HPC Challenge v1.x Benchmark Suite
SC|05 Tutorial — S13

Dr. Piotr Luszczek (luszczek@cs.utk.edu)
Dr. David Koester (dkoester@mitre.org)

13 November 2005
Afternoon Session
Acknowledgements

• This work was supported in part by the Defense Advanced Research Projects Agency (DARPA), the Department of Defense, the National Science Foundation (NSF), and the Department of Energy (DOE) through the DARPA High Productivity Computing Systems (HPCS) program under grant FA8750-04-1-0219 and under Army Contract W15P7T-05-C-D001

• Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government
People

• ICL Team
  - Jack Dongarra
dongarra@cs.utk.edu
University
Distinguished Professor
  - Piotr Luszczek
luszczek@cs.utk.edu
Research Scientist

• Collaborators
  - David Bailey (Lawrence Berkeley National Laboratory)
  - David Koester (MITRE)
  - John McCalpin (IBM)
  - Rolf Rabenseifner (The High Performance Computing Center Stuttgart)
  - R. Clint Whaley (University of Texas San Antonio)
  - Jeremy Kepner (MIT LL)
  - Bob Lucas (USC/ISI)
  - Antoine Petitet (Sun Microsystems)
  - Daisuke Takahashi
daisuke@is.tsukuba.ac.jp
University of Tsukuba
HPC Challenge v1.x Benchmark Suite Outline

- Introduction
- Motivations
  - HPCS
  - Performance Characterization
- Component Kernels
- HPC Challenge Awards
- Unified Benchmark Framework
- Rules
  - Running HPC Challenge
  - Optimizations
  - Etiquette
- Performance Data
  - Available Benchmark Data
  - Kiviat Charts
- Hands-on Demonstrations/Exercises
  - Installing the HPC Challenge v1.x Benchmark Unified Benchmark Framework
  - Running the HPC Challenge v1.x Benchmark suite
- Summary/Conclusions
HPC Challenge Benchmark Suite

http://icl.cs.utk.edu/hpcc/

HPC Challenge Benchmark

The HPC Challenge benchmark consists of basically 7 tests:

1. **HPL** - the Linpack TPP benchmark which measures the floating point rate of execution for solving a linear system of equations.

2. **DGEMM** - measures the floating point rate of execution of double precision real matrix-matrix multiplication.

3. **STREAM** - a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.

4. **PTRANS** (parallel matrix transpose) - exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.

5. **RandomAccess** - measures the rate of integer random updates of memory (GUPS).

6. **FFT** - measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).

7. Communication bandwidth and latency - a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns; based on b_eff (effective bandwidth benchmark).
HPC Challenge v1.x Benchmark Suite
Introduction (1 of 6)

• HPCS has developed a spectrum of benchmarks to provide different views of system
  – ~40 Kernel Benchmarks
  – HPCS Spanning Set of Kernels
  – HPC Challenge Benchmark Suite
• HPC Challenge Benchmark Suite
  – To examine the performance of HPC architectures using kernels with more *challenging* memory access patterns than HPL
  – To *augment* the Top500 list
  – To provide benchmarks that *bound* the performance of many real applications as a function of memory access characteristics — e.g., spatial and temporal locality
  – To outlive HPCS
• HPCchallenge pushes spatial and temporal boundaries and defines performance bounds

Available for download http://icl.cs.utk.edu/hpcc/
HPC Challenge v1.x Benchmark Suite
Introduction (2 of 6)

1. HPL — High Performance LINPACK
2. DGEMM — matrix x matrix multiply
3. STREAM
   - Copy
   - Scale
   - Add
   - Triad
4. PTRANS — parallel matrix transpose
5. FFT
6. RandomAccess
7. Communications Bandwidth and Latency

• Scalable framework — Unified Benchmark Framework
  - By design, the HPC Challenge Benchmarks are scalable with the size of data sets being a function of the largest HPL matrix for the tested system
## HPC Challenge v1.x Benchmark Suite

### Introduction (3 of 6)

<table>
<thead>
<tr>
<th>Local and Embarrassingly Parallel</th>
<th>Global</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. EP-DGEMM (matrix x matrix multiply)</td>
<td>1. High Performance LINPACK (HPL)</td>
</tr>
<tr>
<td>2. STREAM</td>
<td></td>
</tr>
<tr>
<td>- Copy</td>
<td></td>
</tr>
<tr>
<td>- Scale</td>
<td></td>
</tr>
<tr>
<td>- Add</td>
<td></td>
</tr>
<tr>
<td>- Triad</td>
<td>2. PTRANS — parallel matrix transpose</td>
</tr>
<tr>
<td>5. Communications Bandwidth &amp; Latency</td>
<td>5. Communications Bandwidth &amp; Latency</td>
</tr>
</tbody>
</table>

| 1. HPL — High Performance LINPACK  |
| 2. DGEMM — matrix x matrix multiply |
| 3. STREAM                            |
|   - Copy                             |
|   - Scale                            |
|   - Add                              |
|   - Triad                            |
| 4. PTRANS — parallel matrix transpose |
| 5. FFT (EP & G)                      |
| 7. Communications Bandwidth and Latency |
Many of the component benchmarks were widely used before the HPC Challenge suite of Benchmarks was assembled

- HPC Challenge has been more than a packaging effort
- Almost all component benchmarks were augmented from their original form to provide consistent verification and reporting

We stress the importance of running these benchmarks on a single machine — with a single configuration and options

- The benchmarks were useful separately for the HPC community, meanwhile
- The unified HPC Challenge framework creates an unprecedented view of performance characterization of a system
  - A comprehensive view with data captured the under the same conditions allows for a variety of analyses depending on end user needs
To characterize a system architecture — consider three testing scenarios:

1. Local – only a single processor is performing computations
2. Embarrassingly Parallel – each processor in the entire system is performing computations but they do not communicate with each other explicitly
3. Global – all processors in the system are performing computations and they explicitly communicate with each other.

All benchmarks operate on either matrices (of size $n^2$) or vectors (of size $m$)

- $n^2 \leq m \leq$ Available Memory
- i.e., the matrices or vectors are large enough to fill almost all available memory.
HPC Challenge v1.x Benchmark Suite
Introduction (6 of 6)

• HPC Challenge encourages users to develop optimized benchmark codes that use architecture specific optimizations to demonstrate the best system performance

• Meanwhile, we are interested in both
  – The base run with the provided reference implementation
  – An optimized run

• The base run represents behavior of legacy code because
  – It is conservatively written using only widely available programming languages and libraries
  – It reflects a commonly used approach to parallel processing sometimes referred to as hierarchical parallelism that combines
    ▪ Message Passing Interface (MPI)
    ▪ OpenMP Threading
  – We recognize the limitations of the base run and hence we encourage optimized runs

• Optimizations may include alternative implementations in different programming languages using parallel environments available specifically on the tested system

• We require that the information about the changes made to the original code be submitted together with the benchmark results
  – We understand that full disclosure of optimization techniques may sometimes be impossible
  – We request at a minimum some guidance for the users that would like to use similar optimizations in their applications
HPC Challenge v1.x Benchmark Suite
Outline

- Introduction
- Motivations
  - HPCS
  - Performance Characterization
- Component Kernels
- HPC Challenge Awards
- Unified Benchmark Framework
- Rules
  - Running HPC Challenge
  - Optimizations
  - Etiquette
- Performance Data
  - Available Benchmark Data
  - Kiviat Charts
- Hands-on Demonstrations/Exercises
  - Installing the HPC Challenge v1.x Benchmark Unified Benchmark Framework
  - Running the HPC Challenge v1.x Benchmark suite
- Summary/Conclusions
High Productivity Computing Systems (HPCS)

Goal:
- Provide a new generation of economically viable high productivity computing systems for the national security and industrial user community (2010)

Impact:
- **Performance** (time-to-solution): speedup critical national security applications by a factor of 10X to 40X
- **Programmability** (idea-to-first-solution): reduce cost and time of developing application solutions
- **Portability** (transparency): insulate research and operational application software from system
- **Robustness** (reliability): apply all known techniques to protect against outside attacks, hardware faults, & programming errors

Applications:
- Intelligence/surveillance, reconnaissance, cryptanalysis, weapons analysis, airborne contaminant modeling and biotechnology

Fill the Critical Technology and Capability Gap
Today (late 80’s HPC technology)…..to…..Future (Quantum/Bio Computing)
HPCS Program Phases I - III

Productivity Assessment (MIT LL, DOE, DoD, NASA, NSF)

Industry Milestones

MP Peta-Scale Procurements

Mission Partner Peta-Scale Application Dev

MP Language Dev

Year (CY)

02 03 04 05 06 07 08 09 10 11

(Funded Five) Phase I Industry Concept Study

(Funded Three) Phase II R&D

Phase III Prototype Development

Program Reviews

Critical Milestones

Program Procurements

Mission Partners

Council on Competitiveness
Phase II Program Goals

• Phase II Overall Productivity Goals
  – Execution (sustained performance) – 1 Petaflop/sec (scalable to greater than 4 Petaflop/sec). Reference: Workflow 3
  – Development – 10X over today’s systems. Reference: Workflows 1, 2, 4, 5

• Productivity Framework
  – Establish experimental baseline
  – Evaluate emerging vendor execution and development productivity concepts
  – Provide a solid reference for evaluation of vendor’s Phase III designs
  – Early adoption or phase in of execution and development metrics by mission partners

• Subsystem Performance Indicators (Vendor Specified Goals)
  – 3.2 PB/sec bisection bandwidth;
  – 64,000 GUPS;
  – 6.5 PB/sec data streams bandwidth;
  – 2+ PF/s Linpack

Documented and Validated Through Simulations, Experiments, Prototypes, and Analysis
HPCS I/O Challenges

An Envelope on HPCS Mission Partner Requirements

- 1 Trillion files in a single file system
  - 32K file creates per second
- 10K metadata operations per second
  - Needed for Checkpoint/Restart files
- Streaming I/O at 30 GB/sec full duplex
  - Needed for data capture
- Support for 30K nodes
  - Future file system need low latency communication
Productivity Team Lead

Jeremy Kepner

September 2003 — July 2005
(Phase II Years 1 and 2)

Development Experiments
Existing Code Analysis
Workflows, Models, Metrics
Benchmarks
High Productivity Language Systems
Execution Time Models
Test & Specifications

July 2005 — ??
(Phase II Year 3 and Early Phase III)

Development Experiments
Workflows, Models, Metrics
High Productivity Language Systems
Execution Time Models
Test & Spec Specifications
HPCS Benchmark Working Group Goals

- Provide the HPCS Vendors and HPCS Productivity Team the Benchmarks and Applications for
  - Scoping requirements for designing systems
  - Productivity Testing
    - Execution Time Testing
    - Development Time Testing

System Parameters (Examples)
- BW bytes/flop (Balance)
- Memory latency
- Memory size
- Processor flop/cycle
- Processor integer op/cycle
- Bisection BW
- Size (ft³)
- Power/rack
- Facility operation
- Code size
- Restart time (Reliability)
- Code Optimization time

Benchmarks

Productivity = Utility/Cost
\[ \Psi = \frac{U}{C} = \frac{U(T)}{C_S + C_D + C_M} \]

Utility → U(T)

Production

Constant
HPCS Benchmark Working Group

Goals

Benchmarks and Workflows are non-linear functions representing HPCS Mission Partner requirements that will enable the measurement of the productivity terms utility and cost for systems represented by \textit{traditional} parameter sets.

System Parameters (Examples)
- BW bytes/flop (Balance)
- Memory latency
- Memory size
- Processor flop/cycle
- Processor integer op/cycle
- Bisection BW
- Size (ft$^3$)
- Power/rack
- Facility operation
- Code size
- Restart time (Reliability)
- Code Optimization time

Benchmarks

Exe Time Experiments

Productivity

Actual System or Model

Productivity Metrics

Work Flows

Dev Time Experiments

Utility $\rightarrow U(T)$

Productivity $= \frac{U}{C} = \frac{U(T)}{C_S + C_O + C_M}$

Utility $\rightarrow U(T)$

Production $T$

Constant $T$
Spectrum of benchmarks provide different views of system

- HPCchallenge pushes spatial and temporal boundaries; sets performance bounds
- Applications drive system issues; set legacy code performance bounds
- Kernels and Compact Apps for deeper analysis of execution and development time
HPC Challenge v1.x Benchmark Suite

Introduction (1 of 2)

HPCchallenge Benchmarks
http://icl.cs.utk.edu/hpcc/

- To examine the performance of HPC architectures using kernels with more **challenging** memory access patterns than HPL
- To **augment** the Top500 list
- To provide benchmarks that **bound** the performance of many real applications as a function of memory access characteristics — e.g., spatial and temporal locality
- To outlive HPCS

- HPCchallenge pushes spatial and temporal boundaries; sets performance bounds
- Available for download http://icl.cs.utk.edu/hpcc/
Government HPC (HPCS) Benchmark Spectrum

HPCchallenge Benchmarks
http://icl.cs.utk.edu/hpcc/

- Local and Embarrassingly Parallel
  1. EP-DGEMM (matrix x matrix multiply)
  2. STREAM
     - COPY
     - SCALE
     - ADD
     - TRIADD
  3. EP-RandomAccess
  4. EP-1DFFT

- Global
  1. High Performance LINPACK (HPL)
  2. PTRANS — parallel matrix transpose
  3. G-RandomAccess
  4. G-1DFFT
  5. Communication Bandwidth & Latency

Version 1.0 Now Available!

• Scalable framework — Unified Benchmark Framework
  – By design, the HPC Challenge Benchmarks are scalable with the size of data sets being a function of the largest HPL matrix for the tested system
Motivation for More “Challenging” Benchmarks

• “To examine the performance of HPC architectures using kernels with more *challenging* memory access patterns than HPL”

• Briefly address the questions:
  – What effects do more challenging memory access patterns have on performance?
  – What applications exhibit more challenging memory access patterns?
Uniprocessor Sparse Matrix-Vector Multiply Performance

Source: R. Vuduc, J. Demmel, K. Yelick, UC Berkeley
Uniprocessor Sparse Matrix-Vector Multiply Performance

- **HPL** — dense linear solver
  - High temporal locality or data reuse due to blocked data
  - Architecture able to move data to processors to keep them busy
- **Sparse linear solvers**
  - Difficulties in keeping data moving to the processors to keep them busy

Source: R. Vuduc, J. Demmel, K. Yelick, UC Berkeley
Framework addition: Data Dependency

HABU Memory Performance

PMaC
Performance Modeling and Characterization Lab
San Diego Supercomputer Center
Framework addition: Data Dependency

Intel Itanium MAPS Graph

Itanium Memory Bandwidth

- Stride-one
- Random-stride
- Branch
- FP dep

PMaC Performance Modeling and Characterization Lab
San Diego Supercomputer Center
Node Spatial and Temporal Locality

Generated by PMaC @ SDSC

HPC Challenge Benchmarks

- HPL
- Gamess
- Overflow
- Test3D
- OOCore
- HYCOM
- RFCTH2
- AVUS
- CG
- RandomAccess
- STREAM

- Spatial and temporal data locality here is for one node/processor — i.e., locally or “in the small”
Node Spatial and Temporal Locality

HPC Challenge Benchmarks

- High Temporal Locality
  - Good Performance on Cache-based systems

- No Temporal or Spatial Locality
  - Poor Performance on Cache-based systems

- High Spatial Locality
  - Moderate Performance on Cache-based systems

Generated by PMaC @ SDSC
Node Spatial and Temporal Locality

HPC Challenge v1.x Benchmarks

SC|05 Tutorial S-13

MITRE

Generated by PMaC @ SDSC

High Temporal Locality
Good Performance on
Cache-based systems

HPC Challenge Benchmarks

HPC Challenge Benchmarks
“bound” real application
performance in the
locality space

No Temporal or Spatial Locality
Poor Performance on
Cache-based systems

High Spatial Locality
Moderate Performance on
Cache-based systems
HPC Challenge v1.x Benchmark Suite Outline

- Introduction
- Motivations
  - HPCS
  - Performance Characterization
- Component Kernels
- HPC Challenge Awards
- Unified Benchmark Framework
- Rules
  - Running HPC Challenge
  - Optimizations
  - Etiquette
- Performance Data
  - Available Benchmark Data
  - Kiviat Charts
- Hands-on Demonstrations/Exercises
  - Installing the HPC Challenge v1.x Benchmark Unified Benchmark Framework
  - Running the HPC Challenge v1.x Benchmark suite
- Summary/Conclusions
HPC Challenge v1.x Benchmark Suite
Component Kernels

- HPL (High Performance Linpack)
- DGEMM
- STREAM
- PTRANS (Parallel Matrix Transpose)
- RandomAccess
- FFT
- Communications Latency
- Communications Bandwidth
HPC Challenge v1.x Kernels
HPL (1 of 3)

• HPL (High Performance Linpack)
  – Implementation of the Linpack TPP (Toward Peak Performance) benchmark
  – Measures the floating point rate of execution for solving a linear system of equations

• HPL solves a linear system of equations of order n:

\[ Ax = b; \quad A \in \mathbb{R}^{n \times n}; \quad x, b \in \mathbb{R}^{n} \]

• by computing an LU factorization with row partial pivoting of the n by n+1 coefficient matrix:

\[ P[A, b] = [[L, U], y]. \]

• Since the row pivoting (represented by the permutation matrix P) and the lower triangular factor L are applied to b as the factorization progresses, the solution x is obtained in one step by solving the upper triangular system:

\[ Ux = y. \]
• The lower triangular matrix L is left unpivoted and the array of pivots is not returned.
• The operation counts are
  – Factorization phase — \((\frac{2}{3}n^3 - \frac{1}{2}n^2)\)
  – Solve phase — \((2n^2)\)
• Correctness is ascertained by calculating the scaled residuals where \(\epsilon\) is machine precision for 64-bit floating-point values and \(n\) is the size of the problem

\[
\frac{\|Ax - b\|_\infty}{\epsilon \|A\|_1 n},
\frac{\|Ax - b\|_\infty}{\epsilon \|A\|_1 \|x\|_1}, \text{ and } \frac{\|Ax - b\|_\infty}{\epsilon \|A\|_\infty \|x\|_\infty},
\]
HPC Challenge v1.x Kernels
HPL (3 of 3)

- **Scalability**
  - Assume memory available in the entire system is linearly proportional to the number of processors
  - HPL is dominated by CPU “costs”
    - Computation complexity — $O(n^3)$
    - Communication complexity — $O(n^2)$
  - It can be shown that the rate of execution (flop/s - $r$) for HPL is proportional to the number of processors ($P$)
    $$r_{HPL} \propto P$$
  - It can also be shown that the time ($t$) to run HPL is proportional to the square root of the number of processors
    $$t_{HPL} \propto \sqrt{P}$$

More at http://www.netlib.org/benchmark/hpl/
HPC Challenge v1.x Kernels
DGEMM

- DGEMM measures the floating point rate of execution of double precision real matrix-matrix multiplication
- The exact operation performed is:
  \[ C \leftarrow \beta C + \alpha AB \]
  where:
  \[ A, B, C \in \mathbb{R}^{n \times n}; \quad \alpha, \beta \in \mathbb{R}^n. \]

- The operation count is — \( (2n^3) \)
- Correctness is ascertained by calculating the scaled residual:
  \[ \|C - \hat{C}\|/(\varepsilon n \|C\|_F) \]
  (\( \hat{C} \) is a result of a reference implementation of the multiplication)
HPC Challenge v1.x Kernels
STREAM (1 of 2)

- STREAM is a simple benchmark program that measures sustainable memory bandwidth (in Gbyte/s) and the corresponding computation rate for four simple vector kernels:

  - COPY: \( c \leftarrow a \)
  - SCALE: \( b \leftarrow \alpha \cdot c \)
  - ADD: \( c \leftarrow a + b \)
  - TRIAD: \( a \leftarrow b + \alpha \cdot c \)

  where:

  \[ a, b, c \in \mathbb{R}^m; \quad \alpha \in \mathbb{R}. \]

- HPC Challenge Benchmarks are intended to operate on large data objects
  - Object size is determined at runtime which contrasts with the original version of the STREAM benchmark which uses static storage (determined at compile time) and size
  - The original benchmark gives the compiler more information (and control) over data alignment, loop trip counts, etc.
The benchmark measures Gbyte/s and the amount of data transferred is:
- Copy — (2m)
- Scale — (2m)
- Add — (3m)
- Triad — (3m)

Correctness is ascertained by calculating the norm of the difference between reference and computed vectors:

$$\|x - \hat{x}\|$$

The STREAM run rules require that the data dependency chain implied by the sequence of operations be maintained:
1. Copy
2. Scale
3. Add
4. Triad

PTRANS (parallel matrix transpose) exercises the communications where pairs of processors exchange large messages simultaneously.

It is a useful test of the total communications capacity of the system interconnect.

The performed operation sets a random \( n \times n \) matrix to a sum of its transpose with another random matrix:

\[
A \leftarrow A^T + B
\]

where:

\[
A, B \in \mathbb{R}^{n \times n}.
\]

The data transfer rate (in Gbyte/s) is calculated by dividing the size of \( n^2 \) matrix entries by the time it took to perform the transpose.

Correctness is ascertained by calculating the scaled residual:

\[
\|A - \hat{A}\|/(\varepsilon n)
\]

RandomAccess measures the rate of integer updates to random memory locations measured by the metric Giga-Updates per Second (GUPS)

The operation being performed on an integer array of size m is:

\[ x \leftarrow f(x) \]

\[ f : x \mapsto (x \oplus a_i); \quad a_i \text{ pseudo-random sequence} \]

where:

\[ f : \mathbb{Z}^m \rightarrow \mathbb{Z}^m; \quad x \in \mathbb{Z}^m. \]

The operation count is (4m) and since all the operations are in integral values over GF(2) field they can be checked exactly with a reference implementation.

The verification procedure allows 1% of the operations to be incorrect (skipped or due to data race conditions) which allows loosening concurrent memory update semantics on shared memory architectures.
HPC Challenge v1.x Kernels
RandomAccess (2 of 2)

- **Scalability**
  - Assume memory available in the entire system is linearly proportional to the number of processors
  - Global RandomAccess is communications-limited on distributed memory multiprocessors
  - Depending on the capability of the architecture Global RandomAccess may be scalable with rate \((r)\)
    1. Proportional to the number of processors \((P)\)
       \[ r_{RA} \propto P \]
    2. Independent of the number of processors \((P)\)
       \[ r_{RA} \propto 1 \]
    3. Inversely proportional to the number of processors \((P)\)
       (scaling decreases as the number of processors increases)
       \[ r_{RA} \propto \frac{1}{P} \]

• FFT measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT) of size m measured in Gflop/s:

\[ Z_k \leftarrow \sum_{j}^{m} z_j e^{-\frac{2\pi i j k}{m}}; \quad 1 \leq k \leq m \]

where:

\[ z, Z \in \mathbb{C}^m. \]

• The operation count for the calculation is \((5m\log_2 m)\)

• Correctness is ascertained by calculating the residual:

\[ \frac{\| x - \hat{x} \|}{(\varepsilon \log m)} \]

where \( \hat{x} \) is the result of applying a reference implementation of inverse transform to the outcome of the benchmarked code

– With infinite-precision arithmetic — the residual should be zero

More at http://www.ffte.jp/
The latency and bandwidth benchmark measures two different communication patterns:

- Single-process-pair latency and bandwidth
- Parallel all-processes-in-a-ring latency and bandwidth

For *Single-process-pair latency and bandwidth* ping-pong communication is used on a pair of processes:

- Several different pairs of processes are used and the maximal latency and minimal bandwidth over all pairs is reported
- While the ping-pong benchmark is executed on one process pair all other processes are waiting in a blocking receive
- To limit the total benchmark time to 30 sec — only a subset of the set of possible pairs is used
- The communication is implemented with MPI standard blocking send and receive.
For *Parallel all-processes-in-a-ring latency and bandwidth* communications

- All processes are arranged in a ring topology
- Each process sends and receives a message from its left and its right neighbor in parallel
- Two types of rings are used
  - A naturally ordered ring (i.e., ordered by the process ranks in MPI COMM WORLD)
  - The geometric mean of the bandwidth of ten different randomly chosen process orderings in the ring
- The communication is implemented with
  - MPI standard non-blocking receive and send
  - Two calls to MPI Sendrecv for both directions in the ring
  - Always the fastest of both measurements are used
- Bandwidth per process is defined as total amount of message data divided by the number of processes and the maximal time needed in all processes
- This benchmark is based on patterns studied in the effective bandwidth communication benchmark (*b_eff*)
• Message lengths
  – 8 byte
  – 2,000,000 bytes

• The major results reported by this benchmark are:
  – Maximal ping pong latency
  – Average latency of parallel communication in randomly ordered rings
  – Minimal ping pong bandwidth
  – Bandwidth per process in the naturally ordered ring
  – Average bandwidth per process in randomly ordered rings.

• Additionally results reported by this benchmark are:
  – Latency of the naturally ordered ring
  – Minimum, maximum, and average of the ping-pong latency and bandwidth
Communications Bandwidth and Latency benchmarks model

- Ring based — the communication behavior of multi-dimensional domain-decomposition applications
- Natural ring — the message transfer pattern of a regular grid based application
  - Only in the first dimension
  - Adequate ranking of the processes is assumed
- Random ring — the communication pattern of unstructured grid based applications

More at http://www.hlrs.de/organization/par/services/models/mpi/b_eff/
A Deep Dive into RandomAccess

RandomAccess may be the least familiar of the HPC Challenge Benchmark suite kernels...
GUPS (Giga UPdates Per Second)  
Characteristics of the Metric

- **GUPS (Giga UPdates per Second)**
  - A measurement that profiles the memory architecture of a system
  - A measure of performance similar to MFLOPS
- The HPCS HPCchallenge RandomAccess benchmark exercises the GUPS capability of a system like the LINPACK benchmark is intended to exercise the MFLOPS capability of a computer
- In each case, we would expect these benchmarks to achieve close to the "peak" capability of the memory system
- The extent of the similarities between RandomAccess and LINPACK are limited to both benchmarks attempting to calculate a peak system capabilities
  - RandomAccess is a memory benchmark and not a computational benchmark like LINPACK
- We are interested in the GUPS performance of entire systems and system subcomponents
  - The GUPS rating of a distributed memory multiprocessor
  - The GUPS rating of an SMP node
  - The GUPS rating of a single processor
- While there is typically a strict scaling of MFLOPS to processor count, a similar phenomenon may not always occur for GUPS
Calculating GUPS

- **Calculating GUPS**
  - Identify the number of memory locations that can be randomly updated in one second
  - Divide by 1 billion (1e9)
- “Randomly” means that there is little relationship between one address to be updated and the next — except that they occur in the space of ½ the total system memory
- An update is a read-modify-write operation on a table of 64-bit words
  - An address is generated
  - The value at that address is to be read from memory
  - The value is to be modified by an integer operation (add, and, or, xor) with a literal value
  - The new value is written back to memory
GUPS Rules
Memory and Error Rate

• Memory
  – Select the memory size to be the power of two such that $\frac{1}{4} \leq 2^m \leq \frac{1}{2}$ of the total memory
  – Each CPU operates on its own address stream
  – The single table may be distributed among nodes
  – The distribution of memory to nodes is left to the implementer
    ▪ A uniform data distribution may help balance the workload
    ▪ A non-uniform data distribution may simplify the calculations that identify processor location by eliminating the requirement for integer divides

• Error rate
  – A small (less than 1%) percentage of missed updates are permitted
GUPS Rules
Look Ahead and Stored Updates

• When measuring GUPS on a distributed memory multiprocessor system — define constraints
  – How far in the random address stream each node is permitted to "look ahead"
  – The number of update messages that can be stored before processing to permit multi-level parallelism
• For the purpose of measuring GUPS, each “node” is permitted to
  – Look ahead no more than 1024 random address stream samples
  – Store the same number of update messages before processing

• The limits on “look ahead” and “stored updates” are being implemented to assure that the benchmark meets the intent to profile memory architecture and not induce significant artificial data locality
RandomAccess Text Definition

RandomAccess is Benchmark #0 from the DARPA HPCS Discrete Math Benchmarks
Contact Robert Lucas (rflucas@isi.edu) or David Koester (dkoester@mitre.org) for further information

- Let $T$ be a table of size $2^n$ filled with random 64-bit integers
- Let $\{A_i\}$ be a stream of 64-bit integers of length $2^{n+2}$ generated by the
  primitive polynomial over $\text{GF}(2)$, $X^{63} + X^3 + X+1$
  - $\text{GF}(2)$ (Galois Field of order 2)
  - The elements of $\text{GF}(2)$ can be represented using the integers 0 and 1,
    i.e., binary operands
- For each $a_i$, set $T[a_i <63, 64-n>] = T[a_i <63, 64-n>] + a_i$
  - $+$ denotes addition in $\text{GF}(2)$ i.e. bit-wise exclusive “or” ($\oplus$)
  - $a_ij, i>$ denotes the sequence of bits within $a_i$
    e.g. $<63, 64-n>$ are the highest $n$ bits

- Parameters
  - $n$ is the largest power of 2 that is less than or equal to half of main memory
- Acceptable error — 1%
  - This flexibility would generally be used to allow non-coherent parallel operations
- Look ahead and storage before processing on distributed memory multi-processor systems
  - limited to 1024 per “node”

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>$p \oplus q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Bit-Level Exclusive Or $\oplus$

The Commutative and Associative nature of $\oplus$ allows processing in any order
Sequential RandomAccess Implementation

The expected value of the number of accesses per memory location \( T[k] \)
\[
E[T[k]] = \frac{2^{n+2}}{2^n} = 4
\]

Define Addresses
Sequences of bits within \( a_i \)

Data Stream
\( \{A_i\} \)
Length \( 2^{n+2} \)

Data-Driven Memory Access
\( a_i \) 64 bits

The Commutative and Associative nature of \( \oplus \) allows processing in any order

Acceptable Error — 1%
Look ahead and Storage before processing —1024 per “processor”
Global Address Space (GAS) G-RandomAccess Implementation

Tables

\[ T[k] \]

The expected value of the number of accesses per memory location \( T[k] \)

\[ E[T[k]] = \frac{2^{n+2}}{2^n} = 4 \]

Define Addresses
Sequences of bits within \( a_i \)

\[ k = [a_i <63, 64-n>] \]

Highest \( n \) bits

Data Stream

\( \{A_i\} \)

Length \( 2^{n+2} \)

Data-Driven Memory Access

\[ a_i \]

64 bits

Acceptable Error — 1%
Look ahead — 1024 per “sub-stream”
Storage before processing — 1024 per “processor”
### Distributed Memory G-RandomAccess Implementation — $p = 2^m$

#### Table

<table>
<thead>
<tr>
<th>“Processor”</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
</tr>
</tbody>
</table>

#### Table Size - $2^n$

1/2 Global Memory

#### Define Addresses

Sequences of bits within $a_i$

$q = a_i <(63, (64 - \log_2(p))>$

Processor Number

$k = [a_i <(63 - \log_2(p)), (64 - n)]>$

Local Offset

#### Data Stream

$\{A_i\}$

Data-Driven Memory Access

Length $2^{n+2}$

$\begin{align*}
\text{Data Stream} &:
\{A_i\} \\
\text{Length} &: 2^{n+2}
\end{align*}$

#### For $p$ “processors”

Calculate $a_i$ to $a_{i+p}$ simultaneously

Acceptable Error — 1%

Look ahead — 1024 per “sub-stream”

Storage before processing — 1024 per “processor”
Distributed Memory G-RandomAccess Implementation — $p \neq 2^m$

**Define Addresses**
Sequences of bits within $a_i$

**Data Stream**
$\{A_i\}$

**Data-Driven Memory Access**

- **Table $T$**
  - Table Size - $2^n$
  - 1/2 Global Memory

- **Table Entry**
  - $k_0[q]$ is the Global Offset corresponding to $T[q][0]$
  - $k = d - k_0[q]$ Local Offset
  - $d = a_i <63, 64-n>$ Highest n bits

- **Integer Divide & Conditional Processor Number**
  - $q = f(d)$

- **For $p$ “processors”**
  - Calculate $a_i$ to $a_{i+p}$ simultaneously

- **Acceptable Error — 1%**
  - Look ahead — 1024 per “sub-stream”
  - Storage before processing — 1024 per “processor”

- **Table $T$**
  - Length $2^{n+2}$
  - $a_i$ 64 bits

- **“Processor”**
  - Table Size - $2^n$
  - $k_0[q]$ to $T[q][k]$
  - $p-1$
HPC Challenge v1.x Benchmark Suite Outline

- Introduction
- Motivations
  - HPCS
  - Performance Characterization
- Component Kernels
- HPC Challenge Awards
- Unified Benchmark Framework
- Rules
  - Running HPC Challenge
  - Optimizations
  - Etiquette
- Performance Data
  - Available Benchmark Data
  - Kiviat Charts
- Hands-on Demonstrations/Exercises
  - Installing the HPC Challenge v1.x Benchmark Unified Benchmark Framework
  - Running the HPC Challenge v1.x Benchmark suite
- Summary/Conclusions
The First Annual HPC Challenge Award Competition
Sponsors — DARPA HPCS, DOE, NSF, and HPCWire
http://www.hpcchallenge.org
Goal: to focus the HPC community’s attention on a broad set of HPC hardware and HPC software capabilities that are necessary to effectively use HPC systems.
The core of the HPC Challenge Award Competition is the HPC Challenge benchmark suite
The competition will focus on four of the most challenging benchmarks in the suite:
- Global HPL
- Global RandomAccess
- EP STREAM (Triad) per system
- Global FFT

Prizes sponsored by HPCWire
HPC Challenge Award Competition
Award Classes

- **Class 1: Best Performance (4 awards)**
  - Best performance on a base or optimized run submitted to the HPC Challenge website
    - Global HPL
    - Global RandomAccess
    - EP STREAM (Triad) per system
    - Global FFT
  - The prize will be $500 plus a certificate for the best of each benchmark

- **Class 2: Most Productivity**
  - Most "elegant" implementation of one or more of the HPC Challenge benchmarks listed above
  - This award would be weighted 50% on performance and 50% on code elegance, clarity, and size as determined by an evaluation committee
  - For this award, the implementer must submit by October 15th, 2005, a short description of:
    - The implementation,
    - The performance achieved,
    - Lines-of-code,
    - The actual source code of their implementation.
  - The evaluation committee will select a set of finalists who will be invited to give a short presentation at the HPC Challenge Award BOF at SC|05 that will be judged by the evaluation committee to select the winner
  - The prize will be $1500 plus a certificate for this award and may be split among the "best" entries.

Awards will be presented at the HPC Challenge Award BOF at SC|05
Tuesday 15 November 2005 at noon

Prizes sponsored by HPCWire
HPC Challenge Award Competition

Award Classes

• Class 1: Best Performance (4 awards)
  - Best performance on a base or optimized run submitted to the HPC Challenge website
    ▪ Global HPL
    ▪ Global RandomAccess
    ▪ EP STREAM (Triad) per system
    ▪ Global FFT
  - The prize will be $500 plus a certificate for the best of each benchmark

• Class 2: Most Productivity
  - Most "elegant" implementation of one or more of the HPC Challenge benchmarks listed above
  - This award would be weighted 50% on performance and 50% on code elegance, clarity, and size as determined by an evaluation committee
  - For this award, the implementer must submit by October 15th, 2005, a short description of:
    ▪ The implementation,
    ▪ The performance achieved,
    ▪ Lines-of-code,
    ▪ The actual source code of their implementation.
  - The evaluation committee will select a set of finalists who will be invited to give a short presentation at the HPC Challenge Award BOF at SC|05 that will be judged by the evaluation committee to select the winner
  - The prize will be $1500 plus a certificate for this award and may be split among the "best" entries.

Awards will be presented at the SC|05 HPC Challenge Award BOF

Tuesday 15 November 2005 at noon

Prizes sponsored by HPCWire
HPC Challenge Awards
Evaluation Committee

• David Bailey
  LBNL NERSC

• Jack Dongarra (Co-Chair)
  U of Tenn/ORNL

• Jeremy Kepner (Co-Chair)
  MIT Lincoln Lab

• David Koester
  MITRE

• Bob Lucas
  ISI

• Rusty Lusk
  Argonne National Lab

• Piotr Luszczek
  U of Tennessee

• John McCalpin
  IBM Austin

• Rolf Rabenseifner
  HLRS Stuttgart

• Daisuke Takahashi
  U of Tsukuba
HPC Challenge v1.x Benchmark Suite Outline

• Introduction
• Motivations
  – HPCS
  – Performance Characterization
• Component Kernels
• HPC Challenge Awards
• Unified Benchmark Framework
• Rules
  – Running HPC Challenge
  – Optimizations
  – Etiquette
• Performance Data
  – Available Benchmark Data
  – Kiviat Charts
• Hands-on Demonstrations/Exercises
  – Installing the HPC Challenge v1.x Benchmark Unified Benchmark Framework
  – Running the HPC Challenge v1.x Benchmark suite
• Summary/Conclusions
Unified HPCC Framework

• HPCC unifies a number of existing (and well known) codes in one consistent framework
• A single executable is built to run all of the components
  – Easy interaction with batch queues
  – All codes are run under the same OS conditions – just as an application would
    ▪ No special mode (page size, etc.) for just one test (say Linpack benchmark)
    ▪ Each test may still have its own set of compiler flags
      □ Changing compiler flags in the same executable may inhibit inter-procedural optimization
• Why not use a script and a separate executable for each test?
  – Lack of enforced integration between components
    ▪ Ensure reasonable data sizes
    ▪ Either all tests pass and produce meaningful results or failure is reported
  – Running a single component of HPCC for testing is easy enough
Baseline MPI-1 Implementation

- Publicly available code is required for base submission
  1. Requires C compiler, MPI 1.1, and BLAS
  2. Source code cannot be changed for submission run
  3. Linked libraries have to be publicly available
  4. The code contains optimizations for contemporary hardware systems
  5. Algorithmic variants provided for performance portability

- This to mimic legacy applications’ performance
  1. Reasonable software dependences
  2. Code cannot be changed due to complexity and maintenance cost
  3. Relies on publicly available software
  4. Some optimization has been done on various platforms
  5. Conditional compilation and runtime algorithm selection for performance tuning

Baseline code has over 10k SLOC — there must more productive way of coding
Optimized HPCC Submissions

- Timed portions of the code may be replaced with optimized code
- Verification code still has to pass
  - Must use the same data layout or pay the cost of redistribution
  - Must use sufficient precision to pass residual checks
- Allows to use new parallel programming technologies
  - New paradigms, e.g. one-sided communication of MPI-2:
    MPI_Win_create(...);
    MPI_Get(...);
    MPI_Put(...);
    MPI_Win_fence(...);
  - New languages, e.g. UPC:
    - shared pointers
    - upc_memput()
- Code for optimized portion may be proprietary but needs to use publicly available libraries
- Optimizations need to be described but not necessarily in detail – possible use in application tuning
- Attempting to capture: invested effort per flop rate gain
  - Hence the need for baseline submission
- There can be more than one optimized submission for a single base submission (if a given architecture allows for many optimizations)
HPC Challenge v1.x Benchmark Suite
Outline

• Introduction
• Motivations
  – HPCS
  – Performance Characterization
• Component Kernels
• HPC Challenge Awards
• Unified Benchmark Framework
• Rules
  – Running HPC Challenge
  – Optimizations
  – Etiquette
• Performance Data
  – Available Benchmark Data
  – Kiviat Charts
• Hands-on Demonstrations/Exercises
  – Installing the HPC Challenge v1.x Benchmark Unified Benchmark Framework
  – Running the HPC Challenge v1.x Benchmark suite
• Summary/Conclusions
Running HPC Challenge

• To enter data into the HPC Challenge archive — you must submit a baseline run for each HPC system
  - Only complete benchmark output may be submitted — partial results will not be accepted

• You may also submit an optimized run for each HPC system
  - Again — only complete benchmark output may be submitted
The following optimizations are allowed in the baseline runs

- **Compile and load options**
  - Compiler or loader flags which are supported and documented by the supplier are allowed
  - These include porting, optimization, and preprocessor invocation

- **Libraries**
  - Linking to optimized versions of the following libraries is allowed
    - BLAS
    - MPI
  - Acceptable use of such libraries is subject to the following rules:
    - All libraries used shall be disclosed with the results submission. Each library shall be identified by library name, revision, and source (supplier). Libraries which are not generally available are not permitted unless they are made available by the reporting organization within 6 months
    - Calls to library subroutines should have equivalent functionality to that in the released benchmark code. Code modifications to accommodate various library call formats are not allowed
The following routines may have optimized versions substituted for the baseline codes — the input and output specification must be preserved:

- **HPL**
  - `pdgesv()`
  - `pdtrsv()`
- **DGEMM**
  - no changes are allowed
- **PTRANS**
  - `pdtrans()`
- **STREAM**
  - `Copy()`
  - `Scale()`
  - `Add()`
  - `Triad()`
- **RandomAccess**
  - `MPIRandomAccessUpdate()`
  - `RandomAccessUpdate()`
- **FFT** (all functions are compatible with FFTW 2.1.5 [11, 12])
  - `fftw malloc()`, `fftw free()`, `fftw one()`, `fftw mpi()`
  - `fftw create plan()`, `fftw destroy plan()`
  - `fftw mpi create plan()`, `fftw mpi local sizes()`
  - `fftw mpi destroy plan()`
- **b eff** — alternative MPI routines might be used for communication
  - Only standard MPI calls are to be performed
  - Only MPI libraries that are widely available on the tested system may be used
Rules
Optimizations — Limitations

• Calculations must be performed in 64-bit precision or the equivalent
  – Codes with limited calculation accuracy are not permitted
• All algorithm modifications must be fully disclosed and are subject to review by the HPC Challenge Committee
  – Passing the verification test is a necessary condition for such an approval
  – The replacement algorithm must be as robust as the baseline algorithm
    ▪ For example — the Strassen Algorithm may not be used for the matrix multiply in the HPL benchmark, as it changes the operation count of the algorithm
• Any modification of the code or input data sets — which utilizes knowledge of the solution or of the verification test — is not permitted
• Any code modification to circumvent the actual computation is not permitted
Etiquette

• The HPC Challenge Benchmark suite has been designed to permit academic style usage for comparing
  – Technologies
  – Architectures
  – Programming models
• There is an overt attempt to keep HPC Challenge significantly different than “commercialized” benchmark suites
  – Vendors and users can submit results
  – System “cost/price” is not included intentionally
  – No “composite” benchmark metric

• Be cool about comparisons!
• While we can not enforce any rule to limit comparisons observe rules of
  – Academic honesty
  – Good taste
HPC Challenge v1.x Benchmark Suite
Outline

- Introduction
- Motivations
  - HPCS
  - Performance Characterization
- Component Kernels
- HPC Challenge Awards
- Unified Benchmark Framework
- Rules
  - Running HPC Challenge
  - Optimizations
  - Etiquette
- Performance Data
  - Available Benchmark Data
  - Kiviat Charts
- Hands-on Demonstrations/Exercises
  - Installing the HPC Challenge v1.x Benchmark Unified Benchmark Framework
  - Running the HPC Challenge v1.x Benchmark suite
- Summary/Conclusions
HPC Challenge Benchmark Suite

http://icl.cs.utk.edu/hpcc/

The HPC Challenge benchmark consists of basically 7 tests:

1. **HPL** - the Linpack TPP benchmark which measures the floating point rate of execution for solving a linear system of equations.

2. **DGEMM** - measures the floating point rate of execution of double precision real matrix-matrix multiplication.

3. **STREAM** - a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.

4. **PTRANS** (parallel matrix transpose) - exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.

5. **RandomAccess** - measures the rate of integer random updates of memory (GUPS).

6. **FFTE** - measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).

7. Communication bandwidth and latency - a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns; based on b_eff (effective bandwidth benchmark).
TOP500 and HPCC Data Analysis

- **TOP500**
  - Performance is represented by only a single metric
  - Data is available for an extended time period (1993-2005)

- **Problem:**
  There can only be one “winner”

- **Additional metrics and statistics**
  - Count (single) vendor systems on each list
  - Count total flops on each list per vendor
  - Use external metrics: price, ownership cost, power, …
  - Focus on growth trends over time

- **HPCC**
  - Performance is represented by multiple single metrics
  - Benchmark is new — so data is available for a limited time period (2003-2005)

- **Problem:**
  There cannot be one “winner”

- **We avoid “composite” benchmarks**
  - Perform trend analysis
    - HPCC can be used to show complicated kernel/architecture performance characterizations
  - Select some numbers for comparison
  - Use of kiviat charts
    - Best when showing the differences due to a single independent “variable”

- **Over time — also focus on growth trends**
## HPCC Submissions
Baseline and Optimized Results

**80 Systems As of 1 November 2005**

To highlight the HPCC Challenge Class 1 Awards which will be present at SC05 we are not displaying the new submissions to the HPCC Challenge from November 1st until the Awards session at SC05. The Awards session is on November 15th at noon in room 602-604 of the conference center. We are of course still accepting submissions to the HPCC Challenge Class 1 Awards up until Sunday November 13th, 2005.

### Condensed Results - Base and Optimized Runs - 80 Systems - Generated on Tue Nov 1 14:30:05 2005

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>HPC Challenge v1.x Benchmarks</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Abips Conquest cluster AMD Opteron</td>
<td>base</td>
<td>0.2526110</td>
<td>3.2471</td>
<td>208.525</td>
<td>1.6291</td>
<td>0.03627</td>
<td>23.66</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clustervision Beadie AMD Opteron</td>
<td>base</td>
<td>0.1037640</td>
<td>0.8159</td>
<td>0.0002350</td>
<td>2.15</td>
<td>106.951</td>
<td>3.8422</td>
<td>4.19492</td>
<td>0.02868</td>
<td>33.23</td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>base</td>
<td>0.5215600</td>
<td>3.2338</td>
<td>959.334</td>
<td>14.9896</td>
<td>0.94974</td>
<td>20.34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>base</td>
<td>0.5777790</td>
<td>30.4313</td>
<td>899.446</td>
<td>14.9761</td>
<td>1.03291</td>
<td>20.82</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>base</td>
<td>1.0609700</td>
<td>2.4603</td>
<td>1029.519</td>
<td>0.4959</td>
<td>0.03914</td>
<td>20.12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3E Alpha 21164</td>
<td>base</td>
<td>0.0431695</td>
<td>10.2765</td>
<td>523.242</td>
<td>0.5165</td>
<td>0.03174</td>
<td>12.09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>base</td>
<td>2.8818200</td>
<td>97.4076</td>
<td>3783.404</td>
<td>14.9142</td>
<td>0.42899</td>
<td>22.27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>opt</td>
<td>2.3678200</td>
<td>96.1372</td>
<td>5478.732</td>
<td>21.7410</td>
<td>0.43028</td>
<td>22.64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>opt</td>
<td>0.5788780</td>
<td>31.0723</td>
<td>1306.080</td>
<td>21.7680</td>
<td>1.00956</td>
<td>21.16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>opt</td>
<td>1.0094200</td>
<td>39.5232</td>
<td>1855.664</td>
<td>14.9731</td>
<td>0.70057</td>
<td>20.15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>opt</td>
<td>1.1020200</td>
<td>39.3824</td>
<td>2697.260</td>
<td>21.7521</td>
<td>0.00350</td>
<td>20.85</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# HPCC Submissions Baseline Results

## 74 Systems
As of 1 November 2005

To highlight the HPC Challenge Class 1 Awards which will be present at SC05 we are not displaying the new submissions to the HPC Challenge from November 1st until the Awards session at SC05. The Awards session is on November 15th at noon in room 602-504 of the conference center. We are of course still accepting submissions to the HPC Challenge Class 1 Awards up until Sunday November 13th, 2005.

### Condensed Results - Base Runs Only - 74 Systems - Generated on Tue Nov 1 14:51:26 2005

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>System - Processor - Speed - Count - Threads - Processes</td>
<td>TFlop/s</td>
<td>GB/s</td>
<td>Gop/s</td>
<td>GFlop/s</td>
<td>GB/s</td>
<td>GFlop/s</td>
<td>GB/s</td>
<td>GFlop/s</td>
<td>GB/s</td>
</tr>
<tr>
<td>Abpia Conquest cluster AMD Opteron 1.4GHz 128 1 128</td>
<td>0.2525110</td>
<td>3.2471</td>
<td>206.525</td>
<td>1.6291</td>
<td>0.03627</td>
<td>23.66</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clustervision BV Beestle AMD Opteron 2.4GHz 32 1 32</td>
<td>0.103740</td>
<td>0.8159</td>
<td>0.0002350</td>
<td>2.1470</td>
<td>106.951</td>
<td>1.3422</td>
<td>4.19493</td>
<td>0.02648</td>
<td>53.23</td>
</tr>
<tr>
<td>Cray X1 MSP 0.6GHz 64 1 64</td>
<td>0.5215600</td>
<td>3.2208</td>
<td>959.334</td>
<td>14.5856</td>
<td>0.94074</td>
<td>20.34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP 0.6GHz 60 1 60</td>
<td>0.5777750</td>
<td>30.4313</td>
<td>888.446</td>
<td>14.9741</td>
<td>1.03269</td>
<td>20.83</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP 0.6GHz 120 1 120</td>
<td>1.0509700</td>
<td>2.4600</td>
<td>1019.819</td>
<td>0.4960</td>
<td>0.62014</td>
<td>20.12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3E Alpha 21164 0.6GHz 1024 1 1024</td>
<td>0.0466165</td>
<td>10.2755</td>
<td>529.242</td>
<td>0.5168</td>
<td>0.63174</td>
<td>12.09</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP 0.6GHz 252 1 252</td>
<td>2.3843300</td>
<td>9.4076</td>
<td>3758.404</td>
<td>14.9143</td>
<td>0.42859</td>
<td>22.27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP 0.6GHz 124 1 124</td>
<td>1.2054200</td>
<td>39.5252</td>
<td>1056.064</td>
<td>14.5731</td>
<td>0.70857</td>
<td>20.15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP 0.6GHz 60 1 60</td>
<td>0.5007400</td>
<td>1.0341</td>
<td>894.114</td>
<td>14.9019</td>
<td>10.91520</td>
<td>11.8779</td>
<td>14.66</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray T3E Alpha 21164 0.675GHz 512 1 512</td>
<td>0.2231810</td>
<td>9.7741</td>
<td>272.186</td>
<td>0.3318</td>
<td>0.66077</td>
<td>8.14</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
To highlight the HPC Challenge Class 1 Awards which will be present at SC05 we are not displaying the new submissions to the HPC Challenge from November 1st until the Awards session at SC05. The Awards session is on November 15th at noon in room 602-604 of the conference center. We are of course still accepting submissions to the HPC Challenge Class 1 Awards up until Sunday November 13th, 2005.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MA/PA/PS/PC/TH/PR/CM/CS/IC/IA/SD</td>
<td>TFlop/s</td>
<td>GB/s</td>
<td>Gop/s</td>
<td>GFlop/s</td>
<td>GB/s</td>
<td>GB/s</td>
<td>GFlop/s</td>
<td>GB/s</td>
<td>usec</td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>0.8GHz 252 1 252</td>
<td>2.368</td>
<td>96.1</td>
<td>5479</td>
<td>21.741</td>
<td>0.4303</td>
<td>22.64</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>0.8GHz 60 1 60</td>
<td>0.579</td>
<td>21.1</td>
<td>1306</td>
<td>21.768</td>
<td>1.0099</td>
<td>21.15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>0.0GHz 124 1 124</td>
<td>1.102</td>
<td>30.4</td>
<td>2507</td>
<td>21.752</td>
<td>0.0039</td>
<td>20.05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1 MSP</td>
<td>0.8GHz 124 1 124</td>
<td>1.182</td>
<td>35.4</td>
<td>2507</td>
<td>21.752</td>
<td>0.0039</td>
<td>20.05</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray Mfog® X1E</td>
<td>1.13GHz 248 1 248</td>
<td>2.289</td>
<td>66.0</td>
<td>1.655</td>
<td>1.00</td>
<td>3291</td>
<td>13.229</td>
<td>13.56</td>
<td>0.2989</td>
<td>14.59</td>
</tr>
<tr>
<td>IBM Blue Gene/L PowerPC 440</td>
<td>0.7GHz 1024 1 1024</td>
<td>1.420</td>
<td>28.0</td>
<td>0.130</td>
<td>49.93</td>
<td>0.043</td>
<td>2.47</td>
<td>0.0346</td>
<td>4.03</td>
<td></td>
</tr>
</tbody>
</table>
### System Information

<table>
<thead>
<tr>
<th>System - Processor - Speed - Count - Threads - Processes</th>
<th>Run</th>
<th>G-HPL</th>
<th>G-PTRANS</th>
<th>G-Random Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>MA/PT/PS/PC/TH/PR/CM/CS/IC/IA/SD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray XT3 AMD Opteron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray mfeg8 X1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray XD1 AMD Opteron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1E X1E MSP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray XT3 AMD Opteron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray XT3 AMD Opteron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray xt3 AMD Opteron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray X1E</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cray XT3 AMD Opteron</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Additional “cool” features will be discussed in the conference hands-on session!**

**Lines Depict Relative Performance**
HPC Challenge Benchmark Suite
Selected Results

1000+ Processor Systems
As of 1 November 2005

- 10 of 80 submissions have over 1,000 processors
  - 1008 – 5200 processors
Optimized G-RandomAccess

As of 1 November 2005

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray mfeg8 X1E</td>
<td>opt</td>
<td>3.3889</td>
<td>66.01</td>
<td>1.85475</td>
<td>-1</td>
<td>3280.9</td>
<td>13.229</td>
<td>13.564</td>
<td>0.29886</td>
<td>14.58</td>
</tr>
<tr>
<td>Cray X1E X1E MSP</td>
<td>base</td>
<td>3.1941</td>
<td>85.204</td>
<td>0.014868</td>
<td>15.54</td>
<td>2440</td>
<td>9.682</td>
<td>14.185</td>
<td>0.36024</td>
<td>14.93</td>
</tr>
</tbody>
</table>

- Optimized G-RandomAccess is an UPC code
  - ~125x improvement

Be sure to attend the SC05 HPC Challenge Award BOF
Tuesday 15 November 2005 at noon for new, record-setting results!!
**HPC Challenge Benchmark Suite**

**Top Performers**

As of 1 November 2005

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray XT3 AMD Opteron</td>
<td>2.4GHz</td>
<td>5200</td>
<td>1</td>
<td>5200</td>
<td>20.527</td>
<td>874.899</td>
<td>0.268583</td>
<td>644.73</td>
<td>26020.8</td>
<td>5.004</td>
<td>4.395</td>
<td>0.14682</td>
<td>25.8</td>
</tr>
<tr>
<td>Cray mpeg8 X1E</td>
<td>1.13GHz</td>
<td>248</td>
<td>1</td>
<td>248</td>
<td>3.3889</td>
<td>66.01</td>
<td>1.85475</td>
<td>-1</td>
<td>3280.9</td>
<td>13.229</td>
<td>13.564</td>
<td>0.29886</td>
<td>14.58</td>
</tr>
<tr>
<td>Cray XT3 AMD Opteron</td>
<td>2.6GHz</td>
<td>4096</td>
<td>1</td>
<td>4096</td>
<td>16.9752</td>
<td>302.979</td>
<td>0.533072</td>
<td>905.57</td>
<td>20656.5</td>
<td>5.043</td>
<td>4.782</td>
<td>0.16896</td>
<td>9.44</td>
</tr>
<tr>
<td>NEC SX-7</td>
<td>0.552GHz</td>
<td>32</td>
<td>16</td>
<td>2</td>
<td>0.2174</td>
<td>16.34</td>
<td>0.000178</td>
<td>1.34</td>
<td>984.3</td>
<td>492.161</td>
<td>140.636</td>
<td>8.14753</td>
<td>4.85</td>
</tr>
<tr>
<td>NEC SX-8/SX-8</td>
<td>2GHz</td>
<td>6</td>
<td>1</td>
<td>6</td>
<td>0.0918</td>
<td>25.183</td>
<td>0.000769</td>
<td>3.19</td>
<td>370.6</td>
<td>61.773</td>
<td>15.944</td>
<td>13.5473</td>
<td>3.02</td>
</tr>
<tr>
<td>IBM pSeries 655 Power 4+</td>
<td>1.7GHz</td>
<td>256</td>
<td>4</td>
<td>64</td>
<td>1.0744</td>
<td>23.721</td>
<td>0.005502</td>
<td>10.46</td>
<td>411.7</td>
<td>6.433</td>
<td>17.979</td>
<td>0.72395</td>
<td>8.34</td>
</tr>
<tr>
<td>PathScale Inc. AMD Opteron</td>
<td>2.6GHz</td>
<td>32</td>
<td>1</td>
<td>32</td>
<td>0.1258</td>
<td>6.719</td>
<td>0.030367</td>
<td>10.35</td>
<td>134.3</td>
<td>4.197</td>
<td>4.775</td>
<td>0.26531</td>
<td>1.31</td>
</tr>
</tbody>
</table>

- Machine size (number of processors) matters for global benchmarks
  - HPL, PTRANS, FFT, STREAM,
- G-RandomAccess is an optimized UPC code
- Node “size” matters for local benchmarks
  - STREAM, DGEMM
- Bandwidth and latency are dependent on
  - MPI and architecture
### HPC Challenge Benchmark Suite

#### Top Performers

As of 1 November 2005

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cray XT3 AMD Opteron 2.4GHz</td>
<td>20.527</td>
<td>874.899</td>
<td>0.268583</td>
<td>644.73</td>
<td>26020.8</td>
<td>5.004</td>
<td>4.395</td>
<td>0.14682</td>
<td>25.8</td>
</tr>
<tr>
<td>Cray mpeg8 X1E 1.13GHz</td>
<td>3.389</td>
<td>66.01</td>
<td>1.85475</td>
<td>-1</td>
<td>3280.9</td>
<td>13.229</td>
<td>13.564</td>
<td>0.29886</td>
<td>14.58</td>
</tr>
<tr>
<td>Cray XT3 AMD Opteron 2.6GHz</td>
<td>16.9752</td>
<td>302.979</td>
<td>0.533072</td>
<td>905.57</td>
<td>20656.5</td>
<td>5.043</td>
<td>4.782</td>
<td>0.16896</td>
<td>9.44</td>
</tr>
<tr>
<td>NEC SX-7 0.552GHz</td>
<td>0.2174</td>
<td>16.34</td>
<td>0.000178</td>
<td>1.34</td>
<td>984.3</td>
<td>492.161</td>
<td>140.636</td>
<td>8.14753</td>
<td>4.85</td>
</tr>
<tr>
<td>NEC SX-8/SX-8 2GHz</td>
<td>0.0918</td>
<td>25.183</td>
<td>0.000769</td>
<td>3.19</td>
<td>370.6</td>
<td>61.773</td>
<td>15.944</td>
<td>13.5473</td>
<td>3.02</td>
</tr>
<tr>
<td>IBM pSeries 655 Power 4+ 1.7GHz</td>
<td>1.0744</td>
<td>23.721</td>
<td>0.005502</td>
<td>10.46</td>
<td>411.7</td>
<td>6.433</td>
<td>17.979</td>
<td>0.72395</td>
<td>8.34</td>
</tr>
<tr>
<td>PathScale Inc. AMD Opteron 2.6GHz</td>
<td>0.1258</td>
<td>6.719</td>
<td>0.030367</td>
<td>10.35</td>
<td>134.3</td>
<td>4.197</td>
<td>4.775</td>
<td>0.26531</td>
<td>1.31</td>
</tr>
</tbody>
</table>

- **HPC Challenge Award Competition** will focus on four of the benchmarks in the suite:
  - Global HPL
  - Global RandomAccess
  - Global STREAM Triad (System aggregate)
  - Global FFT

Be sure to attend the SC|05 HPC Challenge Award BOF Tuesday 15 November 2005 at noon for new, record-setting results!!
The NEC SX-7 architecture can permit the definition of threads and processes to significantly enhance performance of the EP versions of the benchmark suite by allocating more powerful “nodes”

- EP-STREAM
- EP-DGEMM
## Top 10 Performance
### HPL

As of 1 November 2005

<table>
<thead>
<tr>
<th>Rank</th>
<th>Manufacturer</th>
<th>System</th>
<th>Processor Type</th>
<th>Processor Speed (GHz)</th>
<th>Procesor Count</th>
<th>MPI Processes</th>
<th>HPL (TFlop/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cray</td>
<td>Cray XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>5200</td>
<td>5200</td>
<td>20.53</td>
</tr>
<tr>
<td>2</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>4096</td>
<td>4096</td>
<td>16.98</td>
</tr>
<tr>
<td>3</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>3744</td>
<td>3744</td>
<td>14.70</td>
</tr>
<tr>
<td>4</td>
<td>NEC</td>
<td>NEC SX-8</td>
<td>NEC SX-8</td>
<td>2.00</td>
<td>576</td>
<td>576</td>
<td>8.01</td>
</tr>
<tr>
<td>5</td>
<td>SGI</td>
<td>Altix 3700 Bx2</td>
<td>Intel Itanium 2</td>
<td>1.60</td>
<td>1008</td>
<td>1008</td>
<td>5.14</td>
</tr>
<tr>
<td>6</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>1100</td>
<td>1100</td>
<td>4.78</td>
</tr>
<tr>
<td>7</td>
<td>Cray</td>
<td>mfg8</td>
<td>Cray X1E</td>
<td>1.13</td>
<td>248</td>
<td>248</td>
<td>3.39</td>
</tr>
<tr>
<td>8</td>
<td>Cray</td>
<td>Cray X1E</td>
<td>CrayX1E MSP</td>
<td>1.13</td>
<td>252</td>
<td>252</td>
<td>3.19</td>
</tr>
<tr>
<td>9</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.80</td>
<td>252</td>
<td>252</td>
<td>2.38</td>
</tr>
<tr>
<td>10</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.80</td>
<td>252</td>
<td>252</td>
<td>2.37</td>
</tr>
</tbody>
</table>

**HPC Challenge Awards Class 1**
### Top 10 Performance

**PTRANS**

As of 1 November 2005

<table>
<thead>
<tr>
<th>Rank</th>
<th>Manufacturer</th>
<th>System</th>
<th>Processor Type</th>
<th>Processor Speed (GHz)</th>
<th>Processor Count</th>
<th>MPI Processes</th>
<th>PTRANS (GB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cray</td>
<td>Cray XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>5200</td>
<td>5200</td>
<td>874.90</td>
</tr>
<tr>
<td>2</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>3744</td>
<td>3744</td>
<td>608.51</td>
</tr>
<tr>
<td>3</td>
<td>NEC</td>
<td>NEC SX-8</td>
<td>NEC SX-8</td>
<td>2.00</td>
<td>576</td>
<td>576</td>
<td>312.71</td>
</tr>
<tr>
<td>4</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>4096</td>
<td>4096</td>
<td>302.98</td>
</tr>
<tr>
<td>5</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>1100</td>
<td>1100</td>
<td>217.92</td>
</tr>
<tr>
<td>6</td>
<td>SGI</td>
<td>Altix 3700 Bx2</td>
<td>Intel Itanium 2</td>
<td>1.60</td>
<td>1008</td>
<td>1008</td>
<td>105.67</td>
</tr>
<tr>
<td>7</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.80</td>
<td>252</td>
<td>252</td>
<td>97.41</td>
</tr>
<tr>
<td>8</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.80</td>
<td>252</td>
<td>252</td>
<td>96.14</td>
</tr>
<tr>
<td>9</td>
<td>NEC</td>
<td>SX-6</td>
<td>NEC SX-6</td>
<td>0.50</td>
<td>192</td>
<td>192</td>
<td>92.97</td>
</tr>
<tr>
<td>10</td>
<td>Cray</td>
<td>Cray X1E</td>
<td>CrayX1E MSP</td>
<td>1.13</td>
<td>252</td>
<td>252</td>
<td>85.20</td>
</tr>
</tbody>
</table>
# Top 10 Performance

## G-RandomAccess

As of 1 November 2005

<table>
<thead>
<tr>
<th>Rank</th>
<th>Manufacturer</th>
<th>System</th>
<th>Processor Type</th>
<th>Processor Speed (GHz)</th>
<th>Processor Count</th>
<th>MPI Processes</th>
<th>Global RandomAccess (GUP/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cray</td>
<td>mfeg8</td>
<td>Cray X1E</td>
<td>1.13</td>
<td>248</td>
<td>248</td>
<td>1.85475</td>
</tr>
<tr>
<td>2</td>
<td>Rackable Systems</td>
<td>Emerald</td>
<td>AMD Opteron</td>
<td>2.20</td>
<td>256</td>
<td>512</td>
<td>0.55474</td>
</tr>
<tr>
<td>3</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>4096</td>
<td>4096</td>
<td>0.53307</td>
</tr>
<tr>
<td>4</td>
<td>IBM</td>
<td>Blue Gene</td>
<td>IBM PowerPC 440</td>
<td>0.70</td>
<td>2048</td>
<td>2048</td>
<td>0.45409</td>
</tr>
<tr>
<td>5</td>
<td>Rackable Systems</td>
<td>Emerald</td>
<td>AMD Opteron</td>
<td>2.20</td>
<td>128</td>
<td>256</td>
<td>0.42255</td>
</tr>
<tr>
<td>6</td>
<td>Rackable Systems</td>
<td>Emerald</td>
<td>AMD Opteron</td>
<td>2.20</td>
<td>64</td>
<td>128</td>
<td>0.30807</td>
</tr>
<tr>
<td>7</td>
<td>IBM</td>
<td>Blue Gene</td>
<td>IBM PowerPC 440</td>
<td>0.70</td>
<td>1024</td>
<td>1024</td>
<td>0.29962</td>
</tr>
<tr>
<td>8</td>
<td>Cray</td>
<td>Cray XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>5200</td>
<td>5200</td>
<td>0.26858</td>
</tr>
<tr>
<td>9</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>3744</td>
<td>3744</td>
<td>0.22030</td>
</tr>
<tr>
<td>10</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>1100</td>
<td>1100</td>
<td>0.13700</td>
</tr>
</tbody>
</table>

- **HPC Challenge Awards Class 1**
# Top 10 Performance

**G-FFTE**

As of 1 November 2005

<table>
<thead>
<tr>
<th>Rank</th>
<th>Manufacturer</th>
<th>System</th>
<th>Processor Type</th>
<th>Processor Speed (GHz)</th>
<th>Processor Count</th>
<th>MPI Processes</th>
<th>Global FFT (GFlop/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>4096</td>
<td>4096</td>
<td>905.57</td>
</tr>
<tr>
<td>2</td>
<td>Cray</td>
<td>Cray XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>5200</td>
<td>5200</td>
<td>644.73</td>
</tr>
<tr>
<td>3</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>3744</td>
<td>3744</td>
<td>417.17</td>
</tr>
<tr>
<td>4</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>1100</td>
<td>1100</td>
<td>266.66</td>
</tr>
<tr>
<td>5</td>
<td>NEC</td>
<td>NEC SX-8</td>
<td>NEC SX-8</td>
<td>2.00</td>
<td>576</td>
<td>576</td>
<td>160.95</td>
</tr>
<tr>
<td>6</td>
<td>IBM</td>
<td>Blue Gene</td>
<td>IBM PowerPC 440</td>
<td>0.70</td>
<td>2048</td>
<td>2048</td>
<td>96.19</td>
</tr>
<tr>
<td>7</td>
<td>IBM</td>
<td>Blue Gene</td>
<td>IBM PowerPC 440</td>
<td>0.70</td>
<td>1024</td>
<td>1024</td>
<td>70.94</td>
</tr>
<tr>
<td>8</td>
<td>Rackable Systems</td>
<td>Emerald</td>
<td>AMD Opteron</td>
<td>2.20</td>
<td>256</td>
<td>512</td>
<td>67.86</td>
</tr>
<tr>
<td>9</td>
<td>IBM</td>
<td>Blue Gene/L</td>
<td>IBM PowerPC 440</td>
<td>0.70</td>
<td>1024</td>
<td>1024</td>
<td>49.93</td>
</tr>
<tr>
<td>10</td>
<td>IBM</td>
<td>Blue Gene</td>
<td>IBM PowerPC 440</td>
<td>0.70</td>
<td>1024</td>
<td>1024</td>
<td>48.99</td>
</tr>
</tbody>
</table>

*• HPC Challenge Awards Class 1*
Top 10 Performance
STREAM Triad (per Process)

As of 1 November 2005

<table>
<thead>
<tr>
<th>Rank</th>
<th>Manufacturer</th>
<th>System</th>
<th>Processor Type</th>
<th>Processor Speed (GHz)</th>
<th>Procesor Count</th>
<th>MPI Processes</th>
<th>EP STREAM Triad</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>per Process</td>
</tr>
<tr>
<td>1</td>
<td>NEC</td>
<td>NEC SX-7</td>
<td>NEC SX-7</td>
<td>0.552</td>
<td>32</td>
<td>2</td>
<td>492.161</td>
</tr>
<tr>
<td>2</td>
<td>NEC</td>
<td>SX-8/6</td>
<td>NEC SX-8</td>
<td>2.000</td>
<td>6</td>
<td>6</td>
<td>61.7735</td>
</tr>
<tr>
<td>3</td>
<td>NEC</td>
<td>NEC SX-8</td>
<td>NEC SX-8</td>
<td>2.000</td>
<td>576</td>
<td>576</td>
<td>40.8954</td>
</tr>
<tr>
<td>4</td>
<td>NEC</td>
<td>NEC SX-6+</td>
<td>NEC SX-6</td>
<td>0.563</td>
<td>32</td>
<td>32</td>
<td>28.6168</td>
</tr>
<tr>
<td>5</td>
<td>NEC</td>
<td>SX-6</td>
<td>NEC SX-6</td>
<td>0.500</td>
<td>64</td>
<td>64</td>
<td>27.0884</td>
</tr>
<tr>
<td>6</td>
<td>NEC</td>
<td>SX-6</td>
<td>NEC SX-6</td>
<td>0.500</td>
<td>128</td>
<td>128</td>
<td>26.8584</td>
</tr>
<tr>
<td>7</td>
<td>NEC</td>
<td>SX-6</td>
<td>NEC SX-6</td>
<td>0.500</td>
<td>32</td>
<td>32</td>
<td>26.8319</td>
</tr>
<tr>
<td>8</td>
<td>NEC</td>
<td>SX-6</td>
<td>NEC SX-6</td>
<td>0.500</td>
<td>192</td>
<td>192</td>
<td>26.3087</td>
</tr>
<tr>
<td>9</td>
<td>NEC</td>
<td>NEC SX-7</td>
<td>NEC SX-7</td>
<td>0.552</td>
<td>32</td>
<td>32</td>
<td>26.1539</td>
</tr>
<tr>
<td>10</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.800</td>
<td>60</td>
<td>60</td>
<td>21.768</td>
</tr>
</tbody>
</table>
Top 10 Performance
STREAM Triad (per System)

As of 1 November 2005

<table>
<thead>
<tr>
<th>Rank</th>
<th>Manufacturer</th>
<th>System</th>
<th>Processor Type</th>
<th>Processor Speed (GHz)</th>
<th>Processor Count</th>
<th>MPI Processes per Process</th>
<th>EP STREAM Triad per Process</th>
<th>EP STREAM Triad per System</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Cray</td>
<td>Cray XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>5200</td>
<td>5.00</td>
<td>26020.80</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>NEC</td>
<td>NEC SX-8</td>
<td>NEC SX-8</td>
<td>2.00</td>
<td>576</td>
<td>40.90</td>
<td>23555.75</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>4096</td>
<td>5.04</td>
<td>20656.46</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.40</td>
<td>3744</td>
<td>4.85</td>
<td>18146.38</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.80</td>
<td>252</td>
<td>21.74</td>
<td>5478.73</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Cray</td>
<td>XT3</td>
<td>AMD Opteron</td>
<td>2.60</td>
<td>1100</td>
<td>4.80</td>
<td>5274.70</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>NEC</td>
<td>SX-6</td>
<td>NEC SX-6</td>
<td>0.50</td>
<td>192</td>
<td>26.31</td>
<td>5051.27</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.80</td>
<td>252</td>
<td>14.91</td>
<td>3758.40</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>NEC</td>
<td>SX-6</td>
<td>NEC SX-6</td>
<td>0.50</td>
<td>128</td>
<td>26.86</td>
<td>3437.88</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Cray</td>
<td>mfeg8</td>
<td>Cray X1E</td>
<td>1.13</td>
<td>248</td>
<td>13.23</td>
<td>3280.92</td>
<td></td>
</tr>
</tbody>
</table>

• HPC Challenge Awards Class 1
Top 10 Performance
DGEMM (per Process)

As of 1 November 2005

<table>
<thead>
<tr>
<th>Rank</th>
<th>Manufacturer</th>
<th>System</th>
<th>Processor Type</th>
<th>Processor Speed (GHz)</th>
<th>Processor Count</th>
<th>MPI Processes</th>
<th>EP DGEMM (GFlop/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NEC</td>
<td>NEC SX-7</td>
<td>NEC SX-7</td>
<td>0.55</td>
<td>32</td>
<td>2</td>
<td>140.64 281.27</td>
</tr>
<tr>
<td>2</td>
<td>IBM</td>
<td>eServer pSeries 655</td>
<td>IBM Power 4+</td>
<td>1.70</td>
<td>256</td>
<td>64</td>
<td>17.98  1150.68</td>
</tr>
<tr>
<td>3</td>
<td>IBM</td>
<td>eServer pSeries 655</td>
<td>IBM Power 4+</td>
<td>1.70</td>
<td>128</td>
<td>32</td>
<td>17.79  569.36</td>
</tr>
<tr>
<td>4</td>
<td>IBM</td>
<td>eServer pSeries 655</td>
<td>IBM Power 4+</td>
<td>1.70</td>
<td>64</td>
<td>16</td>
<td>17.50  280.00</td>
</tr>
<tr>
<td>5</td>
<td>NEC</td>
<td>SX-8/6</td>
<td>NEC SX-8</td>
<td>2.00</td>
<td>6</td>
<td>6</td>
<td>15.94  95.66</td>
</tr>
<tr>
<td>6</td>
<td>NEC</td>
<td>NEC SX-8</td>
<td>NEC SX-8</td>
<td>2.00</td>
<td>576</td>
<td>576</td>
<td>15.22  8768.56</td>
</tr>
<tr>
<td>7</td>
<td>Cray</td>
<td>Cray X1E</td>
<td>CrayX1E MSP</td>
<td>1.13</td>
<td>252</td>
<td>252</td>
<td>14.18  3574.54</td>
</tr>
<tr>
<td>8</td>
<td>Cray</td>
<td>mfeg8</td>
<td>Cray X1E</td>
<td>1.13</td>
<td>248</td>
<td>248</td>
<td>13.56  3363.87</td>
</tr>
<tr>
<td>9</td>
<td>Cray</td>
<td>X1E</td>
<td>Cray X1E</td>
<td>1.13</td>
<td>32</td>
<td>32</td>
<td>11.61  371.38</td>
</tr>
<tr>
<td>10</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.80</td>
<td>60</td>
<td>60</td>
<td>10.92  654.91</td>
</tr>
</tbody>
</table>
### Top 10 Performance RandomRing Latency

As of 1 November 2005

<table>
<thead>
<tr>
<th>Rank</th>
<th>Manufacturer</th>
<th>System</th>
<th>Processor Type</th>
<th>Processor Speed (GHz)</th>
<th>Interconnect</th>
<th>Processor Count</th>
<th>MPI Processes</th>
<th>RandomRing Latency (usec)</th>
<th>RandomRing Bandwidth (GB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PathScale, Inc.</td>
<td>Customer Benchmark Cluster</td>
<td>AMD Opteron</td>
<td>2.6</td>
<td>InfiniPath 1.0</td>
<td>32</td>
<td>32</td>
<td>1.31</td>
<td>0.27</td>
</tr>
<tr>
<td>2</td>
<td>Cray</td>
<td>XD1</td>
<td>AMD Opteron</td>
<td>2.2</td>
<td>RapidArray Interconnect System</td>
<td>64</td>
<td>64</td>
<td>1.63</td>
<td>0.23</td>
</tr>
<tr>
<td>3</td>
<td>Rackable Systems</td>
<td>Emerald</td>
<td>AMD Opteron</td>
<td>2.2</td>
<td>InfiniPath HTX InfiniBand Adapter SilverStorm 9120 InfiniBand Switch</td>
<td>64</td>
<td>128</td>
<td>2.02</td>
<td>0.12</td>
</tr>
<tr>
<td>4</td>
<td>Cray</td>
<td>XD1</td>
<td>AMD Opteron</td>
<td>2.4</td>
<td>Rapid Array Fat Tree</td>
<td>128</td>
<td>128</td>
<td>2.06</td>
<td>0.26</td>
</tr>
<tr>
<td>5</td>
<td>Rackable Systems</td>
<td>Emerald</td>
<td>AMD Opteron</td>
<td>2.2</td>
<td>InfiniPath HTX InfiniBand Adapter SilverStorm 9120 InfiniBand Switch</td>
<td>128</td>
<td>256</td>
<td>2.20</td>
<td>0.10</td>
</tr>
<tr>
<td>6</td>
<td>Rackable Systems</td>
<td>Emerald</td>
<td>AMD Opteron</td>
<td>2.2</td>
<td>InfiniPath HTX InfiniBand Adapter SilverStorm 9120 InfiniBand Switch</td>
<td>256</td>
<td>512</td>
<td>2.33</td>
<td>0.09</td>
</tr>
<tr>
<td>7</td>
<td>NEC</td>
<td>SX-8/6</td>
<td>NEC SX-8</td>
<td>2.0</td>
<td>Internode Crossbar Switch</td>
<td>6</td>
<td>6</td>
<td>3.02</td>
<td>13.55</td>
</tr>
<tr>
<td>8</td>
<td>SGI</td>
<td>Altix 3700 Bx2</td>
<td>Intel Itanium 2</td>
<td>1.6</td>
<td>N/A</td>
<td>32</td>
<td>32</td>
<td>3.26</td>
<td>1.52</td>
</tr>
<tr>
<td>9</td>
<td>SGI</td>
<td>Altix 3700 Bx2</td>
<td>Intel Itanium 2</td>
<td>1.6</td>
<td>N/A</td>
<td>64</td>
<td>64</td>
<td>3.68</td>
<td>0.87</td>
</tr>
<tr>
<td>10</td>
<td>SGI</td>
<td>Altix 3700 Bx2</td>
<td>Intel Itanium 2</td>
<td>1.6</td>
<td>N/A</td>
<td>128</td>
<td>128</td>
<td>3.91</td>
<td>0.90</td>
</tr>
</tbody>
</table>
### Top 10 Performance

**RandomRing Bandwidth**

As of 1 November 2005

<table>
<thead>
<tr>
<th>Rank</th>
<th>Manufacturer</th>
<th>System</th>
<th>Processor Type</th>
<th>Processor Speed (GHz)</th>
<th>Interconnect</th>
<th>Processor Count</th>
<th>MPI Processes</th>
<th>RandomRing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NEC</td>
<td>SX-8/6</td>
<td>NEC SX-8</td>
<td>2.000</td>
<td>Internode Crossbar Switch</td>
<td>6</td>
<td>6</td>
<td>3.02</td>
</tr>
<tr>
<td>2</td>
<td>NEC</td>
<td>NEC SX-7</td>
<td>NEC SX-7</td>
<td>0.552</td>
<td>non</td>
<td>32</td>
<td>2</td>
<td>4.85</td>
</tr>
<tr>
<td>3</td>
<td>NEC</td>
<td>NEC SX-7</td>
<td>NEC SX-7</td>
<td>0.552</td>
<td>non</td>
<td>32</td>
<td>32</td>
<td>14.21</td>
</tr>
<tr>
<td>4</td>
<td>SGI</td>
<td>Altix 3700 Bx2</td>
<td>Intel Itanium 2</td>
<td>1.600</td>
<td>N/A</td>
<td>32</td>
<td>32</td>
<td>3.26</td>
</tr>
<tr>
<td>5</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.800</td>
<td>Cray modified 2-D Torus</td>
<td>32</td>
<td>32</td>
<td>14.94</td>
</tr>
<tr>
<td>6</td>
<td>Cray</td>
<td>X1E</td>
<td>Cray X1E</td>
<td>1.130</td>
<td>Cray Interconnect</td>
<td>32</td>
<td>32</td>
<td>12.21</td>
</tr>
<tr>
<td>7</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.800</td>
<td>Cray modified 2-D Torus</td>
<td>60</td>
<td>60</td>
<td>14.66</td>
</tr>
<tr>
<td>8</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.800</td>
<td>Cray modified 2D torus</td>
<td>60</td>
<td>60</td>
<td>20.83</td>
</tr>
<tr>
<td>9</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.800</td>
<td>Cray modified 2D torus</td>
<td>60</td>
<td>60</td>
<td>21.16</td>
</tr>
<tr>
<td>10</td>
<td>Cray</td>
<td>X1</td>
<td>Cray X1 MSP</td>
<td>0.800</td>
<td>Cray modified 2D torus</td>
<td>64</td>
<td>64</td>
<td>20.34</td>
</tr>
</tbody>
</table>
• How well does HPL data correlate with theoretical peak performance?
HPL versus DGEMM

- Can I Run Just Run DGEMM Instead of HPL?
- DGEMM alone overestimates HPL performance
- Note the 1,000x difference in scales! (Tera/Giga)
HPL versus STREAM Triad

- How well does HPL correlate with G-RandomAccess performance?
HPL versus RandomAccess

- How well does HPL correlate with G-RandomAccess performance?
- Note the 1,000x difference in scales! (Tera/Giga)
HPL versus FFT

- How well does HPL correlate with FFT performance?
- Note the 1,000x difference in scales! (Tera/Giga)
Global STREAM versus PTRANS

- How well does STREAM data correlate with PTRANS performance?
RandomRing Bandwidth versus PTRANS

- How well does RandomRing Bandwidth data correlate with PTRANS performance?
- Possible bad data?
RandomAccess Correlations?

- HPL versus G-RandomAccess
- Number of Processors versus G-RandomAccess
- RandomRing Bandwidth versus G-RandomAccess
- RandomRing Latency versus G-RandomAccess
- Single Processor RandomAccess versus G-RandomAccess
  - per System (Single Processor)
  - per Processor (Single Processor)
- STREAM Triad versus G-RandomAccess
Number of Processors versus G-RandomAccess

- Does G-RandomAccess scale with the number of processors?
RandomRing Bandwidth versus G-RandomAccess

- Does G-RandomAccess scale with the RandomRing Bandwidth?
- Possible bad data?
RandomRing Bandwidth versus G-RandomAccess

- Does G-RandomAccess scale with RandomRing Bandwidth?
- Ignoring possible bad data…
RandomRing Latency versus G-RandomAccess

- Does G-RandomAccess scale with RandomRing Latency?
• Does G-RandomAccess scale with single processor RandomAccess performance (per system)?
Does G-RandomAccess scale with single processor RandomAccess performance?
STREAM Triad (per System) versus G-RandomAccess

- Does G-RandomAccess scale with STREAM Triad?
RandomAccess Correlations?

- HPL versus G-RandomAccess
- Number of Processors versus G-RandomAccess
- RandomRing Bandwidth versus G-RandomAccess
- RandomRing Latency versus G-RandomAccess
- Single Processor RandomAccess versus G-RandomAccess
  - per System (Single Processor)
  - per Processor (Single Processor)
- STREAM Triad versus G-RandomAccess
RandomAccess Correlations?

- HPL versus G-RandomAccess
- Number of Processors versus G-RandomAccess
- RandomRing Bandwidth versus G-RandomAccess
- RandomRing Latency versus G-RandomAccess
- Single Processor RandomAccess versus G-RandomAccess
  - per System (Single Processor)
  - per Processor (Single Processor)
- STREAM Triad versus G-RandomAccess

- Biggest factor in G-RandomAccess improved performance is optimized codes!
  - Rules on storing updates forces non-optimal short messages in MPI
  - UPC

HPC Challenge Awards will be presented at the HPC Challenge Award BOF at SC|05 Tuesday 15 November 2005 at noon!
G-RandomAccess UPC Code‡

‡ Yiyi Yao (GWU)

```c
void RandomAccessUpdate(u64Int TableSize)
{
    s64Int i;
    u64Int ran[MAXJOBS];
    int j;

    /* Translated for loop from upc_forall construct */
    #pragma _CRI ivdep
    #pragma _CRI concurrent
    for( i = MYTHREAD;  i<TableSize;  i += THREADS)
        Table[i] = i;
    upc_barrier;

    /* Translated for loop from upc_forall construct */
    #pragma _CRI ivdep
    #pragma _CRI concurrent
    for( j = MYTHREAD;  j<MAXJOBS;  j += THREADS)
    {
        ran[j] = starts ((NUPDATE/MAXJOBS) * j);
        for (i=0; i<NUPDATE/MAXJOBS; i++ )
            ran[j] = (ran[j] << 1) ^ ((s64Int) ran[j] < 0 ? POLY : 0);
        Table[(ran[j] & (TableSize-1))] ^= ran[j];
    }
}
```

Initialize Table

Update Table
HPC Challenge Benchmark Suite

http://icl.cs.utk.edu/hpcc/

The HPC Challenge benchmark consists of basically 7 tests:

1. **HPL** - the Linpack TPP benchmark which measures the floating point rate of execution for solving a linear system of equations.

2. **DGEMM** - measures the floating point rate of execution of double precision real matrix-matrix multiplication.

3. **STREAM** - a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.

4. **PTRANS** (parallel matrix transpose) - exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.

5. **RandomAccess** - measures the rate of integer random updates of memory (GUPS).

6. **FFTE** - measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).

7. Communication bandwidth and latency - a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns; based on $b_{eff}$ (effective bandwidth benchmark).
Kiviat Charts

- Comparisons of multiple systems for
  1. Per Processor HPL
  2. Per Processor PTRANS
  3. Per Processor Global RandomAccess
  4. Per Processor Global FFTE
  5. Single Node STREAM Triad
  6. Single Node DGEMM
  7. System RandomRing Latency
  8. System RandomRing Bandwidth

- Data in each dimension is normalized to the maximum value
- Represented on a linear scale [0,1]
- Best when showing the differences due to a single independent “variable”
Kiviat Chart Disclaimer

• Please remember that each Kiviat chart should include the following disclaimer
• It has not been included due to vugraph orientation and to minimize clutter

Differences in the benchmark results between computers, even of the same model, can be a result of the number of processors used, the number of threads used, the processor interconnect, the amount of memory allocated for the run, the version of the BLAS and MPI, and other factors. A complete listing of the environment for each benchmark run can be found at: http://icl.cs.utk.edu/hpcc/export/hpcc.xls
1. RandomRing Bandwidth
   InfiniBand has significantly greater bandwidth than other technologies

2. RandomRing Latency
   InfiniBand and SCI have significantly lower latencies than other technologies

3. STREAM, DGEMM, and HPL
   Interconnect technology doesn’t matter
   STREAM and DGEMM have no communications
   HPL scales well with respect to communications

4. RandomAccess
   Interconnect technology does matter! Latency sensitive

5. PTRANS and FFT
   Interconnect technology does matter Bandwidth sensitive
1. **RandomRing Bandwidth**
The Cray XD1 has greater bandwidth than the other technologies

2. **RandomRing Latency**
The Cray XD1 has significantly lower latency than other technologies

3. **STREAM, DGEMM, and HPL**
Interconnect technology doesn’t matter
STREAM and DGEMM have no communications
HPL scales well with respect to communications

4. **RandomAccess**
Interconnect technology does matter Extremely latency sensitive!

5. **PTRANS and FFT**
Interconnect technology does matter! Bandwidth sensitive
HPC Challenge Analysis
Kiviat Diagram — Cray XT3 Comparison

Per System — Absolute Scaling for Operations Benchmarks

Per Processor Performance “Weak Scaling”

1. RandomRing Bandwidth
   The smallest model has the highest bandwidth? MPI?

2. RandomRing Latency
   The newest model has the lowest latency

3. STREAM and DGEMM
   Slight differences in models?

4. HPL
   Some degradation when scaling to larger machines

5. RandomAccess
   Latency dependent and scales inversely proportional to number of processors

6. PTRANS
   Bandwidth sensitive

7. FFTE
   Bandwidth and processor speed sensitive
HPC Challenge Analysis
Kiviat Diagram — Rackable Cluster Comparison

Per System — Absolute Scaling for Operations Benchmarks

Per Processor Performance

“Weak Scaling”

1. RandomRing Bandwidth
Smaller system has greater bandwidth per processor

2. RandomRing Latency
Smaller system has lower latency per processor

3. DGEMM, and HPL
Similar performance

4. STREAM
Minor variations in performance??

5. RandomAccess
Extremely latency or bandwidth sensitive!

6. PTRANS
Variations in performance??

7. FFTE
Some latency or bandwidth sensitivity!
1. **RandomRing Bandwidth**
   System with two cores has significantly lower bandwidth
   \(\Rightarrow\) cores vs interconnect?

2. **RandomRing Latency**
   System with two cores has slightly lower latency than one technology
   \(\Rightarrow\) cores vs connect?

3. **STREAM and DGEMM**
   Significantly reduced performance for 2 cores

4. **HPL, RandomAccess, and FFT**
   Top per processor performance?
   - HPL \(~2x\) single cores
   - RA 2.5-10x single cores
   - FFTE slightly better

5. **PTRANS**
   Bandwidth sensitivity but 2 core better than expected
1. **RandomRing Bandwidth**
   Quadrics QsNet provides greater bandwidth

2. **RandomRing Latency**
   InfiniBand provides lower latency

3. **STREAM, DGEMM, and HPL**
   Interconnect technology doesn’t matter but Intel Xeons are faster
   STREAM and DGEMM have no communications
   HPL scales well with respect to communications

4. **RandomAccess**
   Interconnect technology does matter — but uncertain if bandwidth or latency dependent

5. **PTRANS and FFTTE**
   Interconnect technology does matter Bandwidth sensitive
Comparing Dissimilar Systems Can be Difficult!

1. **RandomRing Bandwidth**
   - X1 and X1E have higher bandwidth

2. **RandomRing Latency**
   - XT3 has lower latency when using MPI

3. **STREAM, DGEMM, and HPL**
   - Interconnect technology doesn’t matter and X1 and X1E are faster

4. **RandomAccess**
   - Interconnect technology does matter — but poor X1 and X1E performance due to MPI latencies

5. **PTRANS**
   - Interconnect technology does matter Bandwidth sensitive

6. **FFTE**
   - XT3 is significantly faster
HPC Challenge Analysis
Kiviat Diagram — Custom Interconnect Comparison

1. RandomRing Bandwidth
   SGI Altix has highest bandwidth

2. RandomRing Latency
   Cray XD1 has lowest latency

3. STREAM
   NEC SX-6 vector processor dominates

4. DGEMM and HPL
   NEC SX-6 vector processor slightly better
   Otherwise similar performance

5. RandomAccess
   Latency dependent

6. PTRANS
   NEC SX-6 vector processor dominates

7. FFT
   NEC SX-6 vector processor slightly better
   Otherwise similar performance
HPC Challenge Analysis

Kiviat Diagram — NEC SX-6/7 Comparison

1. **RandomRing Bandwidth**
   
   SX-7 has significantly higher bandwidth

2. **RandomRing Latency**
   
   SX-7 has lowest latency

3. **All remaining benchmarks**
   
   Clock frequency dependent
HPC Challenge v1.x Benchmark Suite Outline

- Introduction
- Motivations
  - HPCS
  - Performance Characterization
- Component Kernels
- HPC Challenge Awards
- Unified Benchmark Framework
- Rules
  - Running HPC Challenge
  - Optimizations
  - Etiquette
- Performance Data
  - Available Benchmark Data
  - Kiviat Charts
- Hands-on Demonstrations/Exercises
  - Installing the HPC Challenge v1.x Benchmark Unified Benchmark Framework
  - Running the HPC Challenge v1.x Benchmark suite
- Summary/Conclusions
HPC Challenge Tutorial: Hands-on Demonstrations/Exercises

Piotr Luszczek
University of Tennessee
Download

- Always use the latest source code:

---

HPC Challenge Benchmark

The HPC Challenge benchmark consists of basically 7 tests:

1. HPL - the Linpack TPP benchmark which measures the floating point rate of execution for solving a linear system of equations.

2. DGEMM - measures the floating point rate of execution of double precision real matrix-matrix multiplication.

3. STREAM - a simple synthetic benchmark program that measures sustainable memory bandwidth (in GB/s) and the corresponding computation rate for simple vector kernel.

4. PTRANS (parallel matrix transpose) - exercises the communications where pairs of processors communicate with each other simultaneously. It is a useful test of the total communications capacity of the network.

5. RandomAccess - measures the rate of integer random updates of memory (GUPS).

6. FFT - measures the floating point rate of execution of double precision complex one-dimensional Discrete Fourier Transform (DFT).

7. Communication bandwidth and latency - a set of tests to measure latency and bandwidth of a number of simultaneous communication patterns, based on b_eff (effective bandwidth benchmark).
HPCC Software Versioning


  Latest version is always at the top
HPCC Makefile Structure (1of 2)

- Sample Makefiles live in `hpl/setup`
- **BLAS**
  - `LAdir` – BLAS top directory for other `LA`-variables
  - `LAinc` – where BLAS headers live (if needed)
  - `LAlib` – where BLAS libraries live (`libmpi.a` and friends)
  - `F2CDEFS` – resolves Fortran-C calling issues (BLAS is usually callable from Fortran)
    - `-DAdd_`, `-DNoChange`, `-DUpCase`, `-Dadd__`
    - `-DStringSunStyle`, `-DStringStructPtr`, `-DStringStructVal`, `-DStringCrayStyle`
- **MPI**
  - `MPdir` – MPI top directory for other `MP`-variables
  - `MPinc` – where MPI headers live (`mpi.h` and friends)
  - `MPlib` – where MPI libraries live (`libmpi.a` and friends)
• Compiler
  - CC – C compiler
  - CCNOOPT – C flags without optimization (for optimization-sensitive code)
  - CCFLAGS – C flags with optimization

• Linker
  - LINKER – program that can link BLAS and MPI together
  - LINKFLAGS – flags required to link BLAS and MPI together

• Programs/commands
  - SHELL, CD, CP, LN_S, MKDIR, RM, TOUCH
  - ARCHIVER, ARFLAGS, RANLIB
MPI Implementations for HPCC

- **Vendor**
  - Cray (MPT)
  - IBM (POE)
  - SGI (MPT)
  - Dolphin, Infiniband (Mellanox, Voltaire, ...), Myricom (GM, MX), Quadrics, PathScale, Scali, ...

- **Open Source**
  - Lam MPI ([http://www.lam-mpi.org/](http://www.lam-mpi.org/))
  - OpenMPI ([http://www.open-mpi.org/](http://www.open-mpi.org/))

- **MPI implementation components**
  - Compiler (adds MPI header directories)
  - Linker (need to link in Fortran I/O)
  - Exe (poe, mprun, mpirun, aprun, mpiexec, ...)
Fast BLAS for HPCC

• Vendor
  – AMD (AMD Core Math Library)
  – Cray (SciLib)
  – HP (MLIB)
  – IBM (ESSL)
  – Intel (Math Kernel Library)
  – SGI (SGI/Cray Scientific Library)
  – ...

• Free implementations
  – ATLAS
    http://www.netlib.org/atlas/
  – Goto BLAS
    http://www.cs.utexas.edu/users/flame/goto
    http://www.tacc.utexas.edu/resources/software

• Implementations that use Threads
  – Some vendor BLAS
  – Atlas
  – Goto BLAS

• You should never use reference BLAS from Netlib
  – There are better alternatives for every system in existence
Tuning Process - Internal

- Changes to source code are not allowed for submission
- But just for tuning it's best to change a few things
  - Switch off some tests temporarily
- Choosing right parallelism levels
  - Processes (MPI)
  - Threads (OpenMP in code, vendor in BLAS)
  - Processors
  - Cores
- Compile time parameters
  - More details below
- Runtime input file
  - More details below
Tuning Process - External

• MPI settings examples
  – Messaging modes
    ▪ Eager polling is probably not a good idea
  – Buffer sizes
  – Consult MPI implementation documentation

• OS settings
  – Page size
    ▪ Large page size should be better on many systems
  – Pinning down the pages
    ▪ Optimize affinity on DSM architectures
  – Priorities
  – Consult OS documentation
Parallelism Examples

• Pseudo-threading helps but be careful
  – Hyper-threading
  – Simultaneous Multi-Threading
  – ...

• Cores
  – Intel (x86-64, Itanium), AMD (x86)
  – Cray: SSP, MSP
  – IBM Power4, Power5, ...
  – Sun SPARC

• SMP
  – BlueGene/L (single/double CPU usage per card)
  – SGI (NUMA, ccNUMA, DSM)
  – Cray, NEC

• Others
  – Cray MTA (no MPI !)
HPCC Input and Output

- Parameter file hpccinf.txt
  - HPL parameters
    - Lines 5-31
  - PTRANS parameters
    - Lines 32-36
  - Indirectly: sizes of arrays for all HPCC components
    - Hard coded

- Memory file hpccmemf.txt
  - Memory available per MPI process
    - Process=64
  - Memory available per thread
    - Thread=64
  - Total available memory
    - Total=64
  - Many HPL and PTRANS parameters might not be optimal

- Output file hpccoutf.txt
  - Must be uploaded to the website
  - Easy to parse
  - More details later...
Tuning HPL - Introduction

• Performance of HPL comes from
  – BLAS
  – Input file hpccinf.txt

• Essential parameters in the input file
  – N – matrix size
  – NB – blocking factor — influences BLAS performance and load balance
  – PMAP – process mapping — depends on network topology
  – PxQ – process grid

• Definitions

\[ A \ x = b \]

\[ N \ x = b \]

\[ \begin{pmatrix} P_X & P_Y & P_Z & P_X \end{pmatrix} \]

\[ NB \]
Tuning HPL – More Definitions

- Process grid parameters: P, Q, and PMAP

\[ Q = 4 \]

\[ P = 3 \]

PMAP = C

PMAP = R
Tuning HPL – Selecting Process Grid

Performance of HPL with Atlas 3.7 on Intel Xeon 64 EMT 3.2 GHz

Matrix size

Performance [Gflop/s]

- 12x10
- 10x12
- 8x15
- 6x20
- 5x24
- 4x30
- 3x40
- 2x60
- 1x120
Tuning HPL – Number of Processors

Time to solve 70k linear system on AMD Athlon 1.4 GHz cluster with Myrinet 2000

Worst virtual process grid
Best virtual process grid

Prime numbers

Number of processors

Wall clock time [s]
Tuning HPL – Matrix Size

Performance of HPL with Atlas 3.7 on Intel Xeon 64EMT 3.2 GHz

- Too small
- Best performance
- Not optimal parameters
- Too big

- Too big: 12 x 10
- Too big: 6 x 10
HPL - Website


- Much more details from HPL's author:

- Antoine Petitet
Tuning FFT

• Compile-time parameters
  – FFTE_NBLK – blocking factor
  – FFTE_NP – padding (to alleviate negative cache-line effects)
  – FFTE_L2SIZE – size of level 2 cache

• Use FFTW instead of FFTE
  – Define USING_FFTW symbol during compilation
  – Add FFTW location and library to linker flags
Tuning STREAM

- Intended to measure main memory bandwidth
- Requires many optimizations to run at full hardware speed
  - Software pipelining
  - Prefetching
  - Loop unrolling
  - Data alignment
  - Removal of array aliasing
- Original STREAM has advantages
  - Constant array sizes (known at compile time)
  - Static storage of arrays (at full compiler's control)
Tuning PTRANS

- Parameter file `hpccinf.txt`
  - Line 33 — number of matrix sizes
  - Line 34 — matrix sizes
    - Must not be too small – enforced in the code
  - Line 35 — number of blocking factors
  - Line 36 — blocking factors
    - No need to worry about BLAS
    - Very influential for performance
Tuning b_eff

- b_eff (Effective bandwidth and latency) test can also be tuned
- Tuning must use only standard MPI calls
- Examples
  - Persistent communication
  - One-sided communication
HPCC Output File

• The output file has two parts
  – Verbose output (free format)
  – Summary section
    ▪ Pairs of the form: `name=value`

• The summary section names
  – MPI* — global results
    ▪ Example: MPIRandomAccess_GUPs
  – Star* — embarrassingly parallel results
    ▪ Example: StarRandomAccess_GUPs
  – Single* — single process results
    ▪ Example: SingleRandomAccess_GUPs
Submitting Result Data

- Output file `hpccoutf.txt` should be submitted along with system info.
HPC Challenge - Benchmark Results Submission Form

Please fill out all fields below and be sure to include your benchmark results. Your benchmark results should not have been edited or modified in any way. You will be sent an email to the address entered below containing an URL that you will be required to visit to confirm your submission. (Your submission will not be entered into the database unless this step is taken.) Thank you.

First name: ___________________________
Last name: ___________________________
Email address: ________________________
Machine Location: _________________
City: ___________________________
Institution/Affiliation: ______________
URL: ____________________________

Help?
Optimized Run Ideas

• For optimized run the same MPI harness has to be run on the same system
• Certain routines can be replaced – the timed regions
• The verification has to pass – limits data layout and accuracy of optimization
• Variations of the reference implementation are allowed (within reason)
  – No Strassen algorithm for HPL due to different operation count
• Various non-portable C directives can significantly boost performance
  – Example: #pragma ivdep
• Various messaging substrates can be used
  – Removes MPI overhead
• Various languages can be used
  – Allows for direct access to non-portable hardware features
  – UPC was used to increase RandomAccess performance by orders of magnitude
• Optimizations need to be explained upon results submission
HPC Challenge v1.x Benchmark Suite
Outline

- Introduction
- Motivations
  - HPCS
  - Performance Characterization
- Component Kernels
- HPC Challenge Awards
- Unified Benchmark Framework
- Rules
  - Running HPC Challenge
  - Optimizations
  - Etiquette
- Performance Data
  - Available Benchmark Data
  - Kiviat Charts
- Hands-on Demonstrations/Exercises
  - Installing the HPC Challenge v1.x Benchmark Unified Benchmark Framework
  - Running the HPC Challenge v1.x Benchmark suite
- Summary/Conclusions
Summary/Conclusions

- HPC Challenge Benchmark Suite
  - To examine the performance of HPC architectures using kernels with more challenging memory access patterns than HPL
  - To augment the Top500 list
  - To provide benchmarks that bound the performance of many real applications
  - Available for download http://icl.cs.utk.edu/hpcc/

As of 1 November 2005

HPC Challenge Awards will be presented at the SC|05 HPC Challenge Award BOF Tuesday 15 November 2005 at noon!