

CCGrid2001

IEEE International Symposium on Cluster Computing and the Grid

Three Tools to Help with Cluster and Grid Computing: ATLAS, PAPI, and NetSolve

Jack Dongarra
Innovative Computing Laboratory
University of Tennessee

<http://www.cs.utk.edu/~dongarra/>

1

Overview

- ♦ High Performance Computing
- ♦ ATLAS
- ♦ PAPI
- ♦ NetSolve

2

High Performance Computers Sustainable Performance

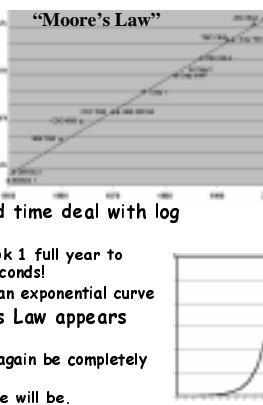
- ♦ ~ 20 years ago
 - 1×10^6 Floating Point Ops/sec (Mflop/s)
 - Scalar based
- ♦ ~ 10 years ago
 - 1×10^9 Floating Point Ops/sec (Gflop/s)
 - Vector & Shared memory computing, bandwidth aware
 - Block partitioned, latency tolerant
- ♦ ~ Today
 - 1×10^{12} Floating Point Ops/sec (Tflop/s)
 - Highly parallel, distributed processing, message passing, network based
 - data decomposition, communication/computation
- ♦ ~ 10 years away
 - 1×10^{15} Floating Point Ops/sec (Pflop/s)
 - Many more levels MH, combination/grids&HPC
 - More adaptive, LT and bandwidth aware, fault tolerant, extended precision, attention to SMP nodes

3

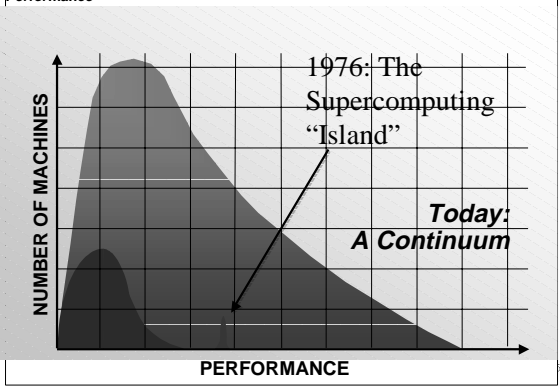
“Moore’s Wall”

— Horst Simon, NERSC

- ♦ Moore’s Law predicts exponential growth
 - Performance doubling every 18 months
 - Usually plotted on semi-log scale, appears as straight line
- ♦ Human experience has a hard time deal with log scale
 - In 1980 a computation that took 1 full year to complete can now be done in seconds!
 - We are sitting at the bend of an exponential curve
- ♦ From our perspective Moore’s Law appears as a “wall”
 - In a few years technology will again be completely different
 - Hard to predict what the future will be.



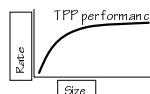
Performance



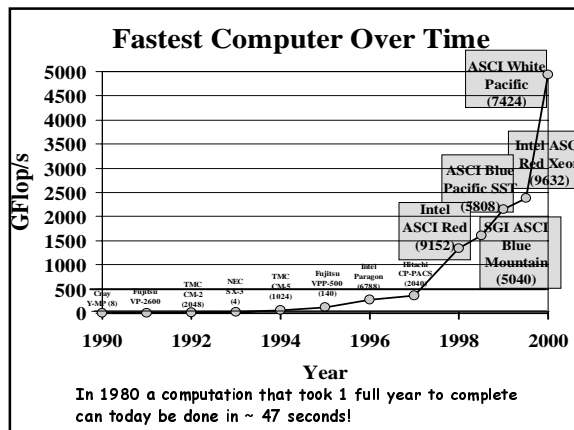
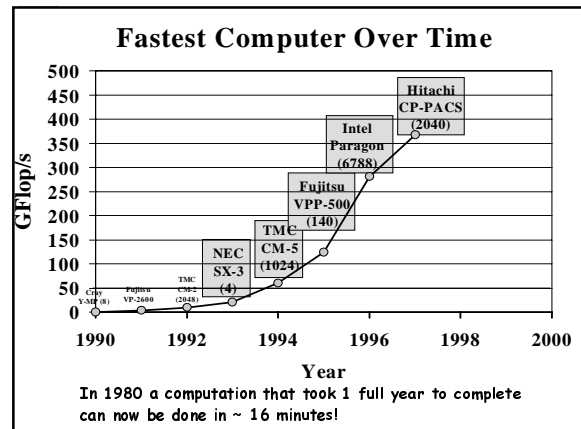
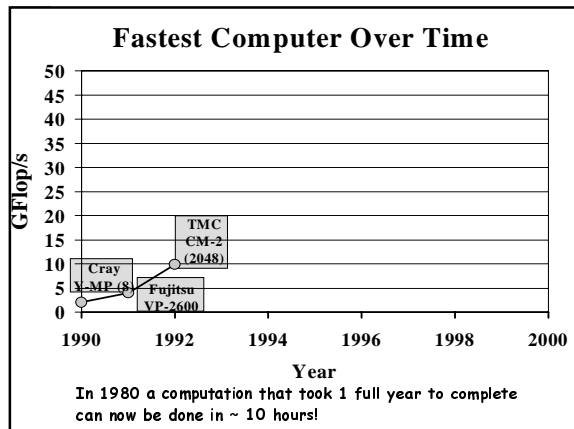
TOP500

- Listing of the 500 most powerful Computers in the World
- Yardstick: Rmax from LINPACK MPP

$$Ax=b, \text{ dense problem}$$
- Updated twice a year
 - SC’xy in the States in November
 - Meeting in Mannheim, Germany in June
- All data available from www.top500.org

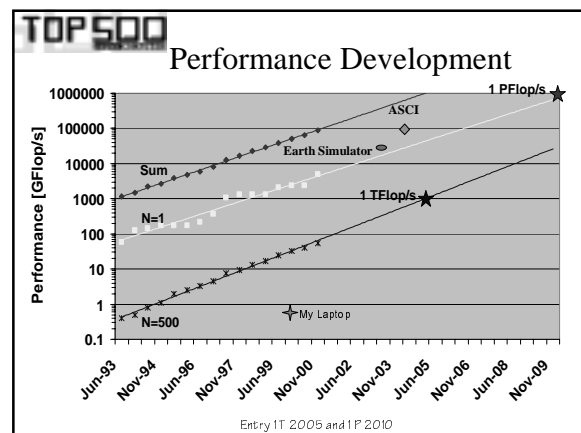
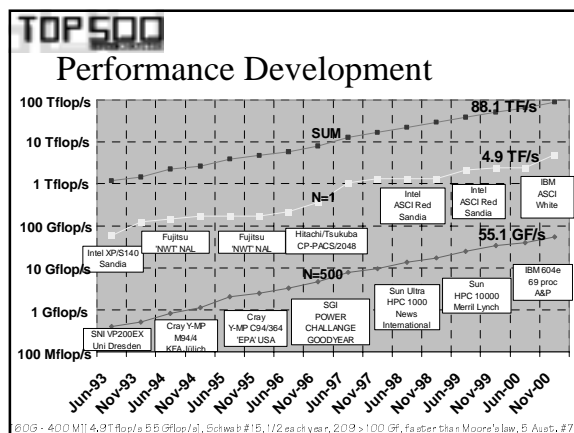


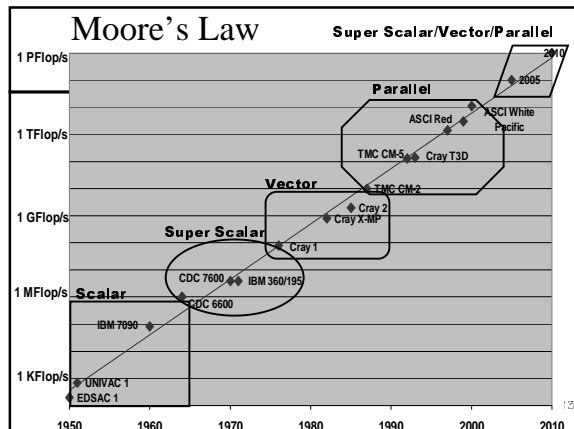
4



TOP500 Top 10 Machines (Nov 2000)

Rank	Company	Machine	Procs	GFlop/s	Place	Country	Year
1	IBM	ASCI White	8192	4938	Livermore National Laboratory	Livermore	2000
2	Intel	ASCI Red	9632	2380	Sandia National Labs	Albuquerque	1999
3	IBM	ASCI Blue-Pacific SST, IBM SP 604e	5808	2144	Lawrence Livermore National Laboratory	Livermore	1999
4	SGI	ASCI Blue Mountain	6144	1608	Los Alamos National Laboratory	Los Alamos	1998
5	IBM	SP Power3 375 MHz	1336	1417	Naval Oceanographic Office (NAVOCEANO)		2000
6	IBM	SP Power3 375 MHz	1104	1179	National Center for Environmental Protection		2000
7	Hitachi	SR8000-F1/112	112	1035	Leibniz Rechenzentrum	Muenchen	2000
8	IBM	SP Power3 375 MHz, 8 way	1152	929	UCSD/San Diego Supercomputer Center		2000
9	Hitachi	SR8000-F1/100	100	917	High Energy Accelerator Research Organization/KEK	Tsukuba	2000
10	Cray Inc.	T3E1200	1084	892	Government		1998





Petaflop Computers Within the Next Decade

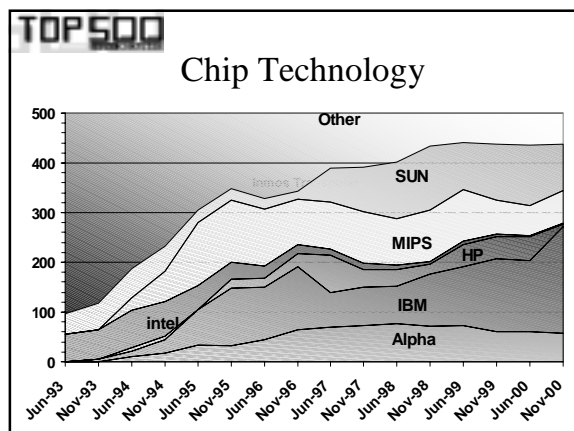
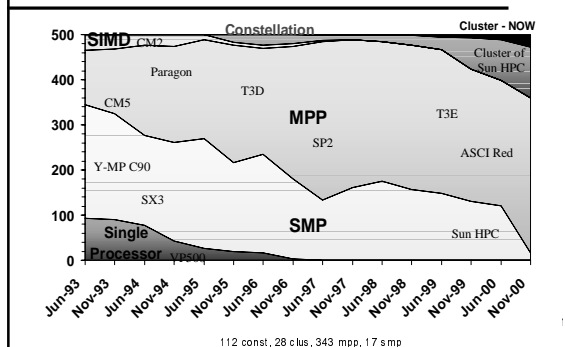
- ♦ Five basis design points:
 - Conventional technologies
 - 4.8 GHz processor, 8000 nodes, each w/16 processors
 - Processing-in-memory (PIM) designs
 - Reduce memory access bottleneck
 - Superconducting processor technologies
 - Digital superconductor technology, Rapid Single-Flux-Quantum (RSFQ) logic & hybrid technology multi-threaded (HTMT)
 - Special-purpose hardware designs
 - Specific applications e.g. GRAPE Project in Japan for gravitational force computations
 - Schemes utilizing the aggregate computing power of processors distributed on the web
 - SETI@home ~26 Tflop/s

Petaflops (10^{15} flop/s) Computer Today?

1 GHz processor ($O(10^9)$ ops/s)

- 1 Million PCs
- \$1B (\$1K each)
- 100 Mwatts
- 5 acres
- 1 Million Windows licenses!!
- PC failure every second

Top 500 Architectures



High-Performance Computing Directions: Beowulf-class PC Clusters

Definition:

♦ COTS PC Nodes

- Pentium, Alpha, PowerPC, SMP

♦ COTS LAN/SAN Interconnect

- Ethernet, Myrinet, Gigaset, ATM

♦ Open Source Unix

- Linux, BSD

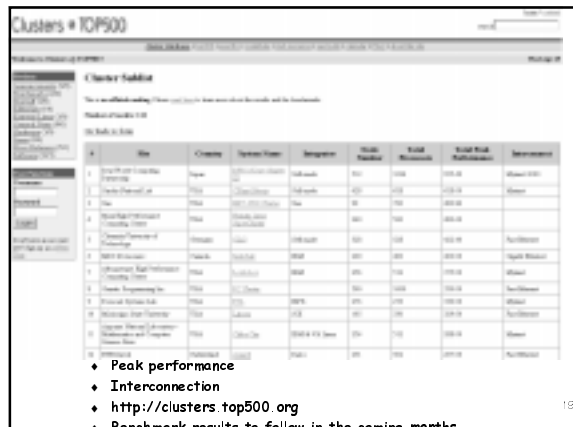
♦ Message Passing Computing

- MPI, PVM
- HPF

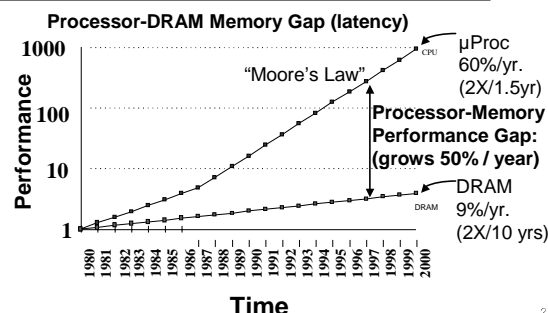
Advantages:

- ♦ Best price-performance
- ♦ Low entry-level cost
- ♦ Just-in-place configuration
- ♦ Vendor invulnerable
- ♦ Scalable
- ♦ Rapid technology tracking

Enabled by PC hardware, networks and operating system achieving capabilities of scientific workstations at a fraction of the cost and availability of industry standard message passing libraries. However, much more of a contact sport.



Where Does the Performance Go? or Why Should I Care About the Memory Hierarchy?



Optimizing Computation and Memory Use

Computational optimizations

- Theoretical peak: $(\# \text{ fpus}) * (\text{flops/cycle}) * \text{Mhz}$
 - PIII: $(1 \text{ fpu}) * (1 \text{ flop/cycle}) * (850 \text{ Mhz}) = 850 \text{ MFLOP/s}$
 - Athlon: $(2 \text{ fpu}) * (1 \text{ flop/cycle}) * (600 \text{ Mhz}) = 1200 \text{ MFLOP/s}$
 - Power3: $(2 \text{ fpu}) * (2 \text{ flops/cycle}) * (375 \text{ Mhz}) = 1500 \text{ MFLOP/s}$

Operations like:

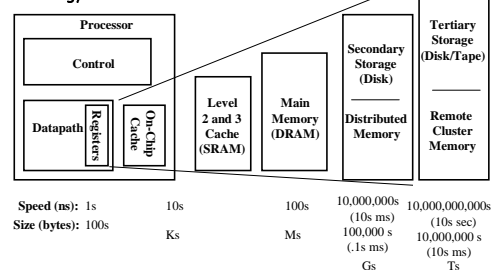
- $\alpha = x^T y$: 2 operands (16 Bytes) needed for 2 flops; at 850 Mflop/s will require 1700 MW/s bandwidth
- $y = \alpha x + y$: 3 operands (24 Bytes) needed for 2 flops; at 850 Mflop/s will require 2550 MW/s bandwidth

Memory optimization

- Theoretical peak: $(\text{bus width}) * (\text{bus speed})$
 - PIII: $(32 \text{ bits}) * (133 \text{ Mhz}) = 532 \text{ MB/s} = 66.5 \text{ MW/s}$
 - Athlon: $(64 \text{ bits}) * (133 \text{ Mhz}) = 1064 \text{ MB/s} = 133 \text{ MW/s}$
 - Power3: $(128 \text{ bits}) * (100 \text{ Mhz}) = 1600 \text{ MB/s} = 200 \text{ MW/s}$

Memory Hierarchy

- By taking advantage of the principle of locality:
 - Present the user with as much memory as is available in the cheapest technology.
 - Provide access at the speed offered by the fastest technology.



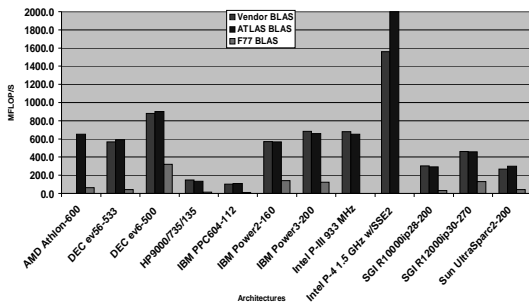
Self-Adapting Numerical Software (SANS)

- Today's processors can achieve high-performance, but this requires extensive machine-specific hand tuning.
- Operations like the BLAS require many man-hours / platform
 - Software lags far behind hardware introduction
 - Only done if financial incentive is there
- Hardware, compilers, and software have a large design space w/many parameters
 - Blocking sizes, loop nesting permutations, loop unrolling depths, software pipelining strategies, register allocations, and instruction schedules.
 - Complicated interactions with the increasingly sophisticated micro-architectures of new microprocessors.
- Need for quick/dynamic deployment of optimized routines.
- ATLAS - Automatic Tuned Linear Algebra Software

Software Generation Strategy - BLAS

- Parameter study of the hw
- Generate multiple versions of code, w/difference values of key performance parameters
- Run and measure the performance for various versions
- Pick best and generate library
- Level 1 cache multiply optimizes for:
 - TLB access
 - L1 cache reuse
 - FP unit usage
 - Memory fetch
 - Register reuse
 - Loop overhead minimization
- Takes ~ 20 minutes to run.
- "New" model of high performance programming where critical code is machine generated using parameter optimization.
- Designed for RISC arch
 - Super Scalar
 - Need reasonable C compiler
- Today ATLAS in use by Matlab, Mathematica, Octave, Maple, Debian, Scyld Beowulf, SuSE, ...

ATLAS (DGEMM $n = 500$)

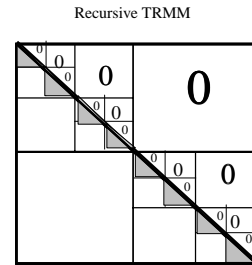


- ◆ ATLAS is faster than all other portable BLAS implementations and it is comparable with machine-specific libraries provided by the vendor.

Recursive Approach for Other Level 3 BLAS

- ◆ Recur down to L1 cache block size
- ◆ Need kernel at bottom of recursion

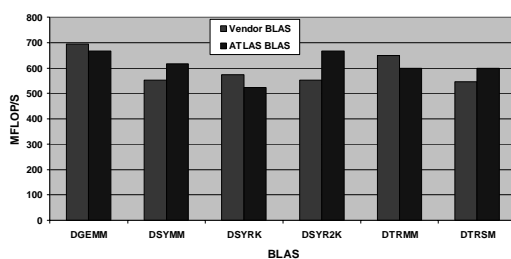
- Use gemm-based kernel for portability



26

Intel PIII 933 MHz

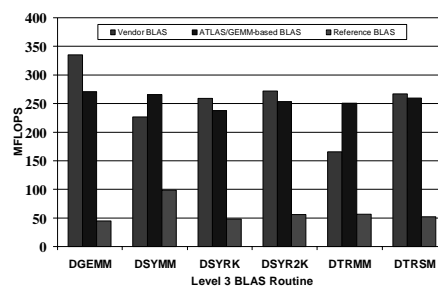
MKL 5.0 vs ATLAS 3.2.0 using Windows 2000



- ◆ ATLAS is faster than all other portable BLAS implementations and it is comparable with machine-specific libraries provided by the vendor.

27

500x500 Recursive BLAS on UltraSparc 2200



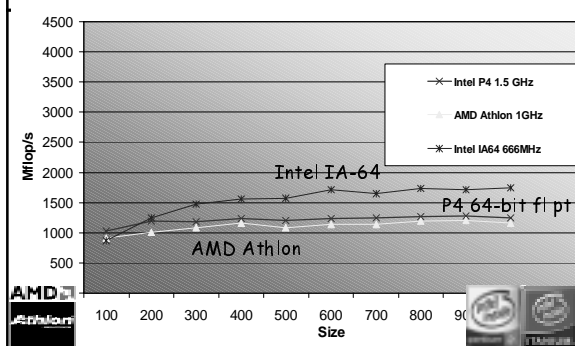
28

Related Tuning Projects

- ◆ PHiPAC
 - Portable High Performance ANSI C
 - www.lcs.berkeley.edu/~bilmes/phipac initial automatic GEMM generation project
- ◆ FFTW Fastest Fourier Transform in the West
 - www.fftw.org
- ◆ UHFFT
 - tuning parallel FFT algorithms
 - rodin.cs.uh.edu/~mirkovic/fft/parfft.htm
- ◆ SPIRAL
 - Signal Processing Algorithms Implementation Research for Adaptable Libraries maps DSP algorithms to architectures
- ◆ Sparsity
 - Sparse-matrix-vector and Sparse-matrix-matrix multiplication
 - www.cs.berkeley.edu/~ejim/publication/ tunes code to sparsity structure of matrix more later in this tutorial
 - University of Tennessee

29

ATLAS Matrix Multiply (64-bit floating point results)



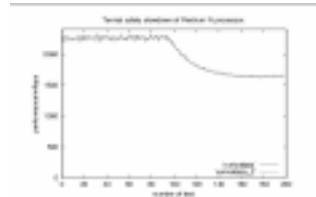
Pentium 4 - SSE2

- ♦ 1.5 GHz, 400 MHz system bus, 16K L1 & 256K L2 Cache, theoretical peak of 1.5 Gflop/s, high power consumption
- ♦ Streaming SIMD Extensions 2 (SSE2)
 - which consists of 144 new instructions
 - includes SIMD IEEE double precision floating point
 - Peak for 64 bit floating point 2X
 - Peak for 32 bit floating point 4X
 - SIMD 128-bit integer
 - new cache and memory management instructions.
 - Intel's compiler supports these instructions today

31

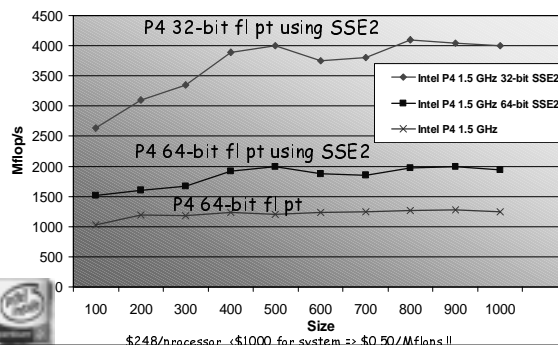
Pentium 4

- ♦ The Pentium 4 consumes a lot of power, 55 - 64 Watts, (33 W for PIII) and is able to generate a lot of heat.
- ♦ Processor may get too hot and if so processor is shutdown for short periods



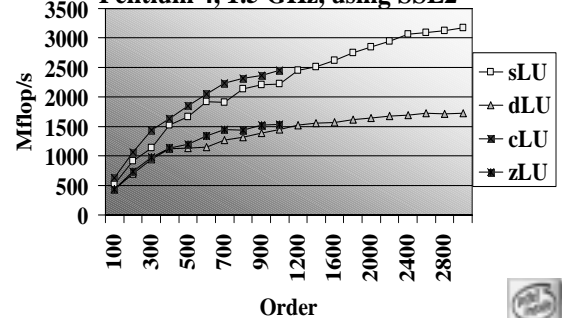
32

ATLAS Matrix Multiply Intel Pentium 4 – using SSE2

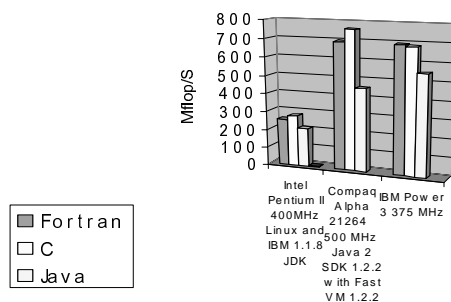


LU Factorization

Pentium 4, 1.5 GHz, using SSE2



Experiments with C, Fortran, and Java for ATLAS (DGEMM kernel)



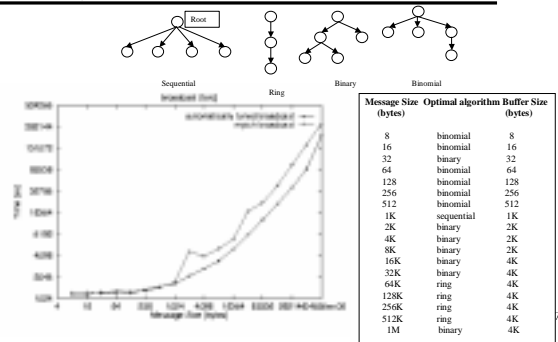
Machine-Assisted Application Development and Adaptation

- ♦ Communication libraries
 - Optimize for the specifics of one's configuration.
- ♦ Algorithm layout and implementation
 - Look at the different ways to express implementation

36

Work in Progress: ATLAS-like Approach Applied to Broadcast

(PII 8 Way Cluster with 100 Mb/s switched network)



Reformulating/Rearranging/Reuse

- ♦ Example is the reduction to narrow band from for the SVD

$$A_{new} = A - uy^T - wv^T$$

$$y_{new} = A^T u$$

$$w_{new} = A_{new} v$$

- ♦ Fetch each entry of A once
- ♦ Restructure and combined operations
- ♦ Results in a speedup of > 30%

3.8

Conjugate Gradient Variants by Dynamic Selection at Run Time

- ♦ Variants combine inner products to reduce communication bottleneck at the expense of more scalar ops.
- ♦ Same number of iterations, no advantage on a sequential processor
- ♦ With a large number of processor and a high-latency network may be advantages.
- ♦ Improvements can range from 15% to 50% depending on size.



4.7

Conjugate Gradient Variants by Dynamic Selection at Run Time

- ♦ Variants combine inner products to reduce communication bottleneck at the expense of more scalar ops.
- ♦ Same number of iterations, no advantage on a sequential processor
- ♦ With a large number of processor and a high-latency network may be advantages.
- ♦ Improvements can range from 15% to 50% depending on size.



4.7

Tools for Performance Evaluation

- ♦ Timing and performance evaluation has been an art
 - Resolution of the clock
 - Issues about cache effects
 - Different systems
 - Can be cumbersome and inefficient with traditional tools
- ♦ Situation about to change
 - Today's processors have internal counters



4.1

Performance Counters

- ♦ Almost all high performance processors include hardware performance counters.
- ♦ Some are easy to access, others not available to users.
- ♦ On most platforms the APIs, if they exist, are not appropriate for the end user or well documented.
- ♦ Existing performance counter APIs
 - Compaq Alpha EV 6 & 6/7
 - SGI MIPS R10000
 - IBM Power Series
 - CRAY T3E
 - Sun Solaris
 - Pentium Linux and Windows
 - IA-64
 - HP-PA RISC
 - Hitachi
 - Fujitsu
 - NEC



Performance Data That May Be Available

- Cycle count
- Floating point instruction count
- Integer instruction count
- Instruction count
- Load/store count
- Branch taken / not taken count
- Branch mispredictions
- Pipeline stalls due to memory subsystem
- Pipeline stalls due to resource conflicts
- I/D cache misses for different levels
- Cache invalidations
- TLB misses
- TLB invalidations

4.3

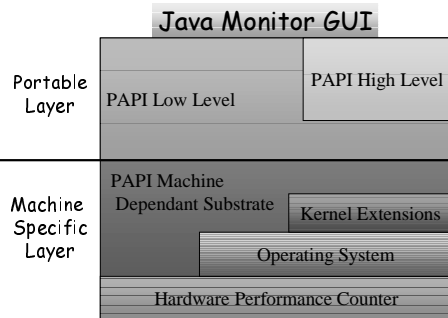
Overview of PAPI

- ◆ Performance Application Programming Interface
- ◆ The purpose of the PAPI project is to design, standardize and implement a portable and efficient API to access the hardware performance monitor counters found on most modern microprocessors



4.4

PAPI Implementation



4.5

PAPI - Supported Processors

- ◆ Pentium, Pro, II, III,
 - Linux 2.4, 2.2, 2.0 and perf kernel patch
- ◆ Power 3, 604, 604e
 - For AIX 4.3 and pmtoolkit (in 4.3.4 available)
 - (laderose@us.ibm.com)
- ◆ UltraSparc I&II (III soon)
 - Solaris 8
- ◆ MIPS R10K, R12K
- ◆ AMD Athlon
 - Linux 2.4 and perf kernel patch
- ◆ Cray T3E, SV1, SV2
- ◆ Soon: Alpha EV6, EV67

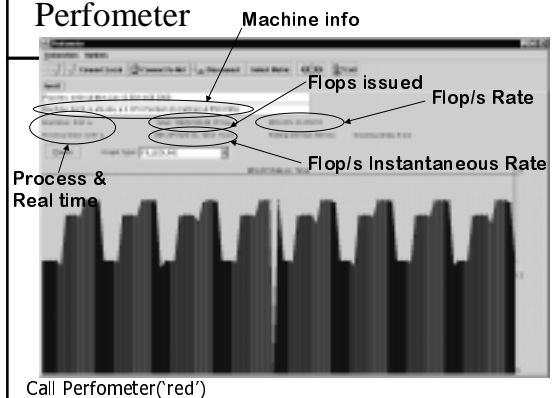
4.6

Graphical Tools Perfometer Usage

- ◆ Application is instrumented with PAPI
 - call `perfometer()`
- ◆ Will be layered over the best existing vendor-specific APIs for these platforms
- ◆ Application is started, at the call to performeter signal handler and timer set to collect and send the information to a Java applet containing the graphical view.
- ◆ Sections of code that are of interest can be designated with specific colors
 - Using a call to `set_perfometer('color')`



Perfometer



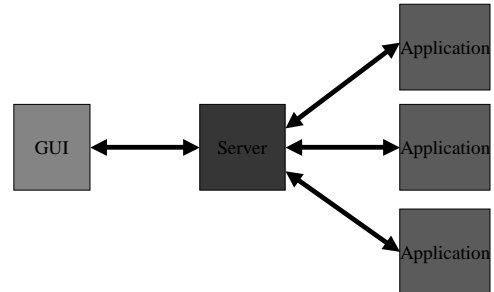
4.8

Go To Demo



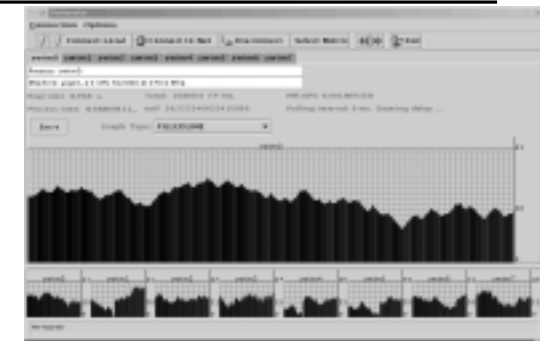
49

Next Version of Perfometer Implementation



50

PAPI's Parallel Interface



51

PAPI Release

♦ Platforms

- Pentium Linux/x86
 - Require patch to kernel
- Sun Solaris/Ultra 2.8
- IBM AIX/Power
 - Contact IBM for PMtoolkit
- SGI IRIX/MIPS
- AMD Athlon Linux
- Compaq Tru64/Alpha (Soon)

♦ Mailing list

- send "subscribe ptools-perfapi" to majordomo@ptools.org
- ptools-perfapi@ptools.org is the reflector



♦ C and Fortran bindings

- ♦ To download software see: <http://icl.cs.utk.edu/papi/>

52

SETI@home

- ♦ Use thousands of Internet-connected PCs to help in the search for extraterrestrial intelligence.
- ♦ Uses data collected with the Arecibo Radio Telescope, in Puerto Rico
- ♦ When their computer is idle or being wasted this software will download a 300 kilobyte chunk of data for analysis.
- ♦ The results of this analysis are sent back to the SETI team, combined with thousands of other participants.



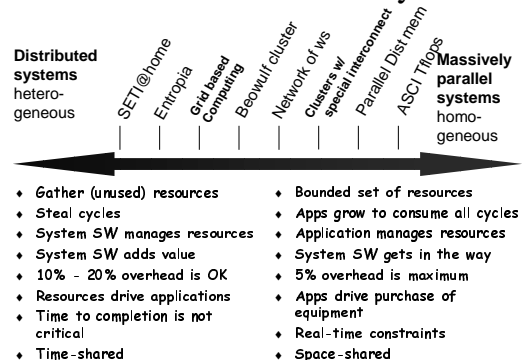
♦ Largest distributed computation project in existence

- ~ 400,000 machines
- Averaging 26 Tflop/s

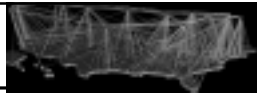
- ♦ Today many companies trying this for profit.

53

Distributed and Parallel Systems



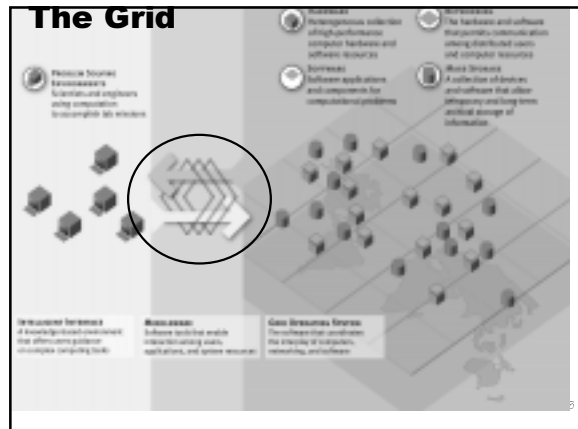
The Grid



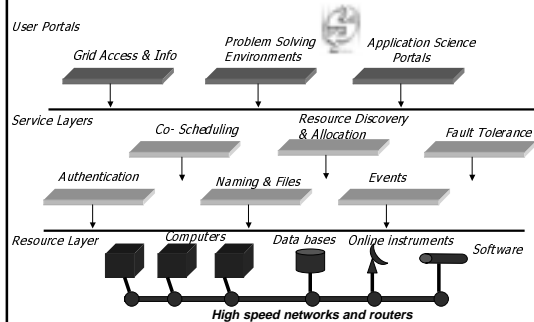
- ♦ To treat CPU cycles and software like commodities.
- ♦ on steroids.
- ♦ Enable the coordinated use of geographically distributed resources - in the absence of central control and existing trust relationships.
- ♦ Computing power is produced much like utilities such as power and water are produced for consumers.
- ♦ Users will have access to "power" on demand

55

The Grid



The Grid Architecture Picture



57

NetSolve Overview

- ♦ Problem Solving Environment Toolkit
- ♦ Client/Agent/Server system.
- ♦ Remote access to hardware AND software.
- ♦ "Robust, fault-tolerant, flexible, heterogeneous environment that provides dynamic management and allocation policies for distributed computational resources."

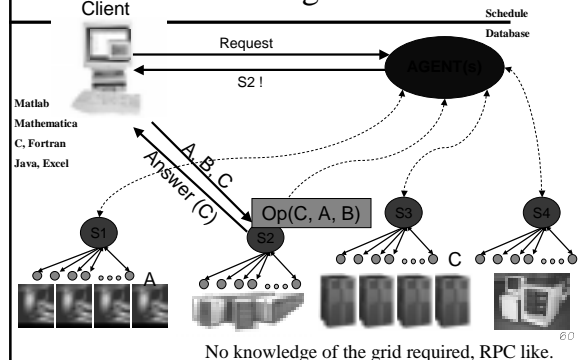
58

NetSolve Network Enabled Server

- ♦ NetSolve is an example of a grid based hardware/software server.
- ♦ Easy-of-use paramount
- ♦ Based on a RPC model but with ...
 - resource discovery, dynamic problem solving capabilities, load balancing, fault tolerance asynchronicity, security, ...
- ♦ Other examples are NEOS from Argonne and NINF Japan.
- ♦ Use a resource, not tie together geographically distributed resources for a single application.

59

NetSolve: The Big Picture



60

NetSolve Agent



- ♦ Name server for the NetSolve system.
- ♦ Information Service
 - client users and administrators can query the hardware and software services available.
- ♦ Resource scheduler
 - maintains both static and dynamic information regarding the NetSolve server components to use for the allocation of resources

61

NetSolve Agent



- ♦ Resource Scheduling (cont'd):
 - CPU Performance (LINPACK).
 - Network bandwidth, latency.
 - Server workload.
 - Problem size/algorithm complexity.
 - Calculates a "Time to Compute." for each appropriate server.
 - Notifies client of most appropriate server.

62

NetSolve Client



- ♦ Function Based Interface.
- ♦ Client program embeds call from NetSolve's API to access additional resources.
- ♦ API available to C, Fortran, Matlab, Mathematica, and Java.
- ♦ Opaque networking interactions.
- ♦ NetSolve can be invoked using a variety of methods: blocking, non-blocking, task farms, ...

63

NetSolve Client



- ♦ Intuitive and easy to use.
- ♦ Matlab Matrix multiply e.g.:
 - `A = matmul(B, C);`

`A = netsolve('matmul', B, C);`

- Possible parallelisms hidden.

64

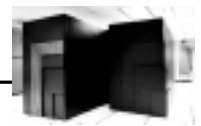
NetSolve Client



- i. Client makes request to agent.
- ii. Agent returns list of servers.
- iii. Client tries each one in turn until one executes successfully or list is exhausted.

65

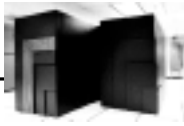
NetSolve Server



- ♦ Computational backbone of NetSolve.
- ♦ Daemon waiting to service client requests.
- ♦ Configured to solve a set of problems.
- ♦ Reports host status to agent periodically (host status includes workload, bandwidth, latency, etc.)

66

NetSolve Server



- ♦ Access control enabled by Kerberos V5.
- ♦ NetSolve is able to appropriately deal with failed servers.
- ♦ Special configurations:
 - Parallel Servers
 - Condor Servers

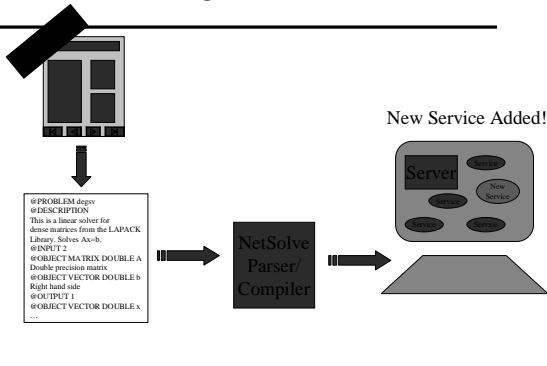
67

Problem Description File

- ♦ Problem Description File defines problem specification used to add functional modules to NetSolve server.
- ♦ Wrapper to provide binding between the NetSolve client interface and server function being integrated.
- ♦ Complex syntax defines input/output objects, calling sequences, libraries to link, etc...
- ♦ Parsed by NetSolve to create "service" program.

68

Generating New Services



69

Basic Usage Scenarios



- ♦ Grid based numerical library routines
 - User doesn't have to have software library on their machine, LAPACK, SuperLU, ScaLAPACK, PETSc, AZTEC, ARPACK
- ♦ Task farming applications
 - "Pleasantly parallel" execution
 - eg Parameter studies
- ♦ Remote application execution
 - Complete applications with user specifying input parameters and receiving output
- ♦ "Blue Collar" Grid Based Computing
 - Does not require deep knowledge of network programming
 - Level of expressiveness right for many users
 - User can set things up, no "su" required
 - In use today, up to 200 servers in 9 countries

70

NetSolve and Numerical Libraries

- ♦ Access to mathematical software without having to install packages
 - User doesn't have to have software library on their machine, LAPACK, ScaLAPACK, PETSc, AZTEC, ARPACK, and others
- ♦ Fortran, C, Matlab, ...

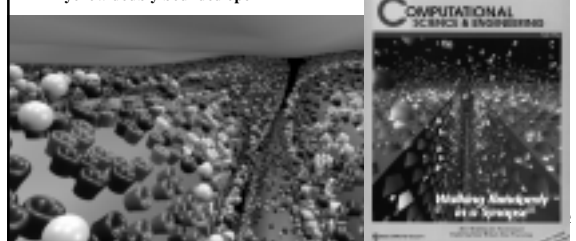
```
>> [x, its] = netsolve( 'f0meth', 'petsc', A, rhs );
```

```
call NETSL( 'DGESV( )', NSINFO,
            N, 1, A, MAX, IPIV, B, MAX, INFO )
```

71

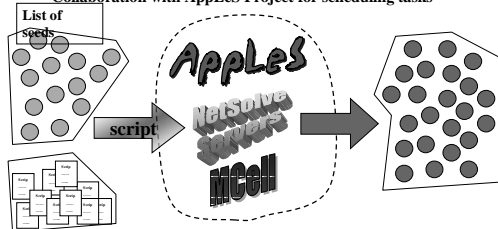
NPACI Alpha Project - MCell: 3-D Monte-Carlo Simulation of Neuro- Transmitter Release in Between Cells

- UCSD (F. Berman, H. Casanova, M. Ellisman), Salk Institute (T. Bartol), CMU (J. Stiles), UTK (Dongarra, R. Wolski)
- Study how neurotransmitters diffuse and activate receptors in synapses
- blue unbounded, red singly bounded, green doubly bounded closed, yellow doubly bounded open



MCell: 3-D Monte-Carlo Simulation of Neuro-Transmitter Release in Between Cells

- Developed at: Salk Institute, CMU
- In the past, manually run on available workstations
- Transparent Parallelism, Load balancing, Fault-tolerance
- Fits the farming semantic and need for NetSolve
- Collaboration with AppLeS Project for scheduling tasks



Web Interface
 Web Server NetSolve Client
 IPARS-enabled Servers

- ♦ Integrated Parallel Accurate Reservoir Simulator.
 - Mary Wheeler's group, UT-Austin
- ♦ Reservoir and Environmental Simulation.
 - models black oil, waterflood, compositions
 - 3D transient flow of multiple phase
- ♦ Integrates Existing Simulators.
- ♦ Framework simplified development
 - Provides solvers, handling for wells, table lookup.
 - Provides pre/postprocessor, visualization.
- ♦ Full IPARS access without Installation.
- ♦ IPARS Interfaces:
 - C, FORTRAN, Matlab, Mathematica, and Web.

74

Data Management

- ♦ Recent work experimenting with data staging and wide area caching - leveraging Distributed Storage Infrastructures (currently IBP).
- ♦ Management and Caching of large objects (i.e. input/output data sets, NOT "messages").

75

Data Management: Work in Progress

- ♦ NetSolve integration with Internet Backplane Protocol (<http://icl.cs.utk.edu/ibp>).
- ♦ Distributed Storage Infrastructure (DSI) that is used to access and manage remote storage components.
- ♦ Daemon runs on storage server and programs use IBP's API to store and manage data for later retrieval (not necessarily from same host).
- ♦ Uses a "capability" or handle to encapsulate remote data's location and ~~accessibility restrictions~~

76

NetSolve and IBP

- ♦ NetSolve API extended with functions to create, destroy, read and write IBP storage.
- ♦ API can also store "named" IBP capabilities at NetSolve agent so that multiple NetSolve clients can use name to access same data.
- ♦ For computational services, client can choose to use either remote or local data sets.

77

NetSolve and IBP (cont'd)

Example of caching near pool of servers.

```
int client_program(){
    NS_DSI_FILE * remote_file;
    NS_DSI_OBJECT * remote_object;
    int *data1, size_data1;

    ...
    remote_file = ns_dsi_open("machine.domain.edu", "write");
    remote_object = ns_dsi_write(remote_file, data1, size_data1);

    netsolve("compare", data1, remote_object);
}
```

78

NetSolve and IBP (cont'd)

Example of client using remote data:

```
int client_program1(){
    NS_DSI_FILE * remote_file;
    NS_DSI_OBJECT * remote_object;
    int *data1, size_data1;

    remote_file = ns_dsi_open("machine.domain.edu", "write");
    remote_object = ns_dsi_write(remote_file, data1, size_data1);
    store_handle(remote_object, "handle_name");
}

int client_program2(){
    NS_DSI_OBJECT * remote_object;

    remote_object = retrieve_handle("handle_name");
    status = netsolve("compute", remote_object, output);
}
```

79

Data Persistence

- ♦ Chain together a sequence of requests.
- ♦ Analyze parameters to determine data dependencies. Essentially a DAG is created where nodes represent computational modules and arcs represent data flow.
- ♦ Transmit superset of all input/output parameters and make persistent near server(s) for duration of sequence execution.
- ♦ Schedule individual request modules for execution.

80

Request Sequencing

- ♦ **Goals:**
 - Transmit no unnecessary (redundant) data parameters.
 - Ensure all necessary data parameters are transmitted.
 - Execute modules simultaneously whenever possible.

81

Request Sequencing Interface

```
...
netsl("command1", A, B, C);
netsl("command2", A, C, D);
netsl("command3", D, E, F);
...
```

```
...
netsl_begin_sequence( );
netsl("command1", A, B, C);
netsl("command2", A, C, D);
netsl("command3", D, E, F);
netsl_end_sequence(C, D);
...
```

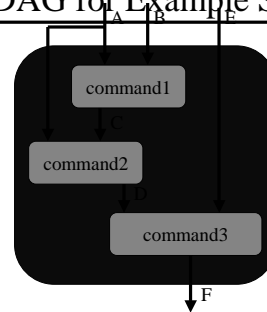
82

DAG Construction

- ♦ "C" Implementation.
- ♦ Analyze all input/output references in the request sequence.
- ♦ Two references are equal if they refer to the same memory address.
- ♦ Size parameters checked for "subset" objects.
- ♦ Only NetSolve MATRICES, VECTORS, and FILES are checked.
- ♦ Constructed DAG scheduled for execution at NetSolve server.

83

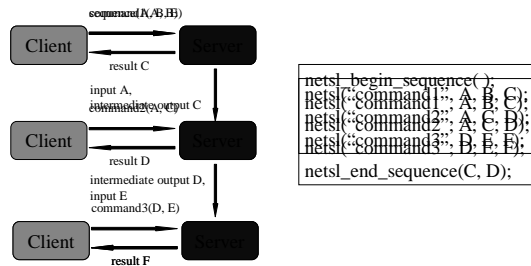
DAG for Example Sequence



```
...
netsl_begin_sequence( );
netsl("command1", A, B, C);
netsl("command2", A, C, D);
netsl("command3", D, E, F);
netsl_end_sequence(C, D);
...
```

84

Data Persistence (cont'd)



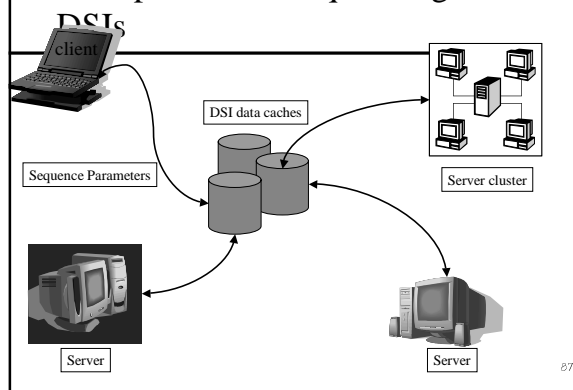
05

Enhanced Sequencing

- ♦ **Multiple NetSolve server sequencing.**
 - First prototype used only single NetSolve server.
 - If no single server possessed all software, sequence could not be executed.
 - Truly parallel execution only on SMPs.
- ♦ **DSIs are heavily leveraged in this on-going work.**

06

Multiple Server Sequencing and



07

Data Management: Future Work

- ♦ **Cache cooperation mechanisms** where distributed caches manage, share and trade both metadata and contents to optimize availability and accessibility.
- ♦ **Grid-accessible data repositories** that scientists can easily utilize within their applications (ideally by a simple abstraction akin to a URL). Promotes Wide Area Collaborations or research on "hard-to-come-by" data. Scientists don't have to worry about storage of raw or input data, only analytical results.

08

Futures for Numerical Algorithms and Software

- ♦ **Numerical software will be adaptive, exploratory, and intelligent**
 - Polyalgorithms, bombardment techniques
- ♦ **Determinism in numerical computing will be gone.**
 - After all, its not reasonable to ask for exactness in numerical computations.
 - Auditability of the computation, reproducibility at a cost
- ♦ **Importance of floating point arithmetic will be undiminished.**
 - 16, 32, 64, 128 bits and beyond.
 - Interval arithmetic
- ♦ **Reproducibility, fault tolerance, and auditability**
- ♦ **Adaptivity is a key so applications can effectively use the resources.**

09

Contributors to These Ideas

- ♦ **Top500**
 - Erich Strohmaier, LBL, NERSC
 - Hans Meuer, Mannheim U
- ♦ **ATLAS**
 - Antoine Petitot, Sun France
 - Clint Whaley, UTK
 - Parallel Computing, Vol 27, No 1-2, pp 3-25, 2001
- ♦ **PAPI**
 - Shirley Browne, UTK
 - Kevin London, UTK
 - Phil Mucci, UTK
 - Keith Seymour, UTK
- ♦ **NetSolve**
 - Dorian Arnold, UTK
 - Susan Blackford, UTK
 - Henri Casanova, UCSD
 - Michelle Miller, UTK
 - Sathish Vadhiyar, UTK

For additional information see...
<http://icl.cs.utk.edu/top500/>
<http://icl.cs.utk.edu/atlas/>
<http://icl.cs.utk.edu/papi/>
<http://icl.cs.utk.edu/netsolve/>
www.cs.utk.edu/~dongarra/

Many opportunities within the group at Tennessee

09

