Benchmarking

Performance required on range of applications

Value of computer dependent on context

Context varies by workload application time

Evaluation valid for one site may not be valid for another

There is no universal metric

LINPACK
Livermore Loops
NAS Parallel Benchmarks
Perfect
SPEC
Parlbench

Twelve Ways to Fool the Masses
D. Bailey, NASA Ames

1. Quote 32-bit performance results, not 64-bit results.
2. Present inner kernel performance figures as the performance of the entire application.
3. Employ assembly code and other low-level language constructs.
4. Scale up the problem size with the number of processors.
5. Quote performance results projected to a full system.
6. Compare parallel results against scalar, unoptimized code on conventional systems.
7. Compare with an old code on an obsolete system.
 Twelve Ways to Fool the Masses (Contd.)

8. Base Mflop/s operation counts on the parallel implementation instead of on the best sequential implementation.

9. Quote performance in processor utilization, parallel speedups or peak Mflop/s.

10. Mutilate the algorithm used in the parallel implementation to match the architecture.

11. Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment.

12. If else fails, show pretty pictures and animated videos, and don’t talk about performance.

LINPACK BENCHMARK

- Linpack Benchmark is really three benchmarks
  
  – Run the Benchmark code, all Fortran, no changes allowed to the software. User only to supply timing function. Benchmark runs the LINPACK routines DGEFA and DGESL which calls the BLAS (DAXPY). Time to factor and solve a system of equations of order 100.
  
  – Run the Benchmark code, solve a system of equations of order 1000. User can supply the software to solve the problem.
  
  – Run the largest system of equations problem on the machine. User can supply the software to solve the problem.

Interested in ...

\( R_{\text{max}} \) Gflop/s for largest problem

\( N_{\text{max}} \) size of largest problem

\( N_{1/2} \) size where half the \( R_{\text{max}} \) achieved

\( R_{\text{peak}} \) theoretical peak performance

Must get the “correct results”.

The results were checked for accuracy by calculating a residual for the problem \( Ax - b \|/ \| A \| \| x \| \).

The execution rate is computed using \( 2/3n^3 + 2n^2 \) operations.
Table 1: Performance in Solving a System of Linear Equations

<table>
<thead>
<tr>
<th>Computer</th>
<th>LINS/PM Benchmark</th>
<th>n = 100</th>
<th>LINS/PM</th>
<th>Benchmark</th>
<th>n = 1000</th>
<th>Mflop/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray X/MSP (proc. 4)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3D (proc. 2)</td>
<td>5.0/0.4</td>
<td>15.0</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3D (proc. 2)</td>
<td>5.0/0.4</td>
<td>15.0</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3D (proc. 2)</td>
<td>5.0/0.4</td>
<td>15.0</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3D (proc. 2)</td>
<td>5.0/0.4</td>
<td>15.0</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3D (proc. 2)</td>
<td>5.0/0.4</td>
<td>15.0</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3D (proc. 2)</td>
<td>5.0/0.4</td>
<td>15.0</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3D (proc. 2)</td>
<td>5.0/0.4</td>
<td>15.0</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3D (proc. 2)</td>
<td>5.0/0.4</td>
<td>15.0</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3D (proc. 2)</td>
<td>5.0/0.4</td>
<td>15.0</td>
<td>15.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Highly Parallel Computing

<table>
<thead>
<tr>
<th>Computer</th>
<th>Number of Procecssors</th>
<th>Mflop/s</th>
<th>Under</th>
<th>Order</th>
<th>Chip/Gp/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Paragon X/MSP</td>
<td>0.0 MHz OS - SUNMOS</td>
<td>510.0</td>
<td>25.0</td>
<td>25.0</td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NEC SX-4/0 (proc. 2)</td>
<td>10.0/0.4</td>
<td>3040</td>
<td>3040.0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Theoretical
Solving a System of Dense Linear Equations

Linpack-HPC Group/

© Universität Mannheim
TOP500 - Manufacturers

Performance [GFlop/s]

Cray Japan Inc. IBM SGI TMC intel


Commerical HPC Vendors

Status

Out of Business
Alliant, American Supercomputer, Astromatics, BBN, CDC, CED, ECN, Fujitsu, NEC, Toshiba, SGI, Total

Currently active
Convex/HP, Cyber Research, Cray Research, Fujitsu, IBM, Intel, NEC, NCUBE, floppy, Fujitsu, NEC, Parsytec, SGI, Terra

Divison closed
Celerity, E&S, FPS, Goedheer, Gould, Local, Merged

Down, not out
Astronautics, BBN, CDC, ETA Systems, ESS, FPS, Goedheer, Gould, Local

Merged
Ardent/Stiller, Ardent/Unisys, Celerity, E&S, FPS, Goedheer, Gould, Local

Status

Out of Business
Alliant, American Supercomputer, Astromatics, BBN, CDC, CED, ECN, Fujitsu, NEC, Toshiba, SGI, Total

Currently active
Convex/HP, Cyber Research, Cray Research, Fujitsu, IBM, Intel, NEC, NCUBE, floppy, Fujitsu, NEC, Parsytec, SGI, Terra

Divison closed
Celerity, E&S, FPS, Goedheer, Gould, Local, Merged

Down, not out
Astronautics, BBN, CDC, ETA Systems, ESS, FPS, Goedheer, Gould, Local

Merged
Ardent/Stiller, Ardent/Unisys, Celerity, E&S, FPS, Goedheer, Gould, Local

Commerical HPC Vendors
Highly Parallel Supercomputing: Where Are We?

1. Performance:

- Sustained performance has dramatically increased during the last year.
- On most applications, sustained performance per dollar now exceeds that of conventional supercomputers.
  
  But
  
  - Conventional systems are still faster on some applications.

2. Languages and compilers:

- Standardized, portable, high-level languages such as HPF, PVM and MPI are available.
  
  But
  
  - Initial HPF releases are not very efficient.
  
  - Message passing programming is tedious and hard to debug.
  
  - Programming difficulty remains a major obstacle to usage by mainstream scientists.

<table>
<thead>
<tr>
<th>Country</th>
<th>Population (in 1K)</th>
<th># of entries</th>
<th>Population (in 1K) per HPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switzerland</td>
<td>6,813</td>
<td>9</td>
<td>757</td>
</tr>
<tr>
<td>Singapore</td>
<td>2,769</td>
<td>3</td>
<td>923</td>
</tr>
<tr>
<td>USA</td>
<td>255,200</td>
<td>261</td>
<td>978</td>
</tr>
<tr>
<td>Denmark</td>
<td>5,158</td>
<td>4</td>
<td>1290</td>
</tr>
<tr>
<td>Norway</td>
<td>4,288</td>
<td>3</td>
<td>1429</td>
</tr>
<tr>
<td>Finland</td>
<td>5,008</td>
<td>3</td>
<td>1669</td>
</tr>
<tr>
<td>Germany</td>
<td>80,250</td>
<td>48</td>
<td>1672</td>
</tr>
<tr>
<td>Netherlands</td>
<td>15,160</td>
<td>9</td>
<td>1684</td>
</tr>
<tr>
<td>Japan</td>
<td>124,500</td>
<td>73</td>
<td>1705</td>
</tr>
<tr>
<td>Hong Kong</td>
<td>5,800</td>
<td>3</td>
<td>1933</td>
</tr>
<tr>
<td>Sweden</td>
<td>8,652</td>
<td>4</td>
<td>2163</td>
</tr>
<tr>
<td>France</td>
<td>57,180</td>
<td>25</td>
<td>2287</td>
</tr>
<tr>
<td>Austria</td>
<td>7,776</td>
<td>3</td>
<td>2592</td>
</tr>
<tr>
<td>UK</td>
<td>57,700</td>
<td>17</td>
<td>3394</td>
</tr>
<tr>
<td>Canada</td>
<td>27,370</td>
<td>8</td>
<td>3421</td>
</tr>
</tbody>
</table>
Scientific Computing: 1985 vs. 1995

- 1985:
  1. Minisupercomputers (1 - 20 Mflop/s): Alliant, Convex, DEC.
  2. Parallel vector processors (PVP) (20 - 2000 Mflop/s): CRI, CDC, IBM.

- 1995:
  1. RISC workstations (10 - 600 Mflop/s): DEC, HP, IBM, SGI, Sun.
  2. RISC based symmetric multiprocessors (SMP) (0.5 - 7 Gflop/s): Convex, CRI, DEC, and SGI.
  3. Parallel vector processors (1 - 30 Gflop/s): Convex, CRI, Fujitsu, Hitachi, IBM, and NEC.
  4. Highly parallel processors (1 - 150 Gflop/s): Convex/HP, CRI, Fujitsu, IBM, Intel, MasPar, nCUBE.

PARKBENCH
(Parallel Kernels and Benchmarks)

- Founded at a special session of SC’92, convened by Dongarra & Hey
- Initial goal - Convergence on an agreed set of benchmarks and procedures within 12 months
- Membership open to all - voting rights acquired by attendance at meetings, a la HPF
- Three monthly meetings with interim email discussion
  – Mail reflector - parkbench-comm@cs.utk.edu
- Supported by most many benchmarking groups
- Supported by computer vendors - Cray Research, Intel, IBM, Meiko, Sun,…
PARKBENCH OBJECTIVES

- To establish a comprehensive set of parallel benchmarks that is generally accepted by both users and vendors of parallel systems.
- To provide a focus for parallel benchmark activities and avoid unnecessary duplication of effort and proliferation of benchmarks.
- To set standards for benchmarking methodology and result-reporting together with a control database/repository for both benchmarks and the results.
- To make the benchmarks and results freely available in the public domain.

MOTIVATION

- Lack of distributed memory message passing benchmarks that were generally acceptable to both users and system vendors

PARKBENCH STRUCTURE

Chairman: Tony Hey (Southampton)
Vice Chair: Jack Dongarra (Knoxville)
Secretary: Mike Berry (Knoxville)

Work divided between 5 subcommittees:

- Methodology (David Bailey)
- Low-Level Benchmarks (Roger Hockney)
- Kernel Benchmarks (Tony Hey)
- Compact Applications (David Walker)
- HPF Compiler Benchmarks (Tom Haupt)
PARKBENCH OUTPUTS

- Draft paper presented at SC’93 and SC’94 Describes philosophy, methodology, current benchmark suite, procedure for submission of compact applications.

- First release of Low-level and Kernel benchmark suite assembled Compromises codes from Genesis, LAPACK & NAS Based on Fortran 77 + PVM for message-passing. Future versions will use MPI for message-passing, also HPF & Fortran 90

- Proposed applications areas and procedure for submission of compact applications identified

- Email discussion group set up - parkbench-comm@cs.utk.edu

- Prototype Performance Database Server set up on Xnetlib and the WWW via Mosaic
  
  - provides access to benchmarks results and literature.

- Finalized, first report available:


METHODOLOGY  David Bailey, NASA Ames

- Units
- Metrics
- Procedure
- Optimization
- Source
- Performance Database
LOW-LEVEL BENCHMARKS
Roger Hockney, Southampton University
Use of Linpack and Livermore Loops recommended to examine single node performance but are not part of suite. Low-Level benchmarks measure basic machine parameters, mostly derived from Genesis.

TICK1 Measurement of clock resolution
TICK2 Measurement of clock correctness
RINF1 Measurement of vectorization parameters \( r_\infty \) and \( n_{1/2} \)
POLY1 In-cache memory bottleneck
POLY2 Out-of-cache memory bottleneck
POLY3 Communications-bottleneck
COMMS1 Unidirectional single message transfer ('pingpong' benchmark)
COMMS2 Bidirectional exchange of two messages ('pingping' benchmark)
COMMS3 Saturation Bandwidth
SYNCH1 Barrier synchronization cost

KERNEL BENCHMARKS
Tony Hey, Southampton University

- Matrix Kernels (ScaLAPACK/PVM)
  - Dense matrix multiply
  - Transpose
  - Dense LU factorization with partial pivoting
  - QR Decomposition
  - Matrix tridiagonalization
- FFT Kernels - 3-D Transforms (NAS)
- PDE Kernels
  - Multigrid (NAS)
- Miscellaneous
  - Embarrassingly Parallel (NAS)
  - Large Integer Sort (NAS)
  - Conjugate Gradient (NAS)
  - Input/Output
PARKBENCH KERNEL BENCHMARKS

- All Available (Input/Output - pencil and paper style)
- NAS Benchmarks - NASA in-house parallel implementations available for the Intel iPSC and the CM-2 and PVM3.3

PARKBENCH COMPACT APPLICATIONS

- SEIS1.2
  - Seismic processing performance evaluation suite from ARCO
  - PVM and NX versions available
- POLMP
  - Proudman Oceanographic Laboratory Multiprocessing Program from the UK Natural Environment Research Council sequential, HPF, and PVM versions available
- PSTSWM
  - Parallel Spectral Transform Shallow Water Model from ORNL
  - PVM and MPI versions available
- NAS
  - BT, SP and LU
  - PVM version in testing, MPI in progress
Anticipated applications areas include:

- Climate and meteorological modelling
- Computational Fluid Dynamics
- Molecular Dynamics
- Finance, e.g. portfolio optimization
- Plasma Physics
- Quantum Chemistry
- Quantum Chromodynamics
- Oil-Reservoir Modelling

PARKBENCH HPF COMPILER BENCHMARKS

- Synthetic applications to test aspects of HPF compilation
- Addresses HPF subset
  
  - FORALL statement - kernel FL
  - Explicit template - kernel TL
  - Communication detection in array assignments
    * kernels AA, SH, ST (structured communication)
    * kernel IR (unstructured communication)
  - Non-elemental intrinsic functions - kernel RD
  - Passing distributed arrays as subprogram arguments
    * kernels AS, IT, IM
PARKBENCH FURTHER WORK REQUIRED

- Investigation of suitable codes for the Input/Output Kernel
- Uniformity of benchmark codes
  - Uniform timer routine (currently, gettimeofday() )
  - Language conformance - C and F77 (with defined extensions)
    * PVM and MPI message passing
    * no machine dependent code
- Standard Makefile
- Minimize Interactive features
- Single source code file if possible
- Standard data sets for benchmarks or a configuration file with
details of how to alter the data and the effect on memory require-
ment
- Reorganize Server Repository (in progress)
  - Headings to clearly identify each benchmark
  - Identify available versions e.g. PVM, MPI
  - Check each code is the current version
  - Test each code on a number of machines
  - Check that documentation is included
- Methods for verifying benchmark results

PARKBENCH FURTHER WORK REQUIRED

- Define Rules for Running the Benchmarks
  - Possibly: One result based on unmodified source code with com-
    piler flags listed
  - A second result which allows any modification to optimize per-
    formance, so long as the final result is correct.
- Further results are required -
  - For GBIS Graphical Interface
  - These results will be forwarded for inclusion in PDS
- Define a method for reporting bugs
- Future - MPI versions of all codes
  - Benchmarks to test gather/scatter/global sum
  - Shared memory versions of codes (using defined primitives, e.g.
    put/get)
  - GUI for setting up and running codes
- Further Compact Applications (e.g. Database applications)
- Analysis of Kernel Benchmarks in terms of low level parameters
STATUS UPDATE
Kernel Benchmarks are publicly available from Netlib:
Contents of the kernels suite:

- Matrix benchmarks
  - Dense matrix multiply
  - Matrix transpose
  - Dense LU factorization with partial pivoting.
  - QR Decomposition.
  - Matrix tridiagonalization.
- Fourier transformations
  - 1-D FFT
  - 3-D FFT
- PDE kernels
  - SOR
  - MG
- Other
  - EP
  - CG
  - IS

PARKBENCH POLMP RESULTS

- Cray Y-MP/8
- Cray C90
- Intel iPSC/860
- Maspar MP-1104
- Meiko CS-1 i860
- TMC CM-200 16k

Results have been published in the following three papers:
Ashworth, M., Parallel Processing Environmental Modelling,
Proceedings of the Fifth Workshop on the Use of Parallel
Processors in Meteorology, ECMWF 23-27 November 1992,
(World Scientific), pp 1-25.

Ashworth, M. and Davies, A.M., Performance of a Three
Dimensional Hydrodynamic Model on a Range of Parallel
Computers, in Proceedings of the Euromicro Workshop on
Parallel and Distributed Computing, Gran Canaria 27-29

in Environmental Modelling, in Proc. of the Conference
Parallel Computational Fluid Dynamics ’93, Paris,
10th-12th May 1993 (Elsevier), in press, 10pp.

The last of these is available in PostScript
by anonymous ftp to bisag.nbi.ac.uk in
/pub/neurosuper/reports/pcfdet93.ps.
Rules

- Run the codes “as is”, must get the correct solution.
- Run driver with a version that must be a supported library that is available with a brief description of what is done (for disclosure).
- We reserve the right to verify the results of the benchmarks.
- Problem size: test, medium, and large
- Data maintained in www.netlib.org.

PARKBENCH

ON-GOING WORK

- Uniformity of benchmark codes
  
  Uniform timer routine
  
  Language conformance
  
  Remove Interactive features
  
  Single source file, machine dependent code in separate file

- Further Results
**PARKBENCH AVAILABILITY**

- **PARKBENCH Description, Source codes, Report, Documentation**
  - WWW URL: http://www.netlib.org/parkbench/html/
  - to go directly to the download page
    http://www.netlib.org/parkbench/

- **Genesis Benchmark Information Service (GBIS) Graphical**
  - Interface
  - Interactive graphical results for genesis and PARKBENCH
    http://hpcc.soton.ac.uk/RandD/gbis/papiani-new-gbis-top.html

- **PDS**

- **HPCC National Software Exchange (NSE)**
  - http://www.netlib.org/nse/home.html

---

**Genesis benchmark Information Service (GBIS) Graphical Interface**

- Graphical benchmark results for Genesis and ParkBench Benchmark suites available on the World Wide Web
- Interactive selection of benchmark and computers from which a performance graph is generated and displayed
- Default graph options can be changed:
  - Available output formats: gif, xbm, postscript, tabular
  - Selectable ranges
  - Choice of log/linear axes
  - Key to traces can be positioned as required
- Links to Genesis and ParkBench information including Benchmark documentation, papers, source codes, etc.

- **World Wide Web URL’s:**
  - http://hpcc.soton.ac.uk/RandD/gbis/papiani-new-gbis-top.html
  - http://hpcc.soton.ac.uk/RandD/gbis/papiani-new-gbis.html
  - http://hpcc.soton.ac.uk/RandD/RandD.html
  - **Anonymous ftp Access to Tabular Results:**
  - par.soton.ac.uk
  - in directory /pub/benchmark_results
PARKBENCH

AVAILABILITY

- PARKBENCH Description, Source codes, Report, Documentation
  WWW URL: http://www.epm.ornl.gov/walker/parkbench/
  to go directly to the download page
  http://www.netlib.org/parkbench/
  or via xnetlib

- Genesis Benchmark Information Service (GBIS) Graphical Interface
  Interactive graphical results for genesis and PARKBENCH
  http://hpcc.soton.ac.uk/RandD/GBIS/papiani-new-GBIS-top.html

- PDS

- HPCC National Software Exchange (NSE)
  http://www.netlib.org/nse/home.html

SPEC High Performance Steering Committee

Founded in January 1994

Members have met twice to elect officers, define charter, discuss policies, benchmark selection criteria, run and reporting rules

Current members are:
- Convex Computers
- Cray Research
- Dartmouth College
- Digital Equipment Corporation
- Fujitsu America
- Hewlett-Packard
- IBM
- Intel SSD
- Kuck and Associates
- NEC/HNSX Supercomputers
- Silicon Graphics Inc.
- Sun Microsystems Inc.
- University of Illinois
- University of Michigan
- University of Minnesota
- University of Tennessee
**SPEC/HPSC - Aims and Scope**

HPSC’s mission is to establish, maintain and endorse a suite of benchmarks representative of real world high-performance computing applications, for standardized cross platform comparison on a level playing field.

Aim is to cover all advanced architectures, will focus on
- symmetric multi-processor systems
- workstation clusters
- distributed memory parallel systems
- vector and vector parallel supercomputers

End users are seen as the ultimate beneficiaries of results
- wish to establish a close relationship with the high-performance computing community.

**SPEC/HPSC - Parkbench Relationship**

SPEC/HPSC will draw on Parkbench experience and expertise
- choice of API, choice of codes, methodology

Parkbench more research oriented than SPEC/HPSC
Parkbench will draw on SPEC/HPSC experience
- commercial user interests
- vendor interests

To encourage collaboration and interaction
- Joint B.O.F. meeting at Supercomputing 95
- Parkbench participation at SPEC/HPSC meetings
- Continuing SPEC/HPSC participation at Parkbench meetings
NHSE

- National HPCC Software Exchange
- NASA (plus other agencies) funded CRPC project
- Center for Research on Parallel Computation (CRPC)
  - Argonne National Laboratory
  - California Institute of Technology
  - Rice University
  - Syracuse University
  - University of Tennessee
- Uniform interface to distributed HPCC software repositories
- Facilitation of cross-agency and interdisciplinary software reuse
- Material from ASTA, HPCS, and IIITA components of the HPCC program
- http://www.netlib.org/nhse/

Goals:

- To facilitate an active exchange program for HPCC software and enabling technologies via the National Information Infrastructure.

- To promote contributions and use by Grand Challenge teams, as well as other members of the high performance computing community.

Software here includes algorithms, specifications, designs, documentations, reports...
NHSE Components

- Outreach and technology transition
  - To the HPCC user community and industry
- Distribution via the WWW
- Discipline oriented repositories
- Uniform user interface
- Software review
  - Simple submission
  - Ongoing review
- Measurement
- Hypertext road map
- Publishing tools
  - Repository-in-a-box
- Naming & authentication architecture
- Selective capitalization of emerging technologies

Virtual Repository Architecture
Benefits:

1. Faster development of high-quality software so that scientists can spend less time writing and debugging programs and more time on research problems.
2. Less duplication of software development effort by sharing of software modules.
3. Less time and effort spent in locating relevant software and information through the use of appropriate indexing and search mechanisms and domain-specific expert help systems.
4. Reducing information overload through the use of filters and automatic search mechanisms.

Intended Audience:

- HPCC application and computer science community
- Source of material for NHSE
- Users of NASA, NSF, DOE and other supercomputer centers
- Good targets for NHSE
- Natural support organization: supercomputer center staff
- Other users of high performance computers
- Current and potential industrial users
- No natural support organization
- Applicable to other domains
NHSE

- Based on Existing Technologies
  - WWW Browser (Mosaic / Netscape / etc)
    * Distributed / Scalable
    * URL: http://www.netlib.org/nhse/
  - Netlib
    * Repository for math software since 1985
- Repositories Currently Available
  - Netlib, Softlib, CITlib
  - ASSET - (Asset Source for SW Engineering Tech.)
  - CARDS - (Comprehensive Approach to Reusable Defense SW)
  - ELSA - (Electronic Library Services and Appl.)
  - GAMS (Virtual Software Repository)
  - STARS - (SW Technology for Adaptable, Reliable Systems)
  - Many examples related to GC problems
- Currently Available Information
  - NHSE currently points to 350+ modules
    * software catalog, tech reports and papers
    * parallel processing tools
    * numerical libraries
    * Grand Challenge prototype codes
    * data analysis and visualization
    * benchmarks

Netlib - Network Access to Mathematical Software and Data

- Began in 1985
  - JD and Eric Grosse, AT&T Bell Labs
- Motivated by the need for cost-effective, timely distribution of high-quality mathematical software to the community.
- Designed to send, by return electronic mail, requested items.
- Automatic mechanism for the disseminate of public domain software.
  - Still in use and growing
  - Mirrored at a number of sites
    * netlib2.cs.utk.edu
    * netlib1.epm.ornl.gov
    * research.att.com
    * netlib.no
    * unix.hensa.ac.uk
    * ftp.zip-berlin.de
    * nchc.edu.tw
- Moderated collection of high-quality math software
- Distributed maintenance
- Model for domain-specific repositories
Requests Made to the Netlib Repositories at the Univ. of Tennessee & ORNL

8,946,816 total requests to these repositories as of Feb 15, 1996

<table>
<thead>
<tr>
<th>Library Name</th>
<th>Number of accesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>lapack</td>
<td>475,979</td>
</tr>
<tr>
<td>pvm3</td>
<td>379,849</td>
</tr>
<tr>
<td>linpack</td>
<td>256,403</td>
</tr>
<tr>
<td>slatec</td>
<td>248,292</td>
</tr>
<tr>
<td>blas</td>
<td>178,728</td>
</tr>
<tr>
<td>clapack</td>
<td>129,256</td>
</tr>
<tr>
<td>linalg</td>
<td>127,022</td>
</tr>
<tr>
<td>eispack</td>
<td>126,116</td>
</tr>
<tr>
<td>slatec/src</td>
<td>118,366</td>
</tr>
<tr>
<td>toms</td>
<td>117,152</td>
</tr>
<tr>
<td>f2c</td>
<td>98,025</td>
</tr>
<tr>
<td>c++</td>
<td>96,774</td>
</tr>
<tr>
<td>benchmark</td>
<td>85,552</td>
</tr>
<tr>
<td>master</td>
<td>69,997</td>
</tr>
<tr>
<td>f2c/src</td>
<td>67,415</td>
</tr>
<tr>
<td>mpack</td>
<td>60,632</td>
</tr>
<tr>
<td>fn</td>
<td>59,781</td>
</tr>
<tr>
<td>fftpack</td>
<td>58,805</td>
</tr>
<tr>
<td>na-digest</td>
<td>50,970</td>
</tr>
<tr>
<td>poe</td>
<td>49,496</td>
</tr>
<tr>
<td>slatec/lin</td>
<td>46,800</td>
</tr>
<tr>
<td>hence</td>
<td>45,229</td>
</tr>
<tr>
<td>confdb</td>
<td>43,118</td>
</tr>
<tr>
<td>slatec/chk</td>
<td>37,640</td>
</tr>
<tr>
<td>c++/answerbook</td>
<td>37,524</td>
</tr>
<tr>
<td>napack</td>
<td>36,719</td>
</tr>
</tbody>
</table>
The following types of software are being made available:

- Systems software and software tools.
  - compilers
  - message-passing communication subsystems
  - parallel monitors and debuggers.
- Basic building blocks for accomplishing common computational and communication tasks.
  - Building blocks are meant to be used by Grand Challenge teams
- Research codes that have been developed to solve difficult computational problems.
  - Many have been developed to solve specific problems
  - Serve as proofs of concept
  - Models for developing general-purpose reusable software

Review Procedure

- Submissions are “subject” to (ongoing) review.
- Review status abstract for each submission.
  - Based on author comments
  - Package documentation
  - Our independent reviewer testing
  - Comments from users
NHSE Software Catalog

- Benchmark and example programs (4)
- Data analysis and visualization (22)
- Numerical libraries and routines (57)
  - Computational geometry (7)
  - Linear algebra (18)
  - Optimization (4)
  - Partial differential equations (3)
  - Other (25)
- Parallel processing tools
  - Communication libraries (25)
  - Execution and performance analyzers (31)
  - Parallel I/O systems (5)
  - Parallel programming environments (12)
  - Parallel programming languages and compilers (26)
  - Parallel runtime systems (10)
  - Source code analyzers and restructurers (7)
  - Miscellaneous (16)
- Scientific and engineering applications (66)

NHSE Software Submission

Goals

- Exercise quality control
  (review classification)
- Ensure fixity of publication
  (file fingerprints, unique name)
- Prevent impersonation and unauthorized changes
  (digital signatures)
- Promote interoperability
  (RIG Basic Interoperability Data Model)
Technical and Political Issues

- How will the naive user find the right software?
  - Answer: Via the NHSE search/browser interface and the Road Map
- How will authentication, integrity, and version control be implemented?
  - Answer: By a publishing system that includes unique naming & digital signatures.
- Will the NHSE support distribution of software that is not free or cannot be freely distributed?
  - Answer: Yes, but...
    * only provide classification, review, and access
    * use of encryption and separate key distribution
    * NHSE will not have an accounting department
- Will the NHSE be responsible for support of software?
  - Answer: NO!
    * Any support will be by author (or appropriate agent)

Summary

- Initial implementation built on existing technologies
  - WWW
    * Distributed
    * Scalable
  - Netlib, etc
  - Rapid deployment
- Multilevel review and classification scheme
- Road Map
- Measurement and evaluation
  - Statistics
  - Evaluations for reviewed
Summary (continued)

- Outreach and technology transition
  - Educational activities aimed at the user community
  - Fostering technology development by industry
- Working on standardization within WWW community
  - member of RIG, IETF, IESG, WWW consort

Some URL’s Related to Scientific Computing

- National HPCC Software Exchange
  http://www.netlib.org/nhse/

- NHSE Software Catalog
  http://www.netlib.org/nhse/sw_catalog/index.html

- Netlib Repository
  http://www.netlib.org/

- Computational Science Education
  http://www.netlib.org/nhse/cse_edu.html

- Books, Course Materials, and Tutorials
  http://www.netlib.org/nhse/cse_edu.html#books

- CS267 - Applications of Parallel Computers
  U.C. Berkeley CS267: Spring 1995 Jim Demmel
  http://www.icsi.berkeley.edu/cs267/

- 18.337 Parallel Scientific Computing
  MIT Spring, 1995 Alan Edelman
  http://web.mit.edu/18.337/WWW/home.html

- North Carolina State University
  Visualizations in Materials Science
  http://vims.ncsu.edu/cgi/index.cgi

- Computational Science Education Project
  http://csep1.phy.ornl.gov/csep.html
High Performance Fortran (HPF)

- Low-level parallel programming is hard
- Parallel programs are not portable
- Parallel programs could not be run on sequential computers
- Data parallel programming model seems to be good for many applications
- HPF is an emerging shared memory programming standard for massively parallel computers
- Limited HPF subset compilers

Conclusions on Interconnects

- High bandwidth and low latency remain a challenge for MPP systems
- However, clustered workstations will provide an efficient throughput system for small and medium range jobs
- Latency is particularly poor on workstation clusters and limits their use on most applications
Lessons Learned During 1985-1995

1. Massive parallel designs are as effective as highly parallel.
2. SIMD hardware was not very successful.
3. Designs employing RISC processors are quite successful.
4. Interconnect latency and bandwidth are very important.
5. Reliable system software is a challenge.
6. Efficient implementations of good serial algorithms do the job and are best parallel algorithms.
7. Different applications have different performance characteristics.

The Maturation of Highly Parallel Technology

- Affordable parallel systems now out-perform the best conventional supercomputers.
- Performance per dollar is particularly favorable.
- The field is thinning to a few very capable systems.
- Reliability is greatly improved.
- Third-party scientific and engineering applications are appearing.
- Business applications are appearing.
- Commercial customers, not just research labs, are acquiring systems.
Vector Parallel Architecture

- Peak performance of vector processor is at least 3-4 times better than microprocessor. Peak performance of a single NEC SX-4 processor is 2.0 Gflop/s and DEC 21164 300 MHz is 600 Mflop/s.
- PVP needs only 1/4 parallelism of HPP
- PVP can use crossbar providing high network efficiency
- Ease of programming

RISC Workstations: The Enemy Below

- Sales revenues now exceed that of mainframe supers and minisupers combined.
- Most include high-resolution color graphics.
- Highly reliable hardware and software.
- Single-CPU performance approaches that of vector supers.
- Vigorous competition between leading vendors assures continued rapid increases in sustained performance.

Lesson to supercomputers centers:

Those centers that do not give priority to large, grand challenge jobs risk losing their funding.

Lesson to supercomputers manufacturers:

Those vendors that do not press the outer envelope of single job performance risk being driven out of business.
Current State of HPC Market

<table>
<thead>
<tr>
<th>Number of installations</th>
<th>Installed Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>$500 M</td>
</tr>
<tr>
<td>1,000</td>
<td>$4,000 M</td>
</tr>
</tbody>
</table>

Big Science

Production Engineering

Memory Latencies

<table>
<thead>
<tr>
<th>Function</th>
<th>EV4 used in Cray T3D</th>
<th>EV5 used in Cray T3E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time in CP</td>
<td>Time in ns</td>
</tr>
<tr>
<td>L1 cache load</td>
<td></td>
<td></td>
</tr>
<tr>
<td>L2 cache load</td>
<td>na</td>
<td>na</td>
</tr>
<tr>
<td>RA/stream hit</td>
<td>13</td>
<td>86.71</td>
</tr>
<tr>
<td>DRAM page hit</td>
<td>22</td>
<td>146.74</td>
</tr>
<tr>
<td>DRAM page miss</td>
<td>37</td>
<td>246.79</td>
</tr>
</tbody>
</table>
CRAY T3E Memory Hierarchy

- Small
  - Registers
  - Data Cache (L1 + L2)
  - Local Memory
  - Remote Memory
  - Disk Memory

- Fast
  - 256B
  - 104 KB
  - L1 8 KB
  - L2 96 KB
  - 64 MB
  - 65 GB
  - 100 GB

- Local Memory

- Remote Memory

- Disk Memory

Power Challenge Memory Hierarchy

- Smallest
  - Registers
  - 1st level data cache (on-chip; in IU)
  - 2nd level data cache (off-chip; in FPU)
  - Streaming data cache

- Fastest
  - 256B
  - 16 KB
  - 4-16 MB
  - 2-16 GB
  - 24 GB
  - 4800 MB/s
  - 4800 MB/s
  - 1200 MB/s
  - 1200 MB/s
  - 20 MB/s
  - Slowest