the user. Additionally, submissions may be made to the repository, which further enhances its benefits to the computing community.

The mathematical software stored in Netlib is classified using the Guide to Available Mathematical Software (GAMS) hierarchy. Currently, the repository has 14,823 files cataloged. Figure 1 illustrates the file requests from Netlib since its inception.

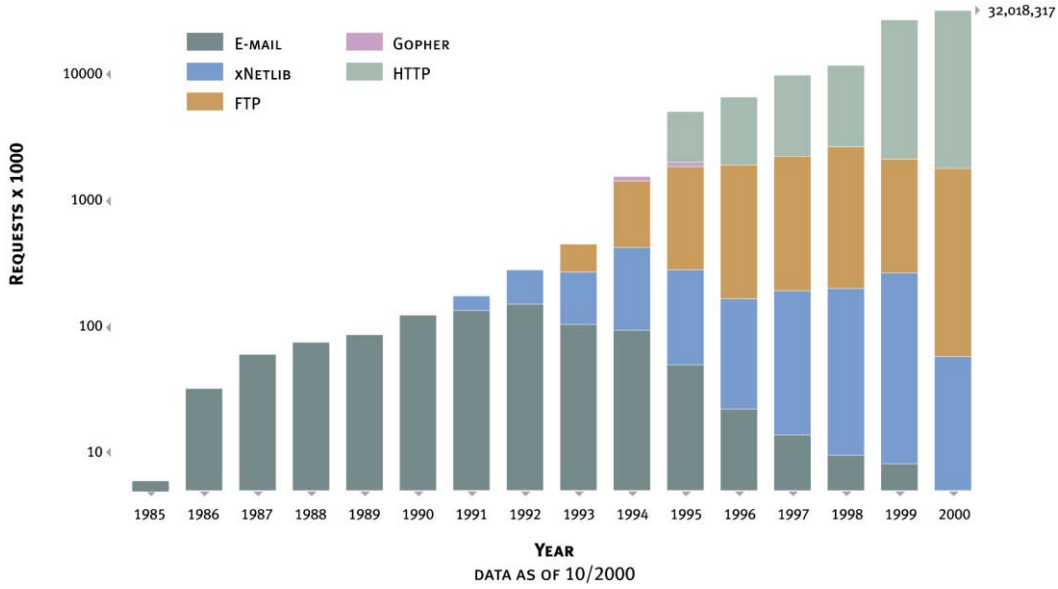



FIGURE 1.

LOGARITHMIC CHART DEPICTING REQUESTS MADE TO NETLIB REPOSITORIES AT UT AND ORNL

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/netlib/

**E-mail**
netlib@cs.utk.edu

**Publications**
See http://www.netlib.org/srwn/index.html

**Agency Funding**
National Science Foundation (NSF)

**Related URLs**
Matrix Market - http://math.nist.gov/MatrixMarket/
National High-Performance Software Exchange (NHSE) - http://www.nhse.org/

**FAQ**
http://www.netlib.org/misc/faq.html

**Mirror Sites**
Bell Labs (Lucent Technologies) - http://netlib.bell-labs.com/netlib/master/readme.html
Norway (Bergen) - http://www.netlib.no/netlib/master/readme.html
United Kingdom (Kent) - http://www.mirror.ac.uk/sites/netlib.bell-labs.com/netlib/master/readme.html

# NHSE

**NATIONAL HIGH-PERFORMANCE SOFTWARE EXCHANGE**

**ICL TEAM (ALPHABETICAL)**

Jack Dongarra
Kevin London
Jeremy Millar
Shirley Moore
Scott Wells

**COLLABORATORS**

Argonne National Laboratory
California Institute of Technology
Rice University
Syracuse University

The National High-performance Software Exchange (NHSE) project started in 1994 as an effort to promote sharing and reuse of parallel computing software and tools among and between the federal high performance computing (HPC) agencies. Using the highly successful Netlib mathematical software repository as a model, the NHSE sought to establish discipline-oriented software repositories that could be contributed to and maintained by experts in their respective fields. Because of the need to share software between organizations and across disciplines, repository interoperability was an important goal. Similar to Netlib, the software was to be reviewed to ensure high quality and adequate documentation. Although much of the software was to be freely available, the NHSE developed recommendations for handling export controlled and otherwise restricted software.

To enable the establishment of discipline-oriented software repositories, the NHSE project developed the Repository in a Box (RIB) toolkit. RIB makes use of a Web server and an underlying database and facilitates creation and maintenance of interoperable, Web-based metadata repositories. To achieve interoperability, RIB is based on standards, including the IEEE Basic Interoperability Data Model (BIDM) for exchanging software metadata and the WWW Consortium's eXtensible Markup Language (XML).

A number of organizations, including the National Aeronautics and Space Administration's (NASA) Goddard Space Flight Center (GSFC) and Jet Propulsion Laboratory (JPL), the National Science Foundation (NSF) PACI sites, and the Department of Defense's (DoD) Major Shared Resource Centers (MSRCs), have used RIB to set up software repositories in such areas as programming tools, computational chemistry, signal-image processing, and challenge applications. The NHSE itself maintains repositories in the areas of high performance math software, parallel tools, and benchmark programs, and provides a high level view of all the repositories with which it interoperates. Many of these repositories are located here at ICL and are maintained by discipline experts within our group.

To promote systematic review of high performance computing (HPC) software, the NHSE helped develop an IEEE standard extension to the BIDM called the Asset Certification Framework (ACF). The ACF is based on the notion of levels and provides a framework within which different organizations can describe their software review and certification policies and procedures so that the resulting metadata can be exchanged with and understood by other organizations. The NHSE has developed its own review levels and criteria that fit within the ACF.

The NHSE led the development of another IEEE standard extension to the BIDM, called the Intellectual Property Rights Framework (IPRF), which organizations can use to describe software access restrictions and property rights such as copyrights, export restrictions, and licensing requirements. In addition, the NHSE has developed its own extensions to the BIDM for describing software deployment (i.e., where software is installed and how to use it) and benchmark and performance results. The extensible data modeling approach based on the BIDM allows all relevant information about HPC software to be maintained and accessed through a common interface.

In the area of software review, the NHSE has also produced several issues of an electronic journal called the *NHSE Review*. The *NHSE Review* has published articles by experts on comparative evaluations of different types of HPC software, including batch queuing systems, Message Passing Interface (MPI) implementations, debugging and performance analysis tools, and iterative methods for solving linear systems. The *Review* is published at ICL and can be found on the NHSE Web site, which is also maintained here at ICL.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/nhse/

**E-mail**
nhse@cs.utk.edu

**Publications**
Browne, S., Dongarra, J., Horner, J., McMahan, P., Wells, S. "National HPCC Software Exchange (NHSE)," *D-Lib Magazine*, 1998 (May). Electronic journal - http://www.dlib.org/dlib/

**Related URLs**
Globus Metacomputing Directory Service (MDS) - http://www.globus.org/mds/
Netlib - http://icl.cs.utk.edu/netlib/
Repository in a Box (RIB) - http://icl.cs.utk.edu/rib/

# RIB

## REPOSITORY IN A BOX

**ICL TEAM** (ALPHABETICAL)
Jack Dongarra
Don Fike
Shirley Moore
Terry Moore
Jeremy Millar
Tammy Race
Scott Wells

Repository In a Box (RIB) is a toolkit for creating and maintaining web-based, interoperable metadata repositories. In this context, a repository is a collection of metadata and a searchable catalog for browsing such metadata. Repositories created with RIB can seamlessly share their data (interoperate) with one another, which is the key functionality of the toolkit. Furthermore, RIB allows the creation of "virtual repositories," repositories that contain no metadata but contain catalogs of metadata gathered through various interoperations.

The RIB software was developed by the National HPCC Software Exchange (NHSE) technical team here at ICL. RIB was originally conceived as a tool to facilitate the exchange and reuse of high performance applications within the HPCC community, and this remains its primary use.

In a general sense, RIB is a large CGI application. It consists of a number of CGI scripts written in Perl, a backend SQL database (mySQL), and a Java applet. The toolkit provides everything needed to produce and manage repositories, including an HTTP server, its own Perl interpreter, and other essentials – hence the "In a Box" moniker.

Each repository created with RIB contains a data model to which the cataloged metadata conforms. The data models supported by RIB are entity-relationship models and are extremely flexible. RIB uses a default data model that has been standardized by the Institute of Electrical and Electronics Engineers (IEEE). This standard, IEEE Std. 1402 - Basic Interoperability Data Model (BIDM), defines the minimal set of information necessary for the exchange of digital library objects between libraries. The primary object defined by the BIDM is an "Asset." This object class is often used to describe a particular piece of software, e.g., a library (ScaLAPACK) or application (Globus). Other object classes defined by the BIDM include Organization, Element (files, etc.), and Library. Each of these classes defines several descriptive attributes. For example, the Asset class defines descriptive attributes, such as abstract, cost, and version. Figure 1 shows an illustration of RIB's interoperability function.

Since publication of the BIDM standard, IEEE and NHSE have defined several extensions to the basic BIDM data model. IEEE extensions include the 1420.A - Asset Certification Framework (ACF) and 1420.B - Intellectual Property Rights Framework (IPRF). These are official extensions to the standard. NHSE extensions, while not standard, are widely deployed throughout the RIB community. These include minor additions to the Asset class, and the addition of Machine and Deployment classes. The addition of the two classes mentioned above allows information regarding software deployments to be cataloged and published.

RIB supports each of these data models as well as the ability to support many others. Additionally, RIB provides a data model editor so that a repository administrator can extend existing models or create new ones. As a metadata management and interoperation tool, RIB is capable of building catalogs of any type of object that can be described by an entity-relationship data model.

RIB also uses the World Wide Web (W3C) Consortium's eXtensible Markup Language (XML) standard to achieve maximum interoperability. Coupled with the RIB application programmer interface (RIB API), XML allows metadata to be shared seamlessly between repositories and allows RIB to export metadata in a readily understandable manner.

RIB generates catalogs based on a repository's data model and the metadata contained within its database. The catalogs are structured hierarchically, based on a key attribute called a domain, which is typically a broad subject area. Each catalog page is dynamically generated by RIB so that updates to the repository propagate immediately. This is in contrast to systems such as Netlib, for example, where updates become visible the next day.

RIB provides the ability to form "joins" for repositories with a properly structured data model. In order to use this functionality, the data model must define at least one "intersection class." Once this is done, RIB can dynamically generate tables depicting the join. This feature is often used to generate software deployment tables with software along one axis, machines along the other, and deployment status as the table data.

Currently, RIB is in release 2.1 and has been deployed by various government organizations,

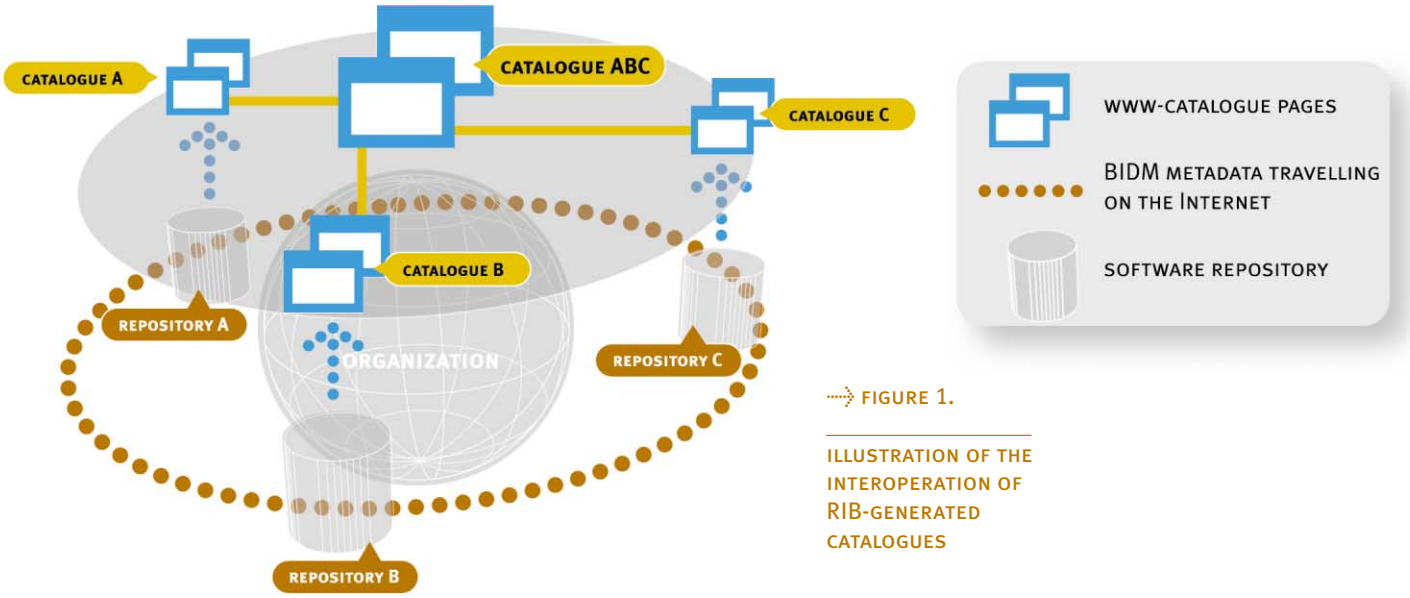



FIGURE 1.

ILLUSTRATION OF THE INTEROPERATION OF RIB-GENERATED CATALOGUES

including DoD, NASA, DOE, and NSF. Commercial/industrial customers such as Raytheon have also begun using the toolkit. Additionally, the NHSE maintains several RIB repositories.

The current release of RIB runs on several widely available platforms, including Linux and Windows. Development of RIB presently focuses on performance enhancements and security issues. On the performance front, support for mod_perl is being added to the Windows version of RIB. This will bring the performance of Windows-based repositories to the same level as those hosted on Unix systems. Regarding security, the RIB team is adding support for SSL and Kerberos. A redesign of the RIB search engine is also planned.

Other projects utilizing RIB include the Globus and Grid Applications Development Software (GrADS) projects. These are wide-scale, distributed computing or grid computing projects. Central to grid computing is the idea of transparent, ubiquitous access to large-scale compute, data, and instrument resources. Ultimately, grid users will not need to specify on which machines to run a computational job. Rather, the user will specify resource requirements (e.g., memory, network bandwidth), and the system will dynamically discover the best machine to run the job.

In order to make intelligent choices about where to run a job, the grid environment must possess a large amount of information concerning each compute resource. In particular, software deployment information will be required.

RIB provides software deployment information to both the Globus and GrADS environments via the RIB API. For the Globus project, this is accomplished by exporting the RIB metadata to the native Globus information service.

The GrADS project, on the other hand, makes more use of RIB's capabilities. Each institution involved in GrADS is deploying a RIB repository to manage local software deployment information. These repositories will then be gathered into a large GrADS repository via RIB's interoperability features.

Additionally, the RIB team is developing tools to automate the gathering and publishing of metadata. Work in this area is focused on automatic generation of machine characteristics and automatic software performance evaluation.

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/rib/

**E-mail**
rib@cs.utk.edu

**Publications**
Browne, S., Dongarra, J., Horner, J., McMahan, P., Wells, S. "Technologies for Repository Interoperation and Access Control," University of Tennessee Computer Science Department *Technical Report*, UT-CS-98-395. April 1998.

Browne, S., McMahan, P., Wells, S. "Repository in a Box Toolkit for Software and Resource Sharing," University of Tennessee Computer Science Department *Technical Report*, UT-CS-99-424. May 1999.

Dongarra, J., McMahan, P., Millar, J. "RIBAPI - Repository in a Box Application Programmer's Interface," University of Tennessee Computer Science Department *Technical Report*, UT-CS-00-438. January 2000.

**Agency Funding**
Department of Defense (DoD)
National Aeronautics and Space Administration (NASA)

**Related URLs**
eXtensible Markup Language (XML) - http://www.w3.org/XML/
Globus Metacomputing Directory Service (MDS) - http://www.globus.org/mds/
Grid Application Development Software (GRaDS) project - http://www.hipersoft.rice.edu/grads/
National HPCC Software Exchange (NHSE) - http://www.nhse.org/
Netlib - http://www.netlib.org/

# LINPACK Benchmark

ICL TEAM (ALPHABETICAL)
Jack Dongarra
Antoine Petitet

The LINPACK Benchmark is in some sense an accident. It was originally designed to assist users of the LINPACK package by providing information on the execution times required to solve a system of linear equations. LINPACK is a collection of Fortran routines designed to solve various systems of linear equations. The package itself is based on another package called the Basic Linear Algebra Subprograms, today referred to as the Level 1 BLAS. The first "LINPACK Benchmark report" appeared as an appendix in the LINPACK Users' Guide in 1979 (See figure 1).

The appendix comprised data for one commonly used path in LINPACK for a matrix problem of size 100 on a collection of widely used computers (23 in all), so users could estimate the time required to solve their matrix problems. Over the years, additional data was added, more as a hobby than any thing else, and today the collection includes thousands of different computer systems all measured using the same piece of software. In addition to adding more computer systems' performance to the list, the scope of the benchmark has also changed. The benchmark today reports four basic sets of performance numbers: the execution rate for solving a system of equations of order 100 using the Fortran LINPACK Software, the execution rate for solving a system of equations of order 1000 using the best method (which does not have to be Fortran), the asymptotic execution rate for solving a system of equations (High Performance LINPACK (HPL)), and the theoretical execution rate for the computer system.

In order to have an entry included in the LINPACK Benchmark report, the results must be computed using full precision. By full precision, we generally mean 64-bit floating point arithmetic or higher. Note that this is not an issue of single or double precision since some systems have 64-bit floating point arithmetic as single precision. It is a function of the arithmetic used. More precisely, the solution to all three benchmarks must satisfy the following mathematical formula:

$$\frac{\| Ax-b \|}{\| A \| \| x \| n\varepsilon} \leq O(1)$$

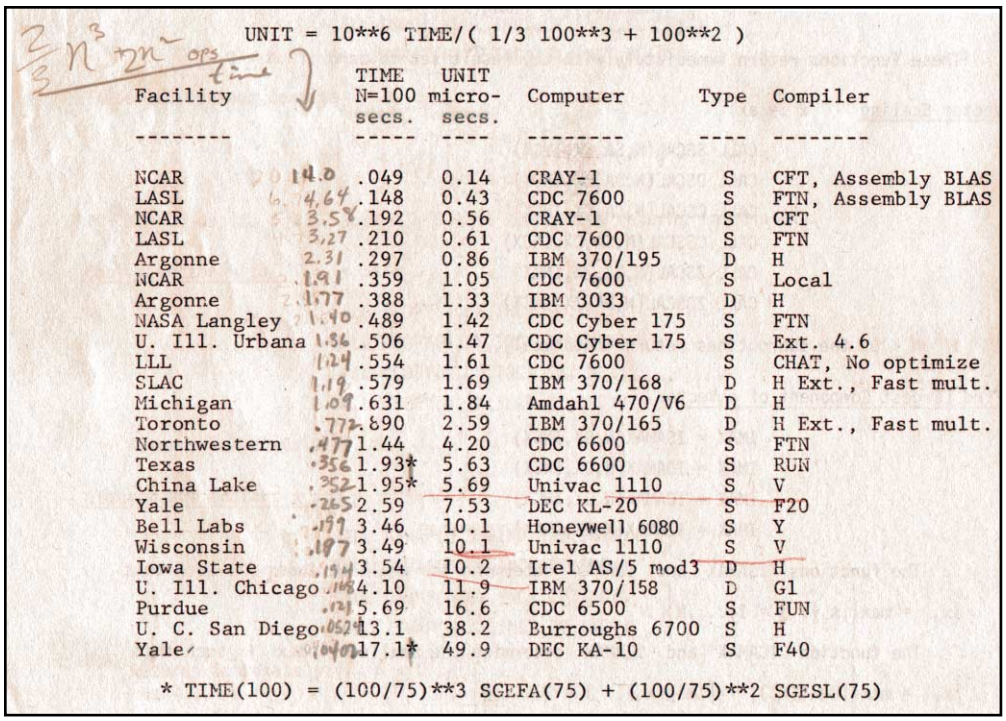
,where ε = the machine precision (On IEEE machines this is $2^{-53}$) and $n$ is the size of the problem.

HPL is a software package that solves a (random) dense linear system in double precision (64 bits) arithmetic on distributed-memory computers. It can thus be regarded as a portable, as well as freely available, implementation of the HPL Benchmark.

The algorithm used by HPL can be summarized by the following keywords: Two dimensional block-cyclic data distribution; Right-looking variant of the LU factorization with row partial pivoting featuring multiple look-ahead depths; Recursive panel factorization with pivot search and column broadcast combined; Various virtual panel broadcast topologies; bandwidth reducing swap-broadcast algorithm; backward substitution with look-ahead of depth 1.

The HPL package provides a testing and timing program to quantify the accuracy of the obtained solution as well as the time it took to compute it. The best performance achievable by this software on a system depends on a large variety of factors. Nonetheless, with some restrictive assumptions on the interconnection network, the algorithm described here and its attached implementation are scalable in the sense that their parallel efficiency is maintained constant with respect to the per processor memory usage.

The HPL software package requires the on system availability of an implementation of the Message Passing Interface (MPI) (1.1 compliant). An implementation of either the Basic Linear Algebra Subprograms (BLAS) or the Vector Signal Image Processing Library (VSIPL) is also needed. Machine-specific as well as generic implementations of MPI, the BLAS, and VSIPL are available for a large variety of systems.



UNIT = 10**6 TIME/( 1/3 100**3 + 100**2 )

| Facility | TIME N=100 secs. | UNIT micro- secs. | Computer | Type | Compiler |
|---|---|---|---|---|---|
| NCAR | .049 | 0.14 | CRAY-1 | S | CFT, Assembly BLAS |
| LASL | .148 | 0.43 | CDC 7600 | S | FTN, Assembly BLAS |
| NCAR | .192 | 0.56 | CRAY-1 | S | CFT |
| LASL | .210 | 0.61 | CDC 7600 | S | FTN |
| Argonne | .297 | 0.86 | IBM 370/195 | D | H |
| NCAR | .359 | 1.05 | CDC 7600 | S | Local |
| Argonne | .388 | 1.33 | IBM 3033 | D | H |
| NASA Langley | .489 | 1.42 | CDC Cyber 175 | S | FTN |
| U. Ill. Urbana | .506 | 1.47 | CDC Cyber 175 | S | Ext. 4.6 |
| LLL | .554 | 1.61 | CDC 7600 | S | CHAT, No optimize |
| SLAC | .579 | 1.69 | IBM 370/168 | D | H Ext., Fast mult. |
| Michigan | .631 | 1.84 | Amdahl 470/V6 | D | H |
| Toronto | .690 | 2.59 | IBM 370/165 | D | H Ext., Fast mult. |
| Northwestern | 1.44 | 4.20 | CDC 6600 | S | FTN |
| Texas | 1.93* | 5.63 | CDC 6600 | S | RUN |
| China Lake | 1.95* | 5.69 | Univac 1110 | S | V |
| Yale | 2.59 | 7.53 | DEC KL-20 | S | F20 |
| Bell Labs | 3.46 | 10.1 | Honeywell 6080 | S | Y |
| Wisconsin | 3.49 | 10.1 | Univac 1110 | S | V |
| Iowa State | 3.54 | 10.2 | Itel AS/5 mod3 | D | H |
| U. Ill. Chicago | 4.10 | 11.9 | IBM 370/158 | D | G1 |
| Purdue | 5.69 | 16.6 | CDC 6500 | S | FUN |
| U. C. San Diego | 13.1 | 38.2 | Burroughs 6700 | S | H |
| Yale | 17.1* | 49.9 | DEC KA-10 | S | F40 |

* TIME(100) = (100/75)**3 SGEFA(75) + (100/75)**2 SGESL(75)

FIGURE 1.

THE FIRST "LINPACK BENCHMARK REPORT" FROM *LINPACK USERS' GUIDE* 1979

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/linpack-benchmark/

**E-mail**
linpack-benchmark@cs.utk.edu

**Publications**
Dongarra, J. "Performance of Various Computers Using Standard Linear Equations Software," University of Tennessee Computer Science Department *Technical Report*, 2000.
http://www.netlib.org/benchmark/performance.ps

**Agency Funding**
Department of Energy (DOE) (for High-Performance Linpack)

**Related URLs**
Basic Linear Algebra Subprograms (BLAS) - http://www.netlib.org/blas/
High Performance Linpack (HPL) - http://icl.cs.utk.edu/hpl/
Message Passing Interface (MPI) - http://www-unix.mcs.anl.gov/mpi/
Vector Signal Image Processing Library (VSIPL) - http://www.vsip.org/

**FAQ**
http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html

# PAPI

## PERFORMANCE APPLICATION PROGRAMMING INTERFACE

**ICL TEAM (ALPHABETICAL)**

Leon Dong (G)

Jack Dongarra

Kevin London

Shirley Moore

Phil Mucci

Keith Seymour

Thomas Spencer (U)

Luke Zhou (G)

(G) = GRADUATE STUDENT

(U) = UNDERGRADUATE STUDENT

Collecting data about an application's performance has long been an imprecise art. In the past, the user had to rely on low resolution timers with differing semantics and imprecise order-of-magnitude estimates about the number and kind of "operations" performed during the execution of the program. Today, hardware performance counters exist on every major microprocessor platform. These counters accumulate the occurrences of selectable "events." In the abstract sense, an event can be defined as a sequence of the usage of hardware resources, both on and off the CPU. Examples of these events include instructions executed, cache misses, and branch mispredictions. These counters can provide application developers valuable information about the performance of their programs with a high degree of accuracy. In addition, the relationships between measurable events are often indicative of ways to improve performance. Compiler writers and tool developers can use hardware counters as a basis for automated performance analysis, diagnosis, and possibly correction. However, the integration of hardware performance metrics into modern software components has been slow and sometimes nonexistent. This is largely due to the fact that access to these counters has been quite substandard. Many of the implementations that do exist either suffer from an overly complicated interface, poor documentation, or unavailable software requirements.

The purpose of the PAPI project is to define a standardized, easy to use interface that provides access to the hardware performance counters on most major processor platforms, thereby providing application developers the information they need to tune their software on different platforms. The goal is to make it easy for users to gain access to the counters to aid in performance analysis, modeling, and tuning.

The approach of the PAPI project has been to work with the high performance computing (HPC) community, including users and vendors, to choose a common set of hardware events and to define a cross platform library interface to the underlying counter hardware. These common events are those considered to be most relevant and useful in tuning application performance. As large a subset as possible has been mapped to the corresponding machine-specific events on the major HPC platforms. It is our hope that most of these events will be made available on all major HPC platforms in the future to improve the capability for tuning applications across multiple platforms. The intent is that the same metric would count similar, and possibly comparable, events on different platforms. Direct comparison between systems is not the intention of the PAPI predefined events. Rather, the intent is to standardize the names for the metrics, not the exact semantics, which necessarily depend on the processor under study.

The PAPI library provides two interfaces to the underlying counter hardware: a high level interface for the acquisition of simple measurements, and a fully programmable, low level interface directed towards users with more sophisticated needs. The high level interface simply provides the capability to start, stop, and read the counters for a specified list of events. The target audience for the high level interface is application engineers and benchmarking teams looking to quickly and simply acquire some rudimentary performance measurements. The tool developer will likely find the high level interface too restrictive.

The low level interface provides the more sophisticated functionality of user callbacks on counter overflow and hardware based SVR4 compatible profiling, regardless of whether or not the operating system supports it. These features provide the necessary basis for any source level performance analysis software. Thus, for any architecture with even the most rudimentary access to hardware performance counters, PAPI provides the foundation for truly portable, source level performance analysis tools based on real processor statistics.

Internally, the PAPI implementation is divided into portable and machine dependent layers. The top portable layer consists of the high and low level PAPI interfaces. This layer is completely machine independent and requires little porting effort. It contains all of the API functions as well as numerous utility functions that perform state handling, memory management, data structure manipulation, and

thread safety. In addition, this layer provides advanced functionality not always provided by the operating system, namely event profiling and overflow handling. The portable layer calls the machine dependent substrate, which is the internal PAPI layer that handles the machine dependent specifics of accessing the counters. For each architecture/operating system pair, only a new substrate layer needs to be written. Experience indicates that no more than a few weeks are required to generate a fully functional substrate for a new platform, if the operating system provides the necessary support for accessing the hardware counters.

Using traditional tools to obtain performance data for large-scale applications can be cumbersome and inefficient. The data collected may consist only of summary statistics and may provide little insight into the dynamic runtime behavior of the application. Most tools require that the data to be collected be specified prior to runtime and do not allow for runtime selection or adjustment of which events to measure or the level of detail or granularity at which measurements should be made. To address these issues, the tool infrastructure being developed to complement the PAPI library facilitates runtime tracing and analysis of hardware counter data, as well as runtime control over performance instrumentation. Utility routines layered on top of the PAPI library enable application developers to easily delineate interesting portions of the application, such as subroutines and loop nests, and to output hardware counter data at runtime to interactive end user performance analysis tools. In addition, a dynamic instrumentation capability currently under development will provide mechanisms for calls to PAPI library and utility routines to be dynamically inserted into and removed from running applications, thus allowing runtime control over the selection of hardware events and of counting modes and granularity.

PAPI aims to provide the tool designer and application engineer with a consistent interface and methodology for use of the performance counter hardware found in most major microprocessor lines. The main motivation for this interface is the increasing divergence of application performance from near peak performance of most machines in the HPC marketplace. This performance gap is largely due to differences in memory and communication bandwidth at different levels of the memory hierarchy. To address this problem, users need a compact set of robust and useful tools to quickly diagnose and analyze processor specific performance metrics. PAPI focuses on developing a reusable, portable, and functionality oriented infrastructure for performance tool design and advanced program analysis.

Since PAPI is Open Source, it has been integrated by a number of tool developers. Pacific Sierra Research has incorporated PAPI into their multi-platform parallel performance analysis tool, DEEP/MPI. Sandia Livermore National Laboratory is using PAPI to develop tools for tuning quantum chemistry simulations. Lawrence Livermore National Laboratory is using PAPI in tools to tune the performance of codes used in monitoring the safety of the nuclear stockpile. PAPI is installed at most of the Department of Defense (DoD) Major Shared Resource Centers (MSRCs) and PAPI is used outside the US as well, with reports of use coming from Japan, Switzerland, Italy, Sweden and Greece.

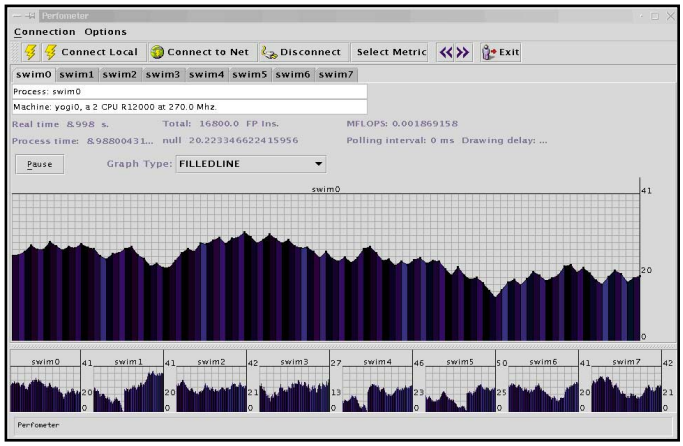



FIGURE 1. SCREEN CAPTURE OF PAPI'S PERFOMETER IN USE

## RESOURCES

**Web Site**
http://icl.cs.utk.edu/papi/

**E-mail**
papi@cs.utk.edu

**Papers**

Browne, S., Deane, C., Ho, G., Mucci, P. "PAPI: A Portable Interface to Hardware Performance Counters," *Proceedings of Department of Defense HPCMP Users Group Conference*, June 1999.

Browne, S., Dongarra, J., Garner, N., Ho, G., Mucci, P. "A Portable Programming Interface for Performance Evaluation on Modern Processors," *The International Journal of High Performance Computing Applications*, Vol. 14, Number 3 (Fall 2000): 189-204.



**Agency Funding**
Department of Defense (DoD)
Department of Energy (DOE)
National Science Foundation (NSF)

**Industrial Support**
IBM


**Related URLs**
DoD Major Shared Resource Centers - http://www.hpcmo.hpc.mil/Htdocs/MSRC/index.html
Lawrence Livermore National Laboratory - http://www.llnl.gov/
Pacific Sierra Research - http://www.psrw.com/
Sandia Livermore National Laboratory - http://www.sandia.gov/

**FAQ**
http://icl.cs.utk.edu/projects/papi/faq.html

# TOP500
## Supercomputers

# TOP100
## Clusters

**ICL TEAM**
Erich Strohmaier
Jack Dongarra

**COLLABORATORS**
Hans Meuer, University of Mannheim

For the first time, in 1993, a list of the top 500 supercomputer sites worldwide was made available. Every year since, the TOP500 list has been published bi-annually. The best Linpack Benchmark performance is used as a performance measure in ranking the computers. The list allows a detailed and well-founded analysis of the state of high performance computing (HPC). Previously, data such as the number and geographical distribution of supercomputer installations were difficult to obtain and only a few analysts undertook such an effort by tracking press releases of dozens of vendors. Data for the TOP500 are submitted by manufacturers of high performance computing systems as well as from users and managers of sites owning such systems. These submissions are reviewed by the TOP500 maintainers and by independent reviewers to ensure consistent, high quality data. With the TOP500 report now easily available, it is possible to present an analysis of the state of HPC.

While many aspects of the HPC market change dynamically over time, the evolution of performance seems to follow some empirical laws such as Moore's law. The TOP500 provides an ideal data basis to verify such an observation. By examining the computing power of the individual machines present in the TOP500 and the evolution of the total installed performance, we plot the performance of the systems at positions one, ten, 100 and 500 in the list as well as the total accumulated performance. In figure 1, the curve of position 500 shows, on average, an increase of a factor of two within one year. All other curves show a growth rate of 1.8 (+/- 0.07) per year.

Based on the current TOP500 data, which cover the last seven years, and the assumption that the current performance development continues for the near future, we can now extrapolate the observed performance and compare these values with the goals of the DOE ASCI program in the USA and the Earth Simulator project in Japan. In figure 2, we extrapolate the observed performance values using linear regression on the logarithmic scale. In other words, we fit exponential growth to all levels of performance in the TOP500. These simple interpretations of the data show surprisingly consistent results. Based on the extrapolation from these interpretations, we can expect to have the first ~100 TFlop/s systems by the year 2005, which is about one to two years later than the ASCI path projected plans. Conversely, by the year 2005 we would also expect that no system smaller than ~1 TFlop/s will be able to make the TOP500.

Looking even further into the future, we could speculate that, based on the current doubling of performance every year, the first Petaflop system would be available around 2009. Currently due to the rapid changes in the technologies used in HPC systems, there is no reasonable projection possible for the architecture of a Petaflop system at the end of the next decade. Even as the HPC market has changed quite substantially since the introduction of the Cray 1 three decades ago, there is no end in sight for such rapid cycles of redefinition.

In an effort to continue the analysis of trends in the HPC area, we have initiated a study of the top 100 clusters and we are planning to rank the top 100 cluster sites. This initially will take the same form as the TOP500 list. There is also a plan to change the performance metric after some experience is gained with the existing set of systems. The new metric will include floating point performance, communication speed, as well as I/O performance.

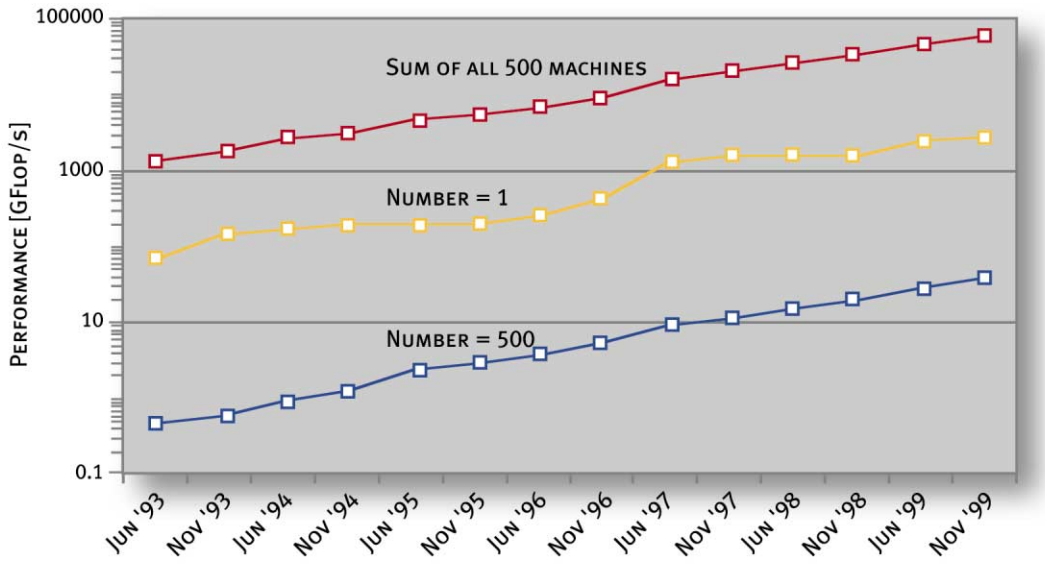



FIGURE 1.

CHART DEPICTING THE OVERALL GROWTH OF ACCUMULATED AND INDIVIDUAL PERFORMANCE AS SEEN IN THE TOP500

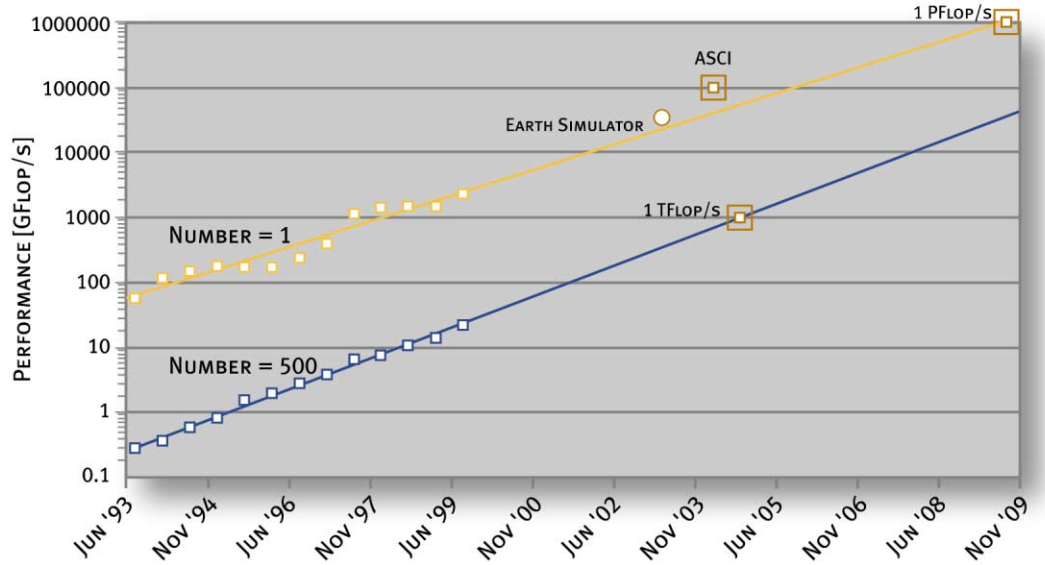



FIGURE 2.

CHART DEPICTING THE EXTRAPOLATION OF RECENT GROWTH RATES OF PERFORMANCE SEEN IN THE TOP500

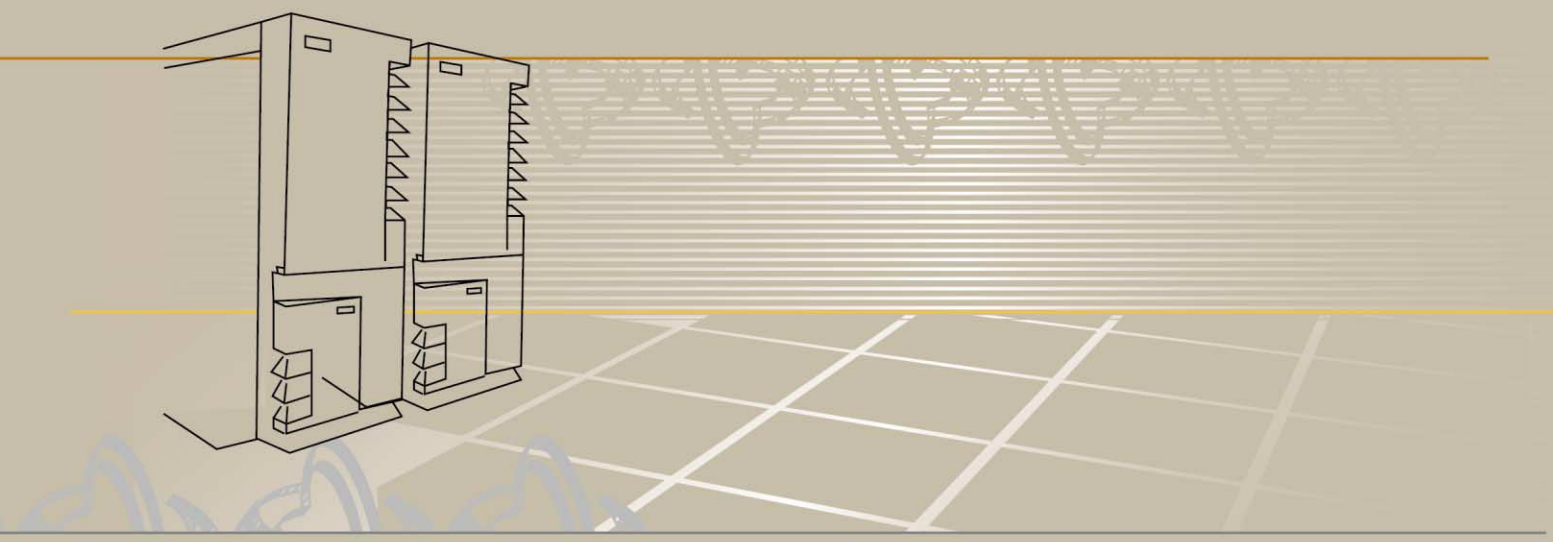
## RESOURCES

**Web Site**
http://icl.cs.utk.edu/top500/

**E-mail**
top500@cs.utk.edu

**Related URLs**
DOE ASCI - http://www.asci.doe.gov/overview/overview.htm
Earth Simulator project - http://www.gaia.jaeri.go.jp/

**FAQ**
http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html



# HARDWARE RESOURCES

Research in the many areas of high performance computing requires access to various hardware resources. As a research group, we take pride in our hardware assets, which consist of 48 desktop computers ranging from Dell Windows machines to Sun Sparc workstations. We also have 20 machines currently configured as servers. In addition, we also have a host of parallel computing machines, which include high-end architectures such as two IBM Power 3s, a commodity-based production PC cluster consisting of 22 Intel-class machines, a non-production cluster consisting of 12 machines of various configurations. We also have an SGI Octane and two Compaq Alphas. Our close working relationship with hardware vendors has been very beneficial due to the fact that over 30 of our desktops and servers were either donated or are on loan.

As part of the UT Computer Science Department, we also retain access to many other resources including several server class machines as well as four high performance computing (HPC) clusters. These clusters include 32 Dual PIII 500Mhz machines, 16 Dual PIII 550Mhz machines, 17 Sun Enterprise 220R 450MHz, and four Quad PIII Xeon 500Mhz boxes. These are arranged in the classic Beowulf cluster configuration in which machines are connected by low latency, high speed network switches.

Because many of our staff frequently travel to conferences, workshops, and other events, it is important that they have access to mobile computing resources. To this end, we have acquired nearly 34 laptops. These also give us the ability to regularly give demonstrations, tutorials, and presentations to our partners and funding agencies around the country.

In addition to the local resources at our disposal, we are also fortunate to have access to remote resources. Due to the many organizations and institutions with which we collaborate, a wide range of hardware architectures are made available to our research staff. Remaining in the forefront of computational research requires our staff to have access to the latest computing technology. Below are some of the architectures currently available for our use:

- IBM SPs
- Cray SV-1s and T3Es
- Sun E10000s
- SGI Origin 2000s and 3000s
- Compaq ES40s

Our ability to harness the computing power of multiple architectures allows us to parallelize many applications that previously ran only serially in addition to performing software development and testing.

# PEOPLE

As is the case with most organizations, our success is driven by our employees. Equally important are the working relationships we have established with individuals and organizations within the high performance computing (HPC) community. Our staff, our partners and collaborators, and the many commercial vendors with which we work have helped us create an extremely strong foundation for fostering creative, original research. We have worked hard to create and maintain many collaborative relationships and are always eager to open doors to new opportunities for sharing research endeavors.

We have been very successful at attracting experts and top researchers that comprise our staff. With a staff numbering nearly 50, we are able to apply adequate people resources to the projects on which we work. Currently, we employ two undergraduate students, 14 graduate students, and 29 full or part time staff, many of whom worked with us as students themselves. Table 1 provides information about our current staff. Because ICL is known internationally as a leading HPC research group, we have been successful in attracting research scientists from around the world. Proudly, our staff includes representatives from the United States, China, France, Germany, India, Italy, Japan, Mexico, and the Netherlands. Our ability to attract such experts from around the world is only one reason ICL remains an HPC research leader.

Equally important to our group are our students (see table 2). As part of the computer science (CS) department of a large university, ICL has access to both graduate and undergraduate students. With a CS program consisting of 198 students, additional help with our projects is just a job posting away. These students represent a resource that is not readily available to many research groups, and we have been very proactive in securing assistantships for those students who are motivated, hard working, and willing to learn.

In addition to our employees, we routinely host numerous visitors from around the globe. While many of our visitors stay briefly to give seminars or presentations, many remain with us for as long as a year collaborating, teaching, and learning. While many of our visitors are professors from various international universities, we also host researchers and administrators from many research institutions. In addition, it is not uncommon to have students (undergraduate as well as graduate) from various universities study with us for months on end, learning about our approaches to computing problems. In fact, many Ph.D. students from universities as far away as Japan have passed through our doors in an effort to broaden their understanding of linear algebra techniques and how we apply them to our research. The experience shared by both our visitors and ourselves has been extremely beneficial to us, and we will continue providing opportunities for visits from our international colleagues in research. See table 3 for the many guests who have stopped by in the last year to exchange ideas and share their expertise with us. It is also our ability to maintain such close working relationships that is crucial to our efforts toward providing original and timely solutions to problems inherent in high performance parallel computing.

With the phenomenal growth over the last several years in parallel computing technology and the demands placed on such technology by government and private business, we are consistently challenged to apply expert-level understanding to each of our research efforts. The areas of distributed and network computing are no exception as we've learned to harness enormous computing power to quickly and efficiently solve mathematical problems that would take humans years or decades to solve by hand.

## TABLE 1. CURRENT ICL STAFF

| NAME | POSITION | DEGREE | PROJECTS |
|---|---|---|---|
| Arnold, Dorian | Research Associate | MS - University of Tennessee, 1998 | NetSolve |
| Bassi, Alex | Research Associate | MS - University of Studies, Milan Italy, 1994 | IBP |
| Beck, Micah | Research Associate Professor | PhD - Cornell University, 1992 | I2-DSI, IBP, SInRG |
| Blackford, Susan | Research Associate | MS - University of Tennessee, 1990 | LAPACK, NetSolve, ScaLAPACK |
| Bukovsky, Antonin | Research Associate | BS - University of Tennessee, 2000 | FT-MPI, HARNESS |
| Cronk, David | Post-Docotral Research Associate | PhD - College of Willaim & Mary, 1998 | Parallel I/O |
| Dongarra, Jack | ICL Director | PhD - University of New Mexico, 1980 | |
| Eijkhout, Victor | Research Assistant Professor | PhD - Catholic University, Nijmegen, Netherlands, 1990 | Sparse Libraries and Algorithms |
| Ellis, Brett | Senior Computer Systems Specialist | BS - University of Tennessee, 1996 | GrADS, SInRG, TORC |
| Fagg, Graham | Research Assistant Professor | PhD - University of Redding, UK, 1998 | FT-MPI, HARNESS, MPI_Connect, Par I/O |
| Fike, Don | Research Associate | Associates - High-Tech Institute, 1988; BS expected 2001 | NHSE, RIB |
| Gangwer, Lynn | Principal Secretary | BS - University of Tennessee, 1999 | |
| Jones, Jan | Publications Coordinator | MS - University of Tennessee, 1990 | |
| Lee, Tracy | Senior Budget Assistant | BA - University of Tennessee, 1995 | |
| London, Kevin | Research Associate | BS expected 2001 | MPI_Connect, PAPI |
| Luczak, Richard | ASC MSRC Programming Tools Onsite Lead | PhD - Technical University of Lodz, Poland, 1988 | DoD Programming tools |
| Millar, Jeremy | Research Associate | BS - University of Tennessee, 2000 | NA-Net, NA-Digest, Netlib, NHSE, RIB |
| Miller, Michelle | Research Associate | MS - University of Utah, 1998 | NetSolve |
| Moore, Shirley | Associate Director for Research | PhD - Purdue University, 1990 | NHSE, RIB, PAPI |
| Moore, Keith | Research Associate | MS - University of Tennessee, 1996 | HARNESS, NA-Net, NA-Digest, NetSolve |
| Moore, Terry | Assoc. Director for Project Development | PhD - University of North Carolina, Chapel Hill, 1993 | I2-DSI, IBP, RIB, SInRG |
| Mucci, Phil | Research Consultant | MS - University of Tennessee, 1998 | PAPI |
| Peltz, Paul | Computer Operations, Systems Programmer | BA expected 2002 | TORC |
| Petitet, Antoine | Research Scientist | PhD - University of Tennessee, 1996 | ATLAS, GrADS, ScaLAPACK |
| Rafferty, Tracy | Manager | | |
| Rogers, David | Graphic Artist | BFA - University of Tennessee, 2000 | |
| Seymour, Keith | Research Associate | MS - University of Tennessee, 1997 | F2J, PAPI |
| Strohmaier, Erich | Post-Doctoral Research Associate | PhD - University of Heidelberg, Germany, 1990 | Parkbench, TOP500 |
| Wells, Scott | Assistant Director for Communications | MS - University of Tennessee, 1996 | NHSE, RIB |
| Whaley, R. Clint | Senior Research Associate | MS - University of Tennessee, 1994 | ATLAS |

## TABLE 2. CURRENT STUDENTS

| NAME | POSITION | PROJECTS |
|---|---|---|
| Agrawal, Sudesh | Graduate Research Assistant | NetSolve |
| Dong, Leon | Graduate Research Assistant | PAPI |
| Downey, Andrew | Graduate Research Assistant | Parkbench |
| Fuentes, Erika | Graduate Research Assistant | Support |
| Henderson, David | Undergraduate Research Assistant | Parkbench |
| Kannon, Balajee | Graduate Research Assistant | IBP |
| Luszczeck, Piotr | Graduate Research Assistant | Sparse libraries and algorithms |
| Liu, Chaoyang | Graduate Research Assistant | I2-DSI |
| Huang, Yan | Graduate Research Assistant | NetSolve |
| Race, Tammy | Graduate Research Assistant | RIB |
| Roche, Ken | Graduate Research Assistant | Sparse libraries and algorithms |
| Spencer, Thomas | Undergraduate Research Assistant | PAPI |
| Thomas, Joe | Graduate Research Assistant | Support |
| Vadhiyar, Sathish | Graduate Research Assistant | FT-MPI, NetSolve |
| Wo, Susan | Graduate Research Assistant | IBP |
| Zhou, Luke | Graduate Research Assistant | PAPI |

## TABLE 3. RECENT VISITORS

| NAME | VISITED | FROM |
|---|---|---|
| Thara Angskun | March - July 2000 | Kasetsart University - Bangkok, Thialand |
| Mark Baker | August 2000 | Univ. of Portsmouth - Portsmouth, UK |
| Ralph Byers | May 2000 | University of Kansas |
| Pierre-Henri Cros | February 2000 | CERFACS - Toulouse, France |
| Georges Da Costa | June 2000 | École Normale de Superieure - Lyon, France |
| Paraskevas Evripidou | October 1999 | University of Cyprus |
| Markus Fischer | July - August 2000 | University of Mannheim - Mannheim, Germany |
| Walter Gander | May 2000 | Eidgenossische Technische Hochschule - Zurich, Switzerland |
| Hirokazu Hasegawa | June 2000 | Tokyo Institute of Technology - Toyko, Japan |
| Constantinos Ierotheou | March 2000 | University of Greenwich - London, UK |
| Laurent Lefevre | July 2000 | Universite Claude Bernard - Lyon, France |
| Mikel Lujan | July 2000 | Manchester University - Manchester, UK |
| Satoshi Matsuoka | July 2000 | Tokyo Institute of Technology - Toyko, Japan |
| MickaËl Melki | June 2000 | École Normale de Superieure - Lyon, France |
| Jose Miguel | July 2000 | University of the Basque Country in Spain |
| Jakob Oestergaard | August 2000 | Danish Technical University - Lyngby, Denmark |
| Peter Soendergaard | September 2000 | Danish Technical University - Lyngby, Denmark |
| Danny Sorensen | October 1999 | Rice University |
| Toyotaro Suzumura | May 2000 | Tokyo Institute of Technology - Toyko, Japan |
| Osamu Tatebe | March 2000 | Electrotechnical Laboratory, AIST, MITI - Tsukuba, Japan |
| Henk van der Vorst | October 2000 | Mathematical Institute - Utrecht University, Netherlands |
| Putchong Uthayopas | July 2000 | Kasetsart University - Bangkok, Thailand |

## TABLE 4. GRADUATED STUDENTS OF JACK DONGARRA

| NAME | DEGREE | CURRENT EMPLOYER |
|---|---|---|
| Barrett, Richard | MS Computer Science (University of Tennessee), 1994 | Los Alamos National Lab |
| Casanova, Henri | PhD Computer Science (University of Tennessee), 1998 | University of California, San Diego |
| Duitz, Mitchell | MS Computer Science (University of Tennessee), 1991 | |
| Garner, Nathan | MS Computer Science (University of Tennessee), 1999 | Cleveland State Community College |
| Green , Stan | MS Computer Science (University of Tennessee), 1994 | |
| Ho, George | MS Computer Science (University of Tennessee), 1999 | Digital Island |
| Kalhan, Ajay | MS Computer Science (University of Tennessee), 1996 | Microsoft |
| Kim, Youngbae | PhD Computer Science (University of Tennessee), 1996 | |
| Larose, Brian | MS Computer Science (University of Tennessee), 1993 | Hewlett-Packard |
| Liebrock, Lorie | PhD Computer Science (Rice University), 1994 | University of Alaska |
| Manchek, Robert | MS Computer Science (University of Tennessee), 1994 | SANgate Systems Inc. |
| McMahan, Paul | MS Computer Science (University of Tennessee), 1999 | IBM |
| Moulton, Steve | MS Computer Science (University of Tennessee), 1992 | |
| Mucci, Phil | MS Computer Science (University of Tennessee), 1998 | ICL/Consultant |
| Nypaver, Delphy | MS Computer Science (University of Tennessee), 1997 | Oak Ridge National Laboratory |
| Payne, James | MS Computer Science (University of Tennessee), 1994 | |
| Petitet, Antoine | PhD Computer Science (University of Tennessee), 1996 | UT/ICL |
| Phillips, Rick | MS Computer Science (University of Tennessee), 1998 | Oak Ridge National Laboratory |
| Raman, Ganapathy | MS Computer Science (University of Tennessee), 2000 | Microsoft |
| Sidani, Majed | PhD Mathematics (University of Tennessee), 1992 | Merrill Lynch, CICG Development |
| Whaley, Clint | MS Computer Science (University of Tennessee), 1993 | UT/ICL |
| Xu, Tinghua | MS Computer Science (University of Tennessee), 2000 | |

## TABLE 5. FORMER STAFF, STUDENTS, AND RESIDENT VISITORS OF ICL

| NAME | POSITION WITH ICL, DATES | CURRENT EMPLOYER |
|---|---|---|
| Aebischer, Carolyn | Graduate Student, 1990-1993 | |
| Anderson, Ed | Research Associate, 1989-1991 | EPA - Ntl Environmental Supercomputing Ctr |
| Bai, Zhaojun | Post Doc, 1990-1992 | University of California, Davis |
| Beguelin, Adam | Post Doc, 1991-1994 | Inktomi |
| Benzoni, Annamaria | Visiting Research Associate 1991 | IBM Rome Scientific Center |
| Betts, Scott | Undergraduate Student, 1997-1998 | |
| Bond, Leon | Principal Secretary, 1999-2000 | Carr America |
| Brown, Randy | Undergraduate Student, 1997-1999 | |
| Bunch, Greg | | |
| Casanova, Henri | GRA and Post Doc, 1995-1998 | University of California, San Diego |
| Chambers, Sharon | Undergraduate Student, 1998-2000 | |
| Choi, Jaeyoung | Post Doc, 1994-1996 | Soongsil University - Seoul, South Korea |
| Clarkson, Eric | Artist, 1998-1999 | |

## TABLE 5. FORMER STAFF, STUDENTS, AND RESIDENT VISITORS OF ICL continued

| NAME | POSITION WITH ICL/DATES | CURRENT EMPLOYER |
|---|---|---|
| Cleary, Andy | Post Doc, 1995-1997 | Data Digest |
| Cox, Jason | Graduate Student, 1993-1997 | |
| Deane, Cricket | Research Associate, 1998-1999 | |
| Desprez, Frederic | Post Doc, 1994-1995 | ÉNS - Lyon |
| Drake, Mary | Office Supervisor, 1989-1992 | University of Tennessee, JICS |
| Eyler-Walker, Zach | Undergraduate Student, 1998 | |
| Fischer, Markus | Visiting Student, 1997-1998 | University of Mannheim |
| Garner, Nathan | GRA and Research Associate, 1998-2000 | Cleveland State Community College |
| Green, Stan | GRA and Sr. Research Associate, 1992-1996 | |
| Hammarling, Sven | Visiting Professor, 1996-1997 | NAG Ltd. |
| Hasegawa, Hidehiko | Visiting Research Associate 1995–1996 | Univ of Library & Info. Science - Tsukuba |
| Hasegawa, Satomi | Visiting Research Associate 1995-1996 | Hitachi Computer |
| Henry, Greg | Post Doc, 1996-1996 | Intel |
| Hill, Sid | Undergraduate Student, 1996-1998 | SGI |
| Horner, Jeff | Undergrad, Research Associate, 1995-1999 | |
| Jacobs, Paul | Undergraduate Student, 1992-1995 | |
| Jiang, Weicheng | Post Doc, 1992-1995 | |
| Jin, Song | Graduate Student, 1998 | |
| Kim, MyungHo | Visiting Scholar, 1998 | |
| Kolatis, Michael | GRA, Research Associate, 1993-1996 | Veterinarian- Rock Hill, SC |
| Letsche, Todd | Grad Student, 1993-1994 | |
| Lewis, Sharon | GRA, Manager, 1992-1995 | |
| Manchek, Robert | Research Associate 1990-1996 | SANgate Systems, Inc. |
| McMahan, Paul | GRA, Program Director, 1994-2000 | IBM |
| Newton, Peter | Post Doc, 1994-1995 | |
| Papadopoulos, Caroline | Grad Student, 1997-1998 | University of California San Diego |
| Pozo, Roldan | Post Doc, 1992-1994 | NIST |
| Robert, Yves | Visiting Professor, 1996-1997 | ÉNS - Lyon |
| Sidani, Majed | GRA/Post Doc, 1990-1992 | Merrill Lynch, CICG Development |
| Singhal, Shilpa | Undergraduate Student, 1996-1998 | |
| Swany, Martin | Research Associate, 1996-1999 | PhD Program, University of Tennessee |
| Talley, Judi | Sr. Computer Systems Specialist, 1993-1999 | |
| Thurman, John | Grad Student, 1998-1999 | |
| Tisseur, Françoise | Post Doc, 1997 | University of Manchester |
| Tourancheau, Bernard | Post Doc, 1993-1994 | Université Claude Bernard de Lyon |
| van de Geijn, Robert | Post Doc, 1990-1991 | University of Texas at Austin |
| Wade, Reed | Research Associate, 1990-1996 | WebMD |

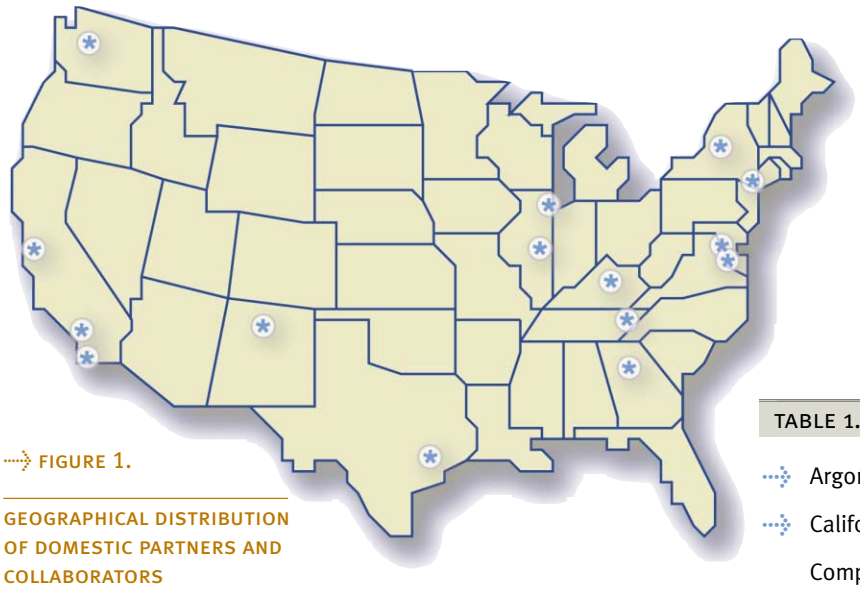


FIGURE 1.

GEOGRAPHICAL DISTRIBUTION OF DOMESTIC PARTNERS AND COLLABORATORS

Over the years, ICL has enjoyed many mutually beneficial working relationships with institutions all over the globe. The high performance computing (HPC) community consists of academic institutions, research centers, branches of the federal government, and various other public and private organizations. As a research group and a member of this community, ICL shares many of the same interests with numerous other institutions. Much of our growth has been in large part due to these relationships. Our ability to collaborate with such institutions has strengthened our research efforts by allowing us to share resources, both material and intellectual. Table 1 highlights many of our domestic partners and collaborators. In addition to our many US government and academic partners, we have also enjoyed a strong working relationship with many commercial software vendors as well as many international HPC research centers and organizations. Included in the list of software vendor collaborators are Etnus, Inc., developer of the TotalView debugger; Kuck and Associates, Inc., developer of the KAP/Pro toolset; and Pallas, developer of the Vampir performance visualization and analysis tool.

Figure 2 shows the geographical location of many of the international partners and collaborators in research with whom we continue to work.

TABLE 1. DOMESTIC PARTNERS AND COLLABORATORS

- Argonne National Laboratory
- California Institute of Technology Center for Advanced Computing Research (CACR)
- Defense Advanced Research Projects Agency (DARPA)
- Department of Defense (DoD)
- Department of Energy (DOE)
- Emory University
- Information Sciences Institute (ISI)
- International Business Machines (IBM)
- Joint Institute for Computational Science (JICS)
- Lawrence Livermore National Laboratory (LLNL)
- Los Alamos National Laboratory (LANL)
- Microsoft Research
- National Aeronautics and Space Administration (NASA)
- National Computational Science Alliance (NCSA)
- National HPCC Software Exchange (NHSE)
- National Institute of Standards and Technology (NIST)
- National Partnership for Advanced Computational Infrastructure (NPACI)
- National Science Foundation (NSF)
- Northeast Parallel Architectures Center (NPAC)
- Oak Ridge National Laboratory (ORNL)
- Rice University
- San Diego Supercomputing Center (SDSC)
- Silicon Graphics Incorporated (SGI)
- Sun Microsystems
- University of California, Berkeley
- University of California, San Diego
- University of Kentucky

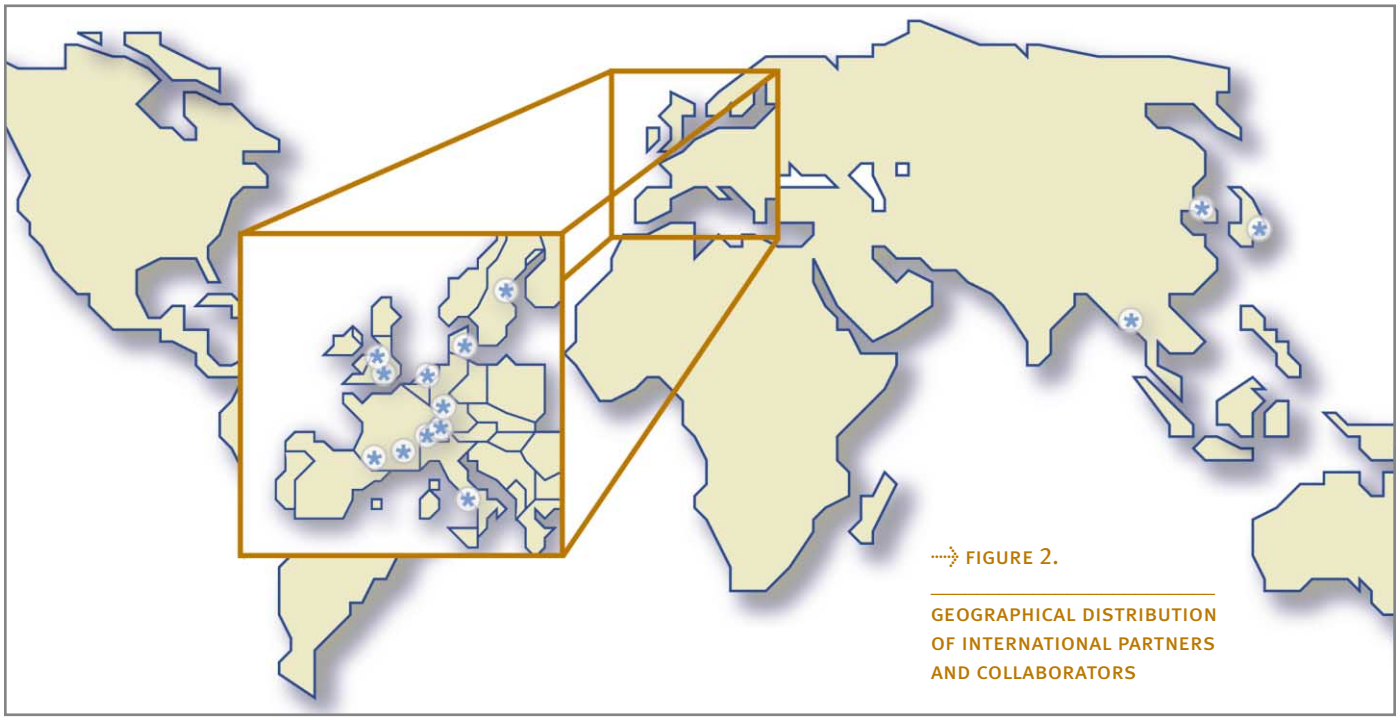


FIGURE 2.

GEOGRAPHICAL DISTRIBUTION OF INTERNATIONAL PARTNERS AND COLLABORATORS

TABLE 2. INTERNATIONAL PARTNERS AND COLLABORATORS

- Danish Computing Centre for Research and Education - Lyngby, Denmark - http://lawra.uni-c.dk/
- Department of Mathematical and Computing Sciences Tokyo Institute of Technology - Tokyo, Japan - http://matsu-www.is.titech.ac.jp/
- Department of Mathematics, University of Manchester - Manchester, England - http://www.maths.man.ac.uk/
- Electrotechnical Laboratory, Computer Systems Division - Tsukuba, Japan - http://phase.etl.go.jp/
- European Centre for Research and Advanced Training in Scientific Computing (CERFACS) - Toulouse, France - http://www.cerfacs.fr/
- Fakultät für Mathematik und Informatik, Universität Mannheim - Mannheim, Germany - http://www.uni-mannheim.de/
- Institut für Wissenschaftliches Rechnen, ETH Zentrum - Zürich, Switzerland - http://www.inf.ethz.ch/
- Istituto per le Applicazioni del Calcolo "Mauro Picone" del C.N.R. - Rome, Italy - http://www.iac.rm.cnr.it/
- Kasetsart University - Bangkok, Thailand - http://smile.cpe.ku.ac.th/
- Laboratoire de l'Informatique du Parallelisme, École Normale Superieure de Lyon - Lyon, France - http://www.ens-lyon.fr/
- Laboratoire Réseaux Haut Débits et Support d'Applications Multimedia (RESAM) Jeune Equipe de l'Université Claude Bernard de Lyon- Lyon, France - http://lhpca.univ-lyon1.fr/
- Mathematical Institute, Utrecht University - Netherlands - http://www.math.uu.nl/
- The Numerical Algorithms Group Ltd. - Oxford, England - http://www.nag.co.uk/
- Rutherford Appleton Laboratory - Oxford, England - http://www.rl.ac.uk/
- Scole Polytechnique Federale de Lausanne - Lausanne, Switzerland - http://capawww.epfl.ch/
- Soongsil University - Seoul, South Korea - http://www.soongsil.ac.kr/english/
- University of Umeå - Umeå, Sweden - http://www.umu.se/umu/index_eng.html

# 1999-2000 PUBLICATIONS

## 2000


Arnold, D., Bachmann, D., Dongarra, J. "Request Sequencing: Optimizing Communication for the Grid," *Lecture Notes in Computer Science: Proceedings of 6th International Euro-Par Conference 2000, Parallel Processing*, (Germany: Springer Verlag 2000), Vol. 1900, 1213-1222.

Arnold, D., Blackford, S., Dongarra, J., Eijkhout, V., Xu, T. "Seamless Access to Adaptive Solver Algorithms," *Proceedings of 16th IMACS World Congress 2000 on Scientific Computing, Applications Mathematics and Simulation*, Lausanne, Switzerland, August 22, 2000.

Arnold, D., Browne, S., Dongarra, J., Fagg, G., Moore, K. "Secure Remote Access to Numerical Software and Computational Hardware," *Proceedings of the DoD HPC Users Group Conference (HPCUG) 2000*, Albuquerque, NM, June 7, 2000.

Arnold, D., Dongarra, J. "Developing an Architecture to Support the Implementation and Development of Scientific Computing Applications," to appear in *Proceedings of Working Conference 8: Software Architecture for Scientific Computing Applications*, Ottawa, Canada, October 2-6, 2000.

Arnold, D., Lee, W., Dongarra, J., Wheeler, M. "Providing Infrastructure and Interface to High-Performance Applications in a Distributed Setting," *High Performance Computing*, 2000.

Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., Johnsson, L., Kennedy, K., Kesselman, C., Reed, D., Torczon, L., Wolski, R. "The GrADS Project: Software Support for High-Level Grid Application Development," *Technical Report*, February 2000.

Browne, S., Dongarra, J., Garner, N., Ho, G., Mucci, P. "A Portable Programming Interface for Performance Evaluation on Modern Processors," *The International Journal of High Performance Computing Applications*, Vol. 14, Number 3 (Fall 2000): 189-204.

Casanova, H., Plank, J., Beck, M., Dongarra, J. "Deploying Fault-tolerance and Task Migration with NetSolve," To appear in The *International Journal on Future Generation Computer Systems*.

Cronk, D., Ellis, B., Fagg, G. "Metacomputing: An Evaluation of Emerging Systems," University of Tennessee Computer Science Department *Technical Report*, July 2000. UT-CS-00-445.

D'Azevedo, E., Dongarra, J. "The Design and Implementation of the Parallel Out of Core ScaLAPACK LU, QR, and Cholesky Factorization Routines," to appear in *Concurrency: Practice and Experience*.

Dongarra, J. "Performance of Various Computers Using Standard Linear Equations Software (Linpack Benchmark Report)," University of Tennessee Computer Science Department *Technical Report*. CS-89-85.

Dongarra, J., Fagg, G., Hempel, R., Walker, D. "Message Passing Software Systems," Webster, J. ed., to appear in *Encyclopedia of Electrical and Electronics Engineering* (New York: Wiley and Sons), 2000.

Dongarra, J., Kacsuk, P., Podhorszki, N. (Eds.) "Recent Advances in Parallel Virtual Machine and Message Passing Interface," *Lecture Notes in Computer Science: Proceedings of 7th European PVM/MPI Users' Group Meeting 2000*, (Hungary: Springer Verlag, 2000), Vol. 1908.

Dongarra, J., Meuer, H., Strohmaier, E. "Top500 Supercomputer Sites (15th edition)," University of Tennessee Computer Science Department *Technical Report*, June 1999. UT-CS-00-442.

Dongarra, J., Raghavan, P. "A New Recursive Implementation of Sparse Cholesky Factorization," *Proceedings of 16th IMACS World Congress 2000 on Scientific Computing, Applications Mathematics*

*and Simulation*, Lausanne, Switzerland, August 22, 2000.

Fagg, G., Dongarra, J. "FT-MPI: Fault Tolerant MPI, Supporting Dynamic Applications in a Dynamic World," *Lecture Notes in Computer Science: Proceedings of EuroPVM-MPI 2000*, (Hungary: Springer Verlag, 2000), Vol. 1908, 346-353.

Henry, G., Watkins, D., Dongarra, J. "A Parallel Implementation of the Nonsymmetric QR Algorithm for Distributed Memory Architectures," to appear in *SIAM Journal on Scientific Computing*.

Kennedy, K., Broom, B., Cooper, K., Dongarra, J., Fowler, R., Gannon, D, Johnsson, L., Mellor-Crummey, J., Torczon, L. "Telescoping Languages: A Strategy for Automatic Generation of Scientific Problem-Solving Systems from Annotated Libraries," to appear in *Journal of Parallel and Distributed Computing*.

Millar, J., McMahan, P., Dongarra, J. "RIBAPI - Repository in a Box Application Programmer's Interface," University of Tennessee Computer Science *Technical Report*, January 2000. UT-CS-00-438.

Raman, G., Dongarra, J. "Design and Implementation of NetSolve using DCOM as the Remoting Layer," University of Tennessee Computer Science Department *Technical Report*, May 2000. UT-CS-00-440.

Vadhiyar, S., Fagg, G., Dongarra, J. "Automatically Tuned Collective Communications," to appear in *Proceedings of SC2000* (Dallas, November 2000).

Whaley, C., Petitet, A., Dongarra, J. "Automated Empirical Optimizations of Software and the ATLAS Project," to appear in *Parallel Computing*. Also a University of Tennessee Computer Science Department *Technical Report*. UT-CS-00-448.

## 1999

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D., *LAPACK Users' Guide*, 3rd ed., Philadelphia: Society for Industrial and Applied Mathematics, 1999.

Arbenz, P., Cleary, A., Dongarra, J., Hegland, M. "A Comparison of Parallel Solvers for Diagonally Dominant and General Narrow Banded Linear Systems II," University of Tennessee Computer Science Department *Technical Report*. UT-CS-99-415.

Arbenz, P., Cleary, A., Dongarra, J., Hegland, M. "A Comparison of Parallel Solvers for General Narrow Banded Linear Systems," University of Tennessee Computer Science Department *Technical Report*. UT-CS-99-214. Also to appear in *Parallel and Distributed Computing Practices*.

Beck, M., Casanova, H., Dongarra, J., Moore, T., Plank, J., Berman, F., Wolski, R. "Logistical Quality of Service in NetSolve," *Computer Communications*, Vol. 22, Number 11 (1999): 1034-1044.

Beck, M., Chawla, R., Dempsey, B., Moore, T. "Portable Representation of Internet Content Channels in I2-DSI," *4th Intl. Web Caching Workshop*, San Diego, March 31-April 2, 1999.

Beck, M., Dongarra, J., Fagg, G., Geist, A., Gray, P., Kohl, J., Migliardi, M., Moore, K., Moore, T., Papadopoulous, P., Scott, S., Sunderam, V. "HARNESS: A Next Generation Distributed Virtual Machine," *International Journal on Future Generation Computer Systems*, Vol. 15, Number 5/6 (1999).

Berry, M., Dongarra, J. "Atlanta Organizers Put Mathematics to Work For the Math Sciences Community," *SIAM News*, Vol. 32, Number 6.

Boulet, P., Dongarra, J., Rastello, F., Robert, Y., Vivien, F. "Algorithmic Issues on Heterogeneous Computing Platforms," *Parallel Processing Letters*, Vol. 9, Number 2 (1999): 197-213.

Boulet, P., Dongarra, J., Robert, Y., Vivien, F. "Static Tiling for Heterogeneous Computing Platforms," *Parallel Computing*, Vol. 25, Number 5 (1999): 547-568.

Browne, S., Deane, C., Ho, G., Mucci, P. "PAPI: A Portable Interface to Hardware Performance Counters," *Proceedings of Department of Defense HPCMP Users Group Conference*, June 1999.

Browne, S., Dongarra, J., Trefethen, A. "Numerical Libraries and Tools for Scalable Parallel Cluster Computing," *IEEE Cluster Computing BOF at SC99*, Portland, Oregon. November 1999.

Browne, S., McMahan, P., Wells, S. "Repository in a Box Toolkit for Software and Resource Sharing," University of Tennessee Computer Science Department *Technical Report*, May 1999. UT-CS-99-424.

Calland, P., Dongarra, J., Robert, Y. "Tiling on Systems with Communication/Computation Overlap," *Concurrency: Practice and Experience*, Vol. 11, Number 3 (1999): 139-153.

Casanova, H., Kim, M., Plank, J., Dongarra, J. "Adaptive Scheduling for Task Farming with Grid Middleware," *International Journal of Supercomputer Applications and High Performance Computing*, Vol. 13, Number 3 (Fall 1999): 231-240.

Casanova, H., Thomason, M., Dongarra, J. "Stochastic Performance Prediction for Iterative Algorithms in Distributed Environments," *Journal of Parallel and Distributed Computing*, Vol. 98, Number 1 (1999): 68-91.

Dempsey, B., Weiss, D. "Towards An Efficient, Scalable Replication Mechanism for the I2-DSI Project," University of North Carolina School of Library and Information Science *Technical Report*, April 1999. TR-1999-01.

Dongarra, J., Eijkhout, V. "Numerical Linear Algebra Algorithms and Software," *Journal CAM (Numerical) Linear Algebra*. Vol. 31, Number 4 (October 1999).

Dongarra, J., Eijkhout, V., Dekker, M. Publisher "Numerical Linear Algebra," in *Encyclopedia of Computer Science and Technology*, eds. Kent, A., Williams, J., Vol. 41, (August 1999): 207-233.

Dongarra, J., Meuer, H., Strohmaier, E. "Top500 Supercomputer Sites (13th edition)," University of Tennessee Computer Science Department *Technical Report*, June 1999. UT-CS-99-425.

Dongarra, J., Meuer, H., Strohmaier, E. "Top500 Supercomputer Sites (14th edition)," University of Tennessee Computer Science Department *Technical Report*, November 1999. UT-CS-99-434.

Doolin, D., Dongarra, J., Seymour, K. "JLAPACK - Compiling LAPACK Fortran to Java," *Scientific Programming*, Vol. 7 Number 2 (1999): 111-138.

Eijkhout, V. "On the Existence Problem of Incomplete Factorisation Methods," University of Tennessee Computer Science Department *Technical Report*, December 1999. UT-CS-99-435.

Eijkhout, V. "The 'Weighted Modification' Incomplete Factorisation Method," University of Tennessee Computer Science Department *Technical Report*, December 1999. UT-CS-99-436.

Elwasif, W., Beck, M., Plank, J. "IBP-MIME: Controlled Delivery of Large Mail Files," University of Tennessee Computer Science Department *Technical Report*, April 1999. UT-CS-99-421.

Elwasif, W., Plank, J., Beck, M. "IBP - Internet Backplane Protocol: Infrastructure for Distributed Storage (V 0.2)," University of Tennessee Computer Science Department *Technical Report*, February 1999. UT-CS-99-430.

Fagg, G., Moore, K., Dongarra, J. "Scalable Networked Information Processing Environment (SNIPE)," *International Journal on Future Generation Computer Systems*, Vol. 15, Number 5/6 (1999): 595-605.

Fischer, M., Dongarra, J. "Experiences with Windows 95/NT as a Cluster Computing Platform for Parallel Computing," *Parallel and Distributed Computing Practices*, February 1999 (Vol. 2, No. 2), Nova Science Publishers, USA.

Petitet, A., Casanova, H., Dongarra, J., Robert, Y., Whaley, C. "Parallel and Distributed Scientific Computing: A Numerical Linear Algebra Problem Solving Environment Designer's Perspective," *Handbook on Parallel and Distributed Processing*, 1999.

Petitet, A., Casanova, H., Whaley, C., Dongarra, J., Robert, Y. "A Numerical Linear Algebra Problem Solving Environment Designer's Perspective," *SIAM Annual Meeting*, Atlanta, GA. May 13, 1999.

Plank, J., Beck, M., Elwasif, W., Moore, T., Swany, M., Wolski, R. "The Internet Backplane Protocol: Storage in the Network," *Proceedings of Network Storage Symposium*, October, 1999.

Strohmaier, E., Dongarra, J., Meuer, H., Simon, H. "The Marketplace of HPC," *Parallel Computing*, 25th anniversary ed., Vol. 25 (1999): 1517-1544.

Tisseur, F., Dongarra, J. "Parallelizing the Divide and Conquer Algorithm for the Symmetric Tridiagonal Eigenvalue Problem on Distributed Memory Architectures," *SIAM Journal on Scientific Computing*, Vol. 6, Number 20 (1999): 2223-2236.

# FROM THE DIRECTOR

The Innovative Computing Laboratory (ICL), aspires to be a world leader in enabling technologies and software for scientific computing. Our vision is to provide high performance tools to tackle science's most challenging problems and to play a major role in the development of standards for scientific computing in general. The result will be a rate of scientific progress previously unknown.

In 2000, the ICL is celebrating 11 years of leadership in enabling technologies for high performance computing. Looking back over the 11-year period, the evolution and growth of the technology for computing have been truly astonishing. In 1989, the speed of a supercomputer was measured in gigaflops and in gigabytes. Today, our measures are teraflops for speed and terabytes for memory, a thousand-fold increase over the standards of 10 years ago. The research that ICL has undertaken in the past decade has followed a natural progression and growth from our original thread of numerical linear algebra to performance evaluation, to software repositories, and to distributed computing.

The ICL staff's ongoing ability to apply the latest technologies to provide advanced services and solutions for the scientific computing community underscores the ICL's leadership role. Standards and efforts such as PVM, MPI, LAPACK, ScaLAPACK, BLAS, Netlib, NHSE, TOP500, and the LINPACK Benchmark have all left their mark on the scientific community. We are continuing these efforts with ATLAS, PAPI, NetSolve, RIB, and the Top100 Clusters, as well as other innovative computing efforts.

One of the highlights for this past year has been the kick-off of the SInRG efforts, which will result in the use and integration of many ICL technologies for applications in an experimental Grid for computational scientists. The pace of integration will continue to accelerate in the coming years.

During these exciting times, I am grateful to our sponsors for their continued endorsement of our efforts. My special thanks and congratulations go to the ICL staff and students for their skill, dedication, and tireless efforts in making the ICL one of the best centers for enabling technology in the world.

-Jack Dongarra

# CONTACT INFORMATION


Web Site
http://icl.cs.utk.edu/

E-mail
icl@cs.utk.edu

Telephone
(865) 974-8295

Fax
(865) 974-8296

Address
Innovative Computing Laboratory (ICL)
Suite 203
1122 Volunteer Blvd.
Knoxville, TN 37996-3450