



# 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**

- **ICL Team**



**Jack Dongarra**  
dongarra@cs.utk.edu University  
Distinguished Professor



**Piotr Luszczek**  
luszczek@cs.utk.edu  
Research Scientist

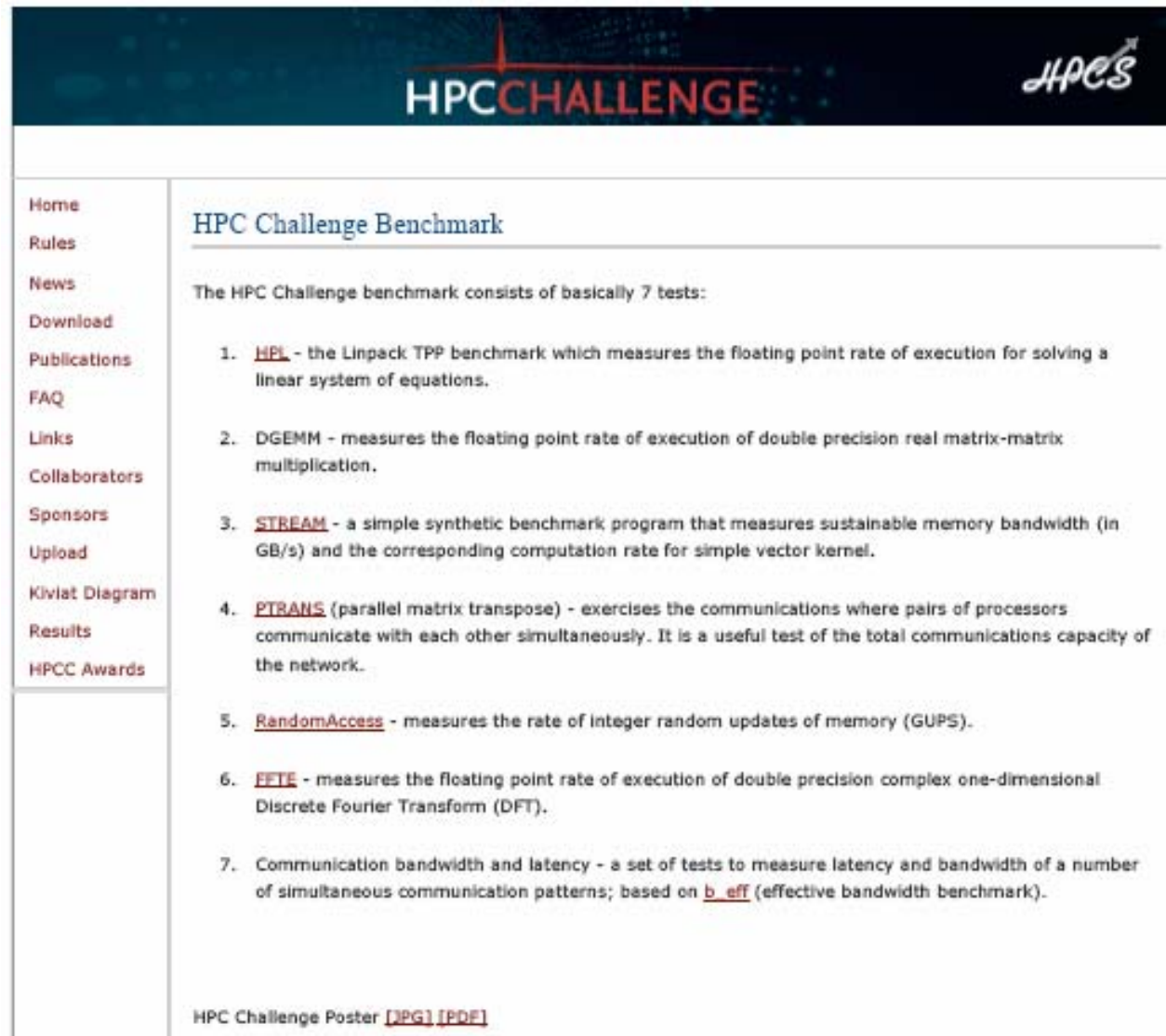
- **Collaborators**

- 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 Petit (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



The screenshot shows the HPC Challenge Benchmark Suite website. The header features the 'HPCCHALLENGE' logo and the 'HPCC' logo. A left sidebar contains navigation links: Home, Rules, News, Download, Publications, FAQ, Links, Collaborators, Sponsors, Upload, Kiviat Diagram, Results, and HPCC Awards. The main content area is titled 'HPC Challenge Benchmark' and lists seven 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).

At the bottom of the main content area, there is a link: HPC Challenge Poster [\[JPG\]](#) [\[PDF\]](#)

# 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)



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)
6. RandomAccess (EP & G)
7. Communications Bandwidth and Latency

### Local and Embarrassingly Parallel

1. EP-DGEMM (matrix x matrix multiply)
2. STREAM
  - Copy
  - Scale
  - Add
  - Triad
3. EP-RandomAccess
4. EP-1DFFT

### Global

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



# HPC Challenge v1.x Benchmark Suite

## Introduction (4 of 6)



- **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

# HPC Challenge v1.x Benchmark Suite

## Introduction (5 of 6)



- **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  $m$ )**
  - $n^2 \leq m \leq \text{Available Memory}$
  - 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)

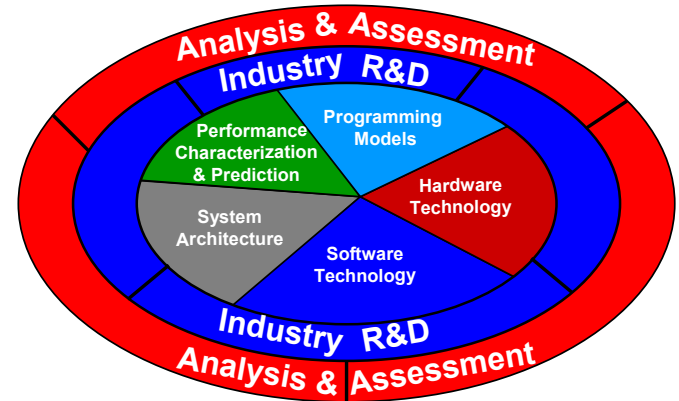


## Goal:

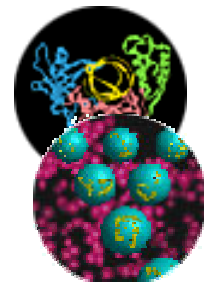
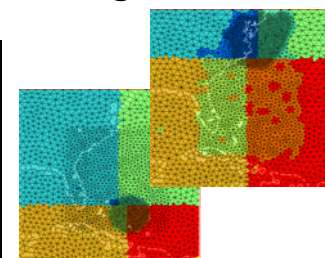
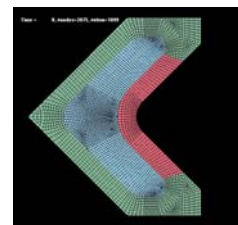
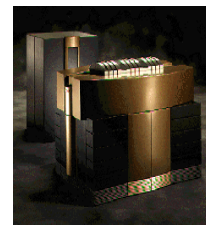
- Provide a new generation of economically viable high productivity computing systems for the national security and industrial user community (2010)

## 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



HPCS Program Focus Areas



## 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)

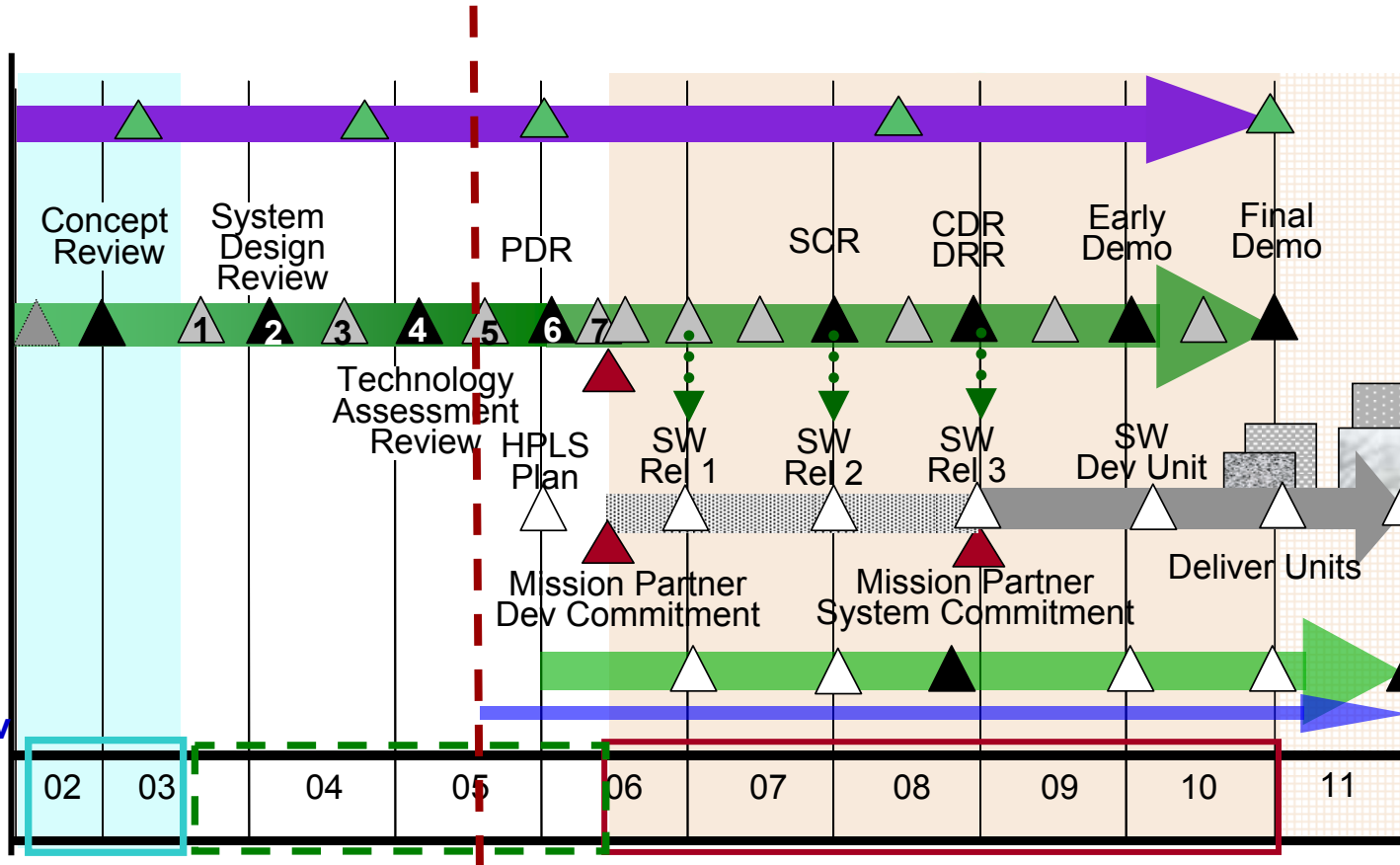
Industry Milestones

MP Peta-Scale Procurements

Mission Partner Peta-Scale Application Dev

MP Language Dev

Year (CY)



(Funded Five) Phase I Industry Concept Study

(Funded Three) Phase II R&D

Phase III Prototype Development

Mission Partners

- ▲ Program Reviews
- ▲ Critical Milestones
- ▲ Program Procurements





# 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

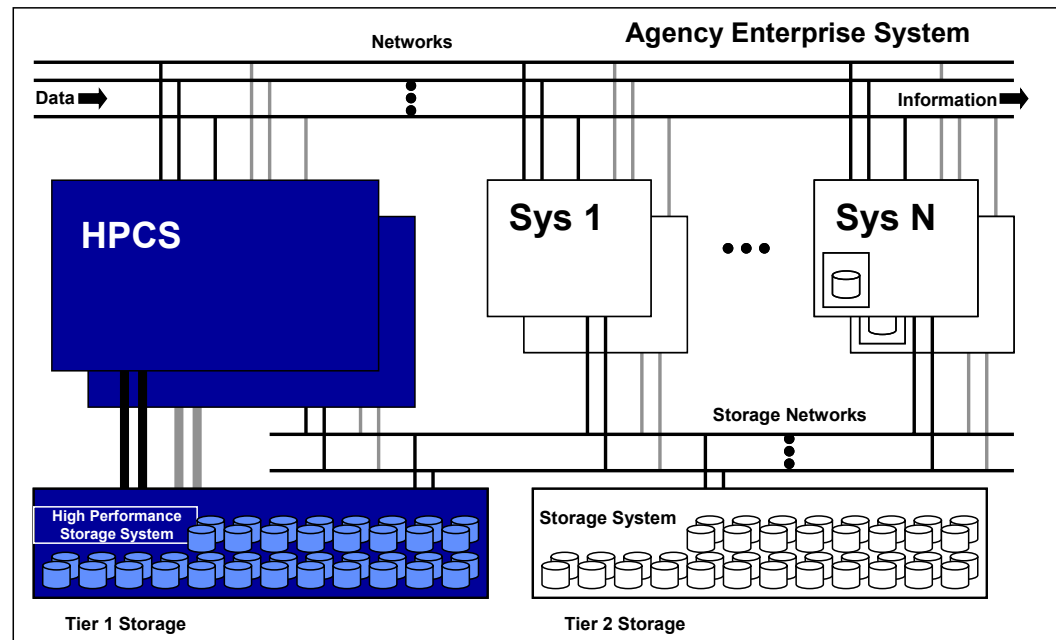
10 to 10K times Delta  
from Business as Usual

**Documented and Validated Through Simulations,  
Experiments, Prototypes, and Analysis**

## 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

## HPCS as part of a Mission Partner's Enterprise Architecture







## Productivity Team Lead



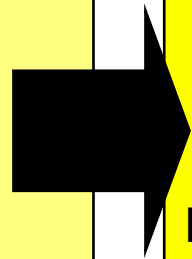
Jeremy Kepner

**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)**

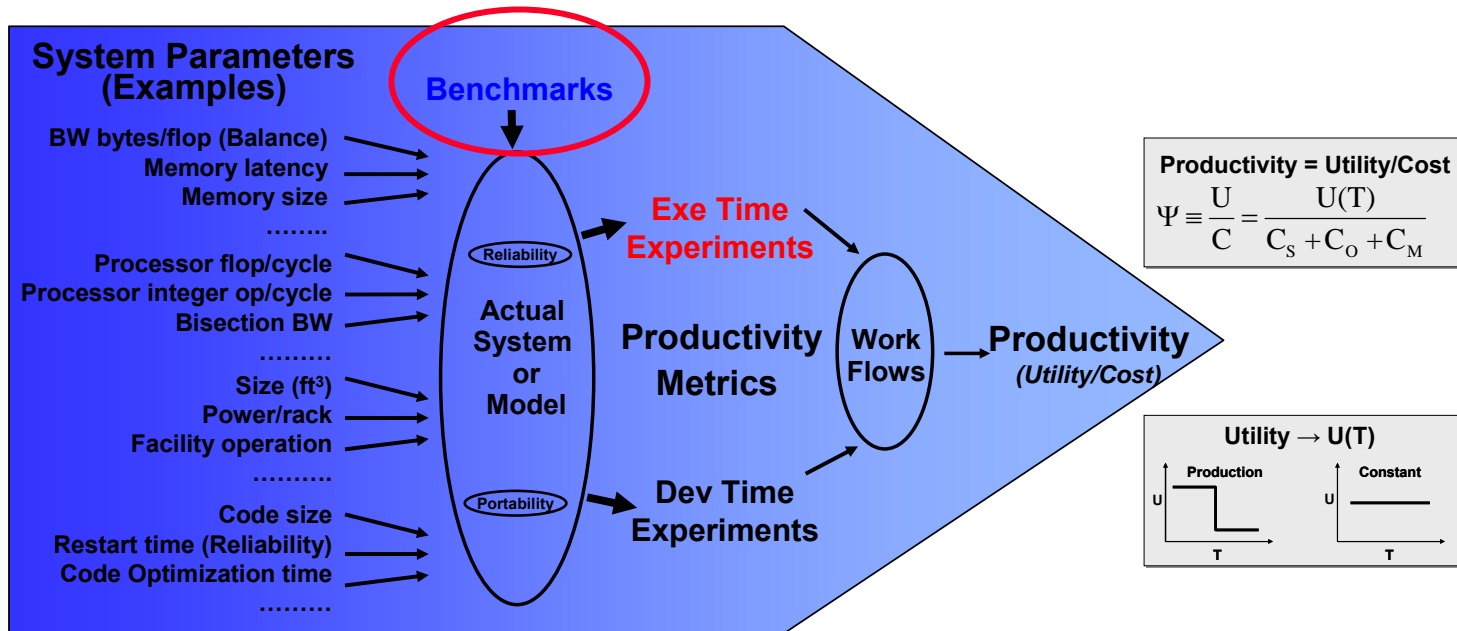
**Development Experiments**  
**Workflows, Models, Metrics**  
**High Productivity Language Systems**  
**Execution Time Models**  
**Test & Spec Specifications**



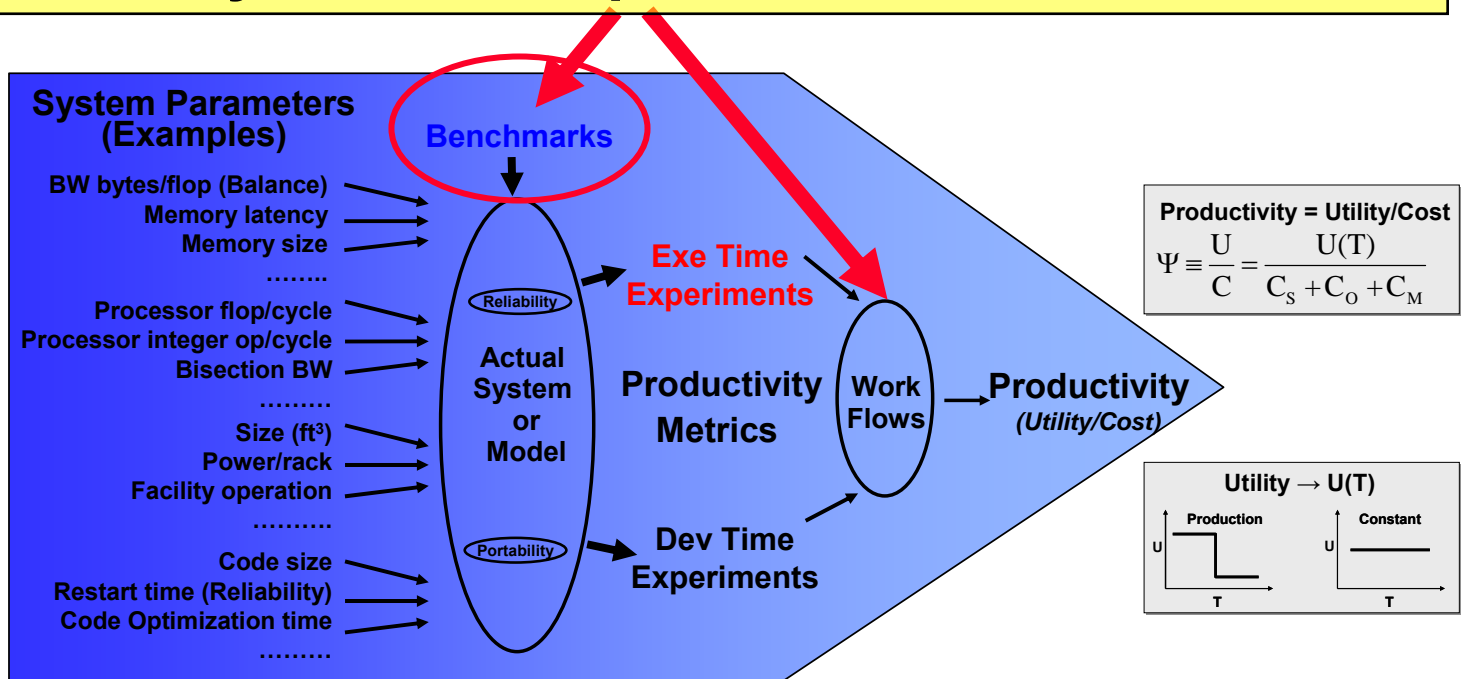
# HPCS Benchmark Working Group Goals

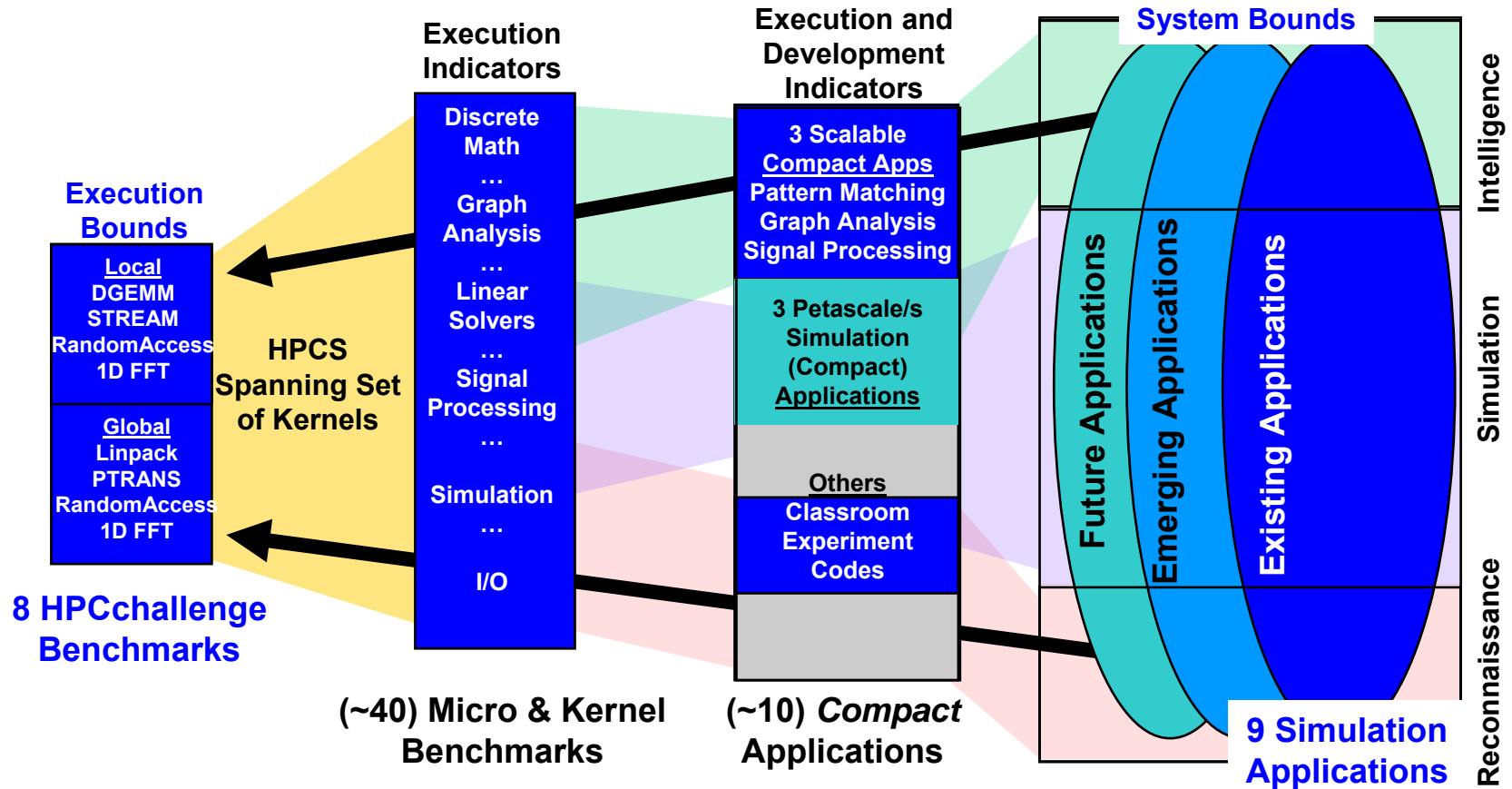


- Provide the HPCS Vendors and HPCS Productivity Team the Benchmarks and Applications for
  - Scoping requirements for designing systems
  - Productivity Testing
    - Execution Time Testing
    - Development Time Testing



**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 *traditional* parameter sets**



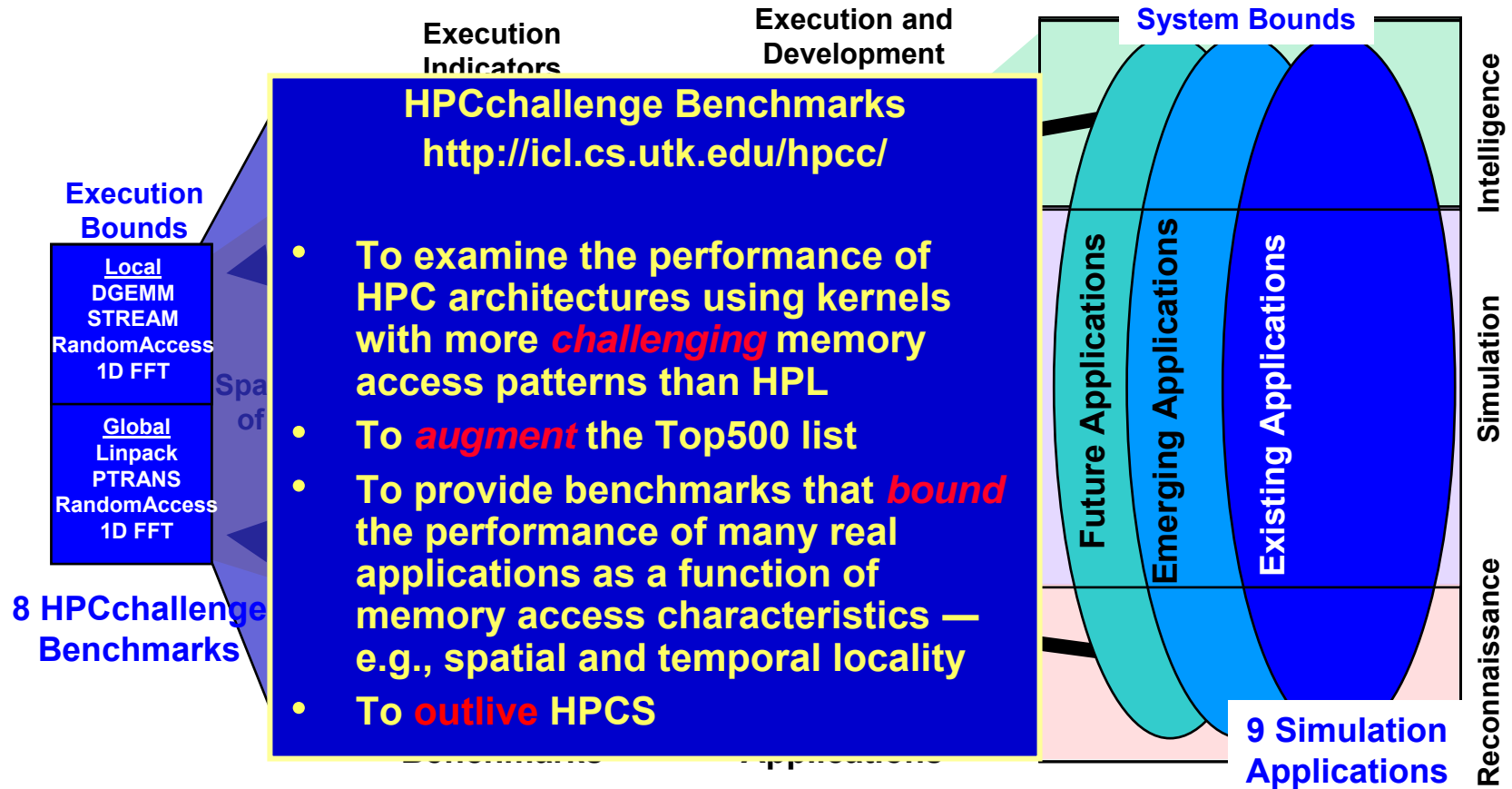


**Spectrum of benchmarks provide different views of system**

- 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)



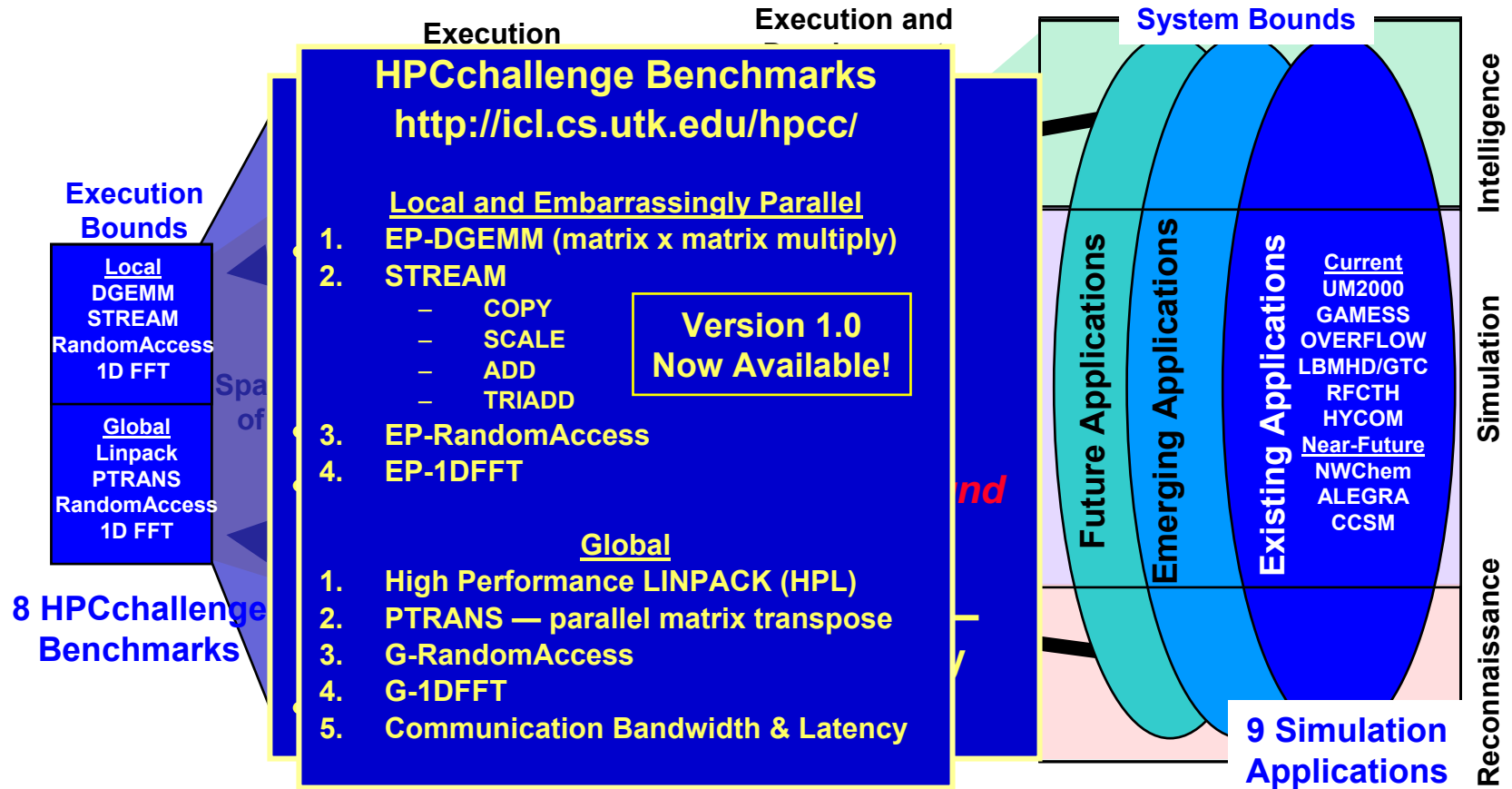
**HPCchallenge Benchmarks**  
<http://icl.cs.utk.edu/hpcc/>

- 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

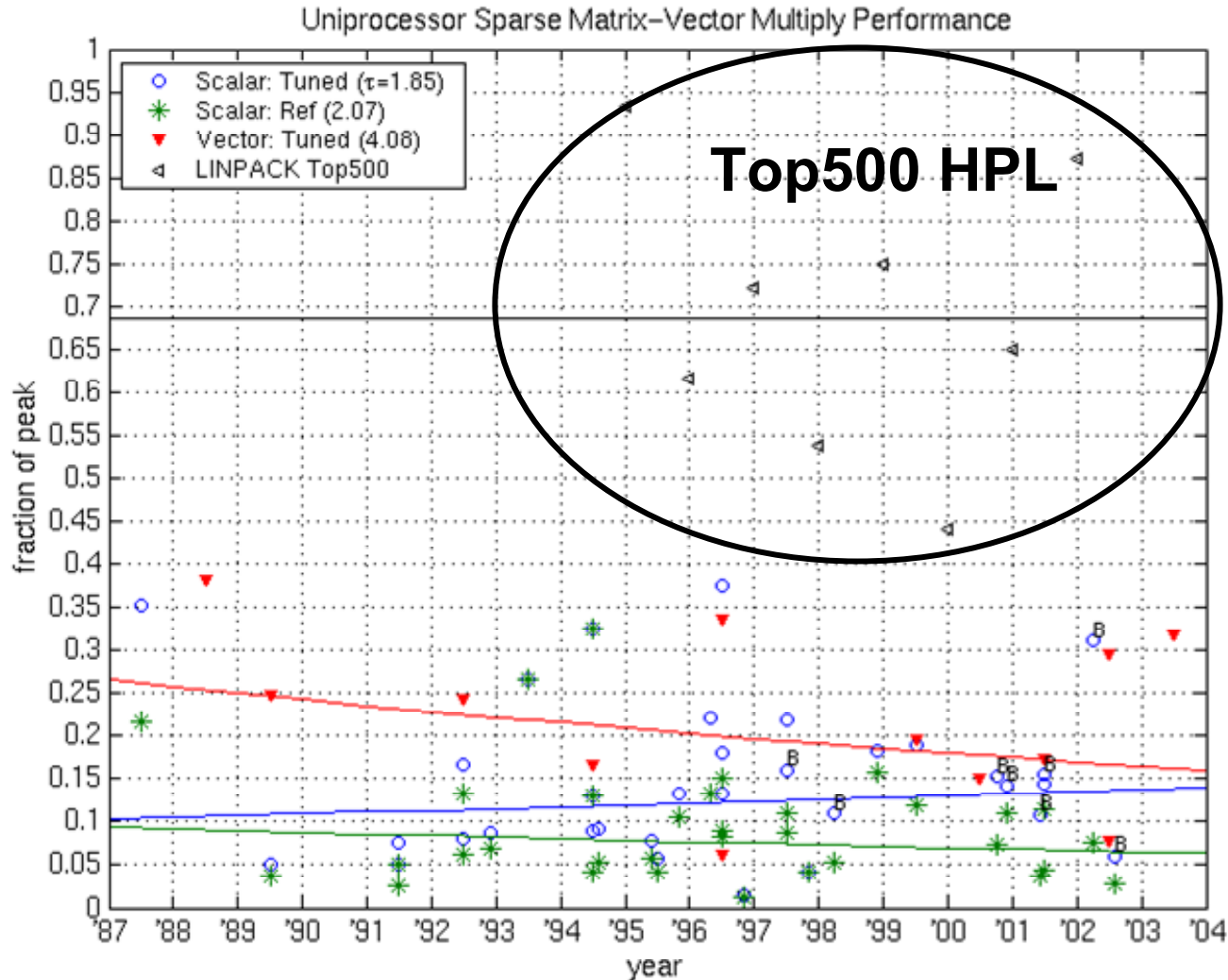


- **Scalable framework — Unified Benchmark Framework**
  - By design, the HPC Challenge Benchmarks are scalable with the size of data sets being a function of the largest HPL matrix for the tested system

# 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?

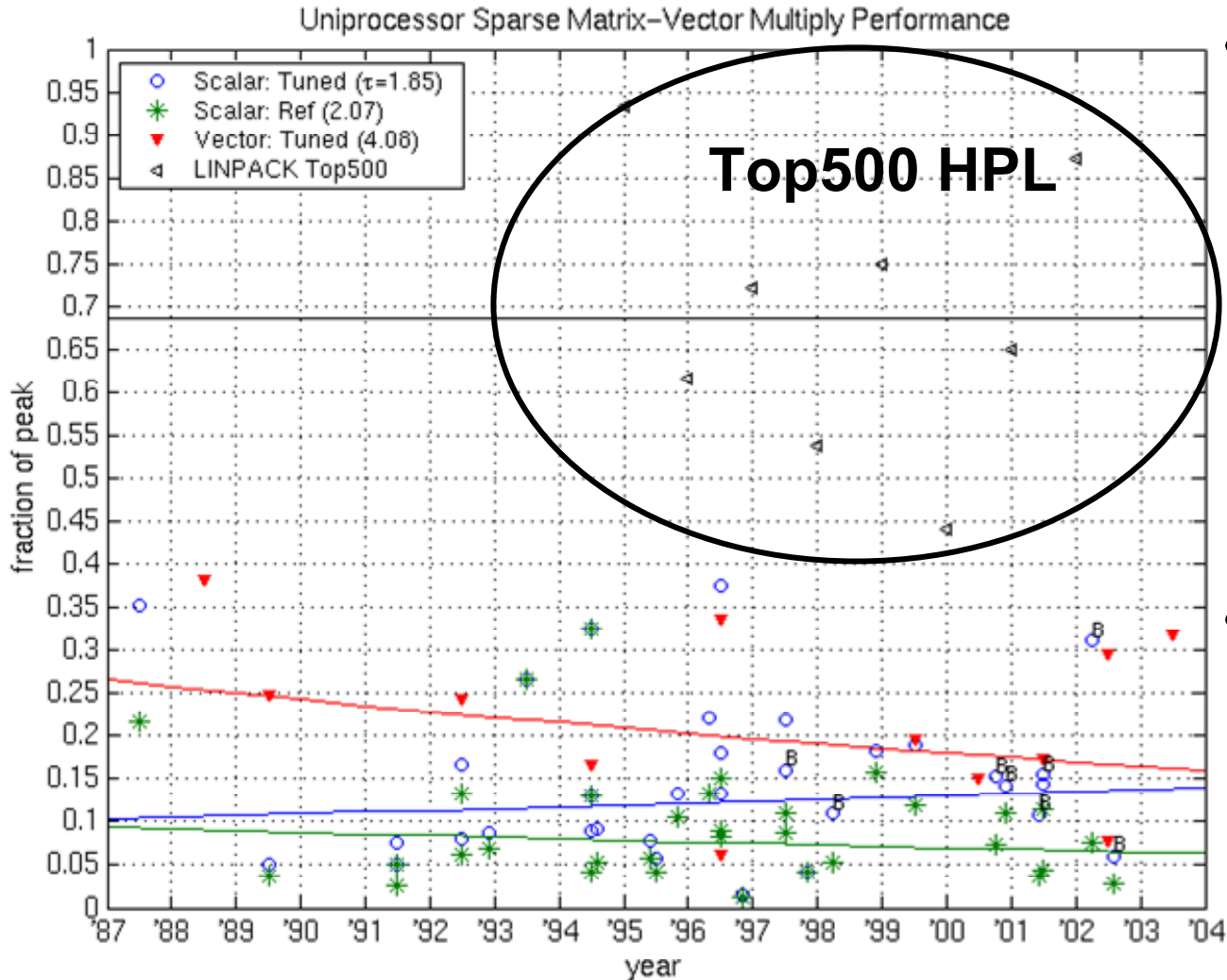
# Uniprocessor Sparse Matrix-Vector Multiply Performance



Source: R. Vuduc, J. Demmel, K. Yelick, UC Berkeley



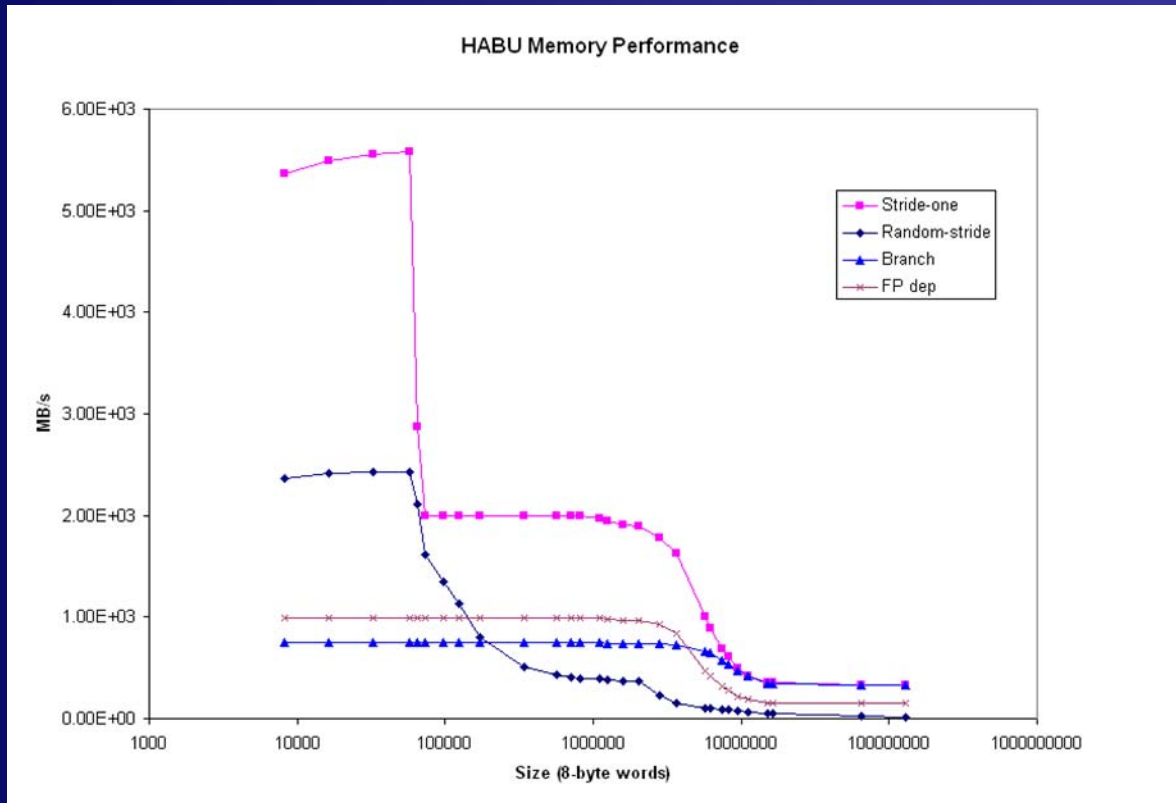
# Uniprocessor Sparse Matrix-Vector Multiply Performance



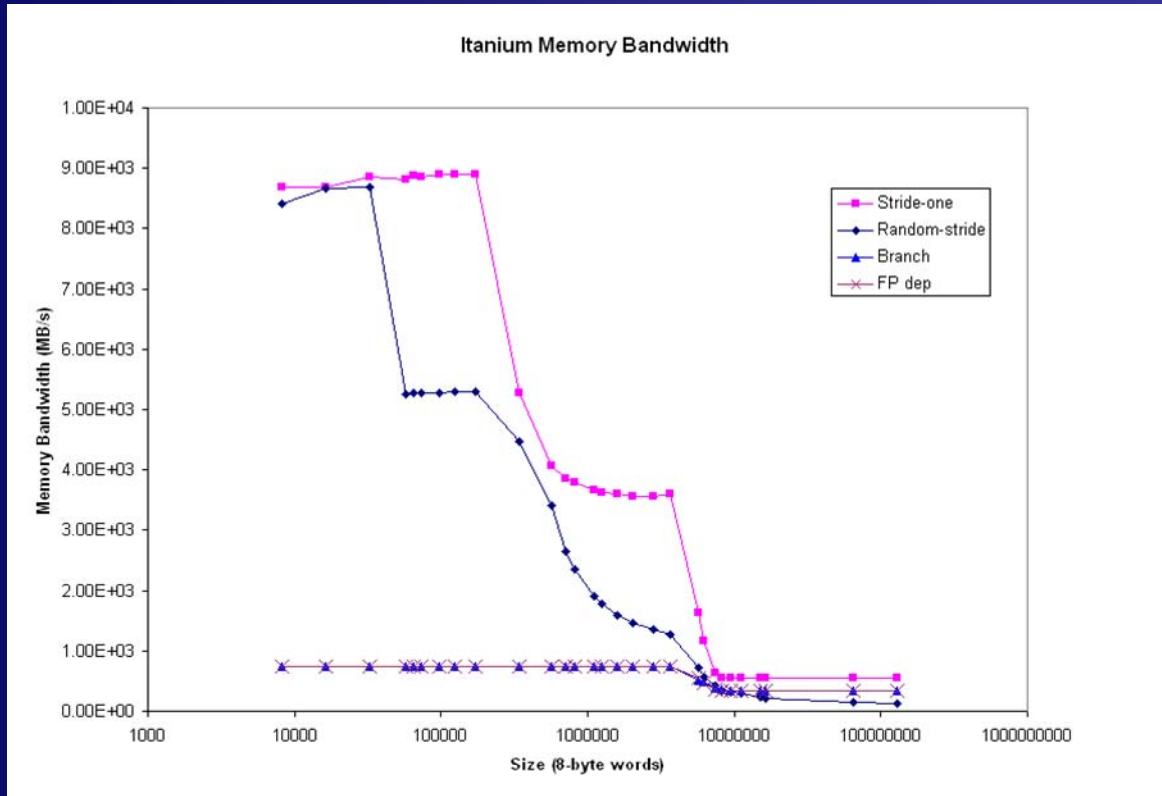
- **HPL — dense linear solver**
  - High temporal locality or data reuse due to blocked data
  - Architecture able to move data to processors to keep them busy
- **Sparse linear solvers**
  - Difficulties in keeping data moving to the processors to keep them busy

Source: R. Vuduc, J. Demmel, K. Yelick, UC Berkeley

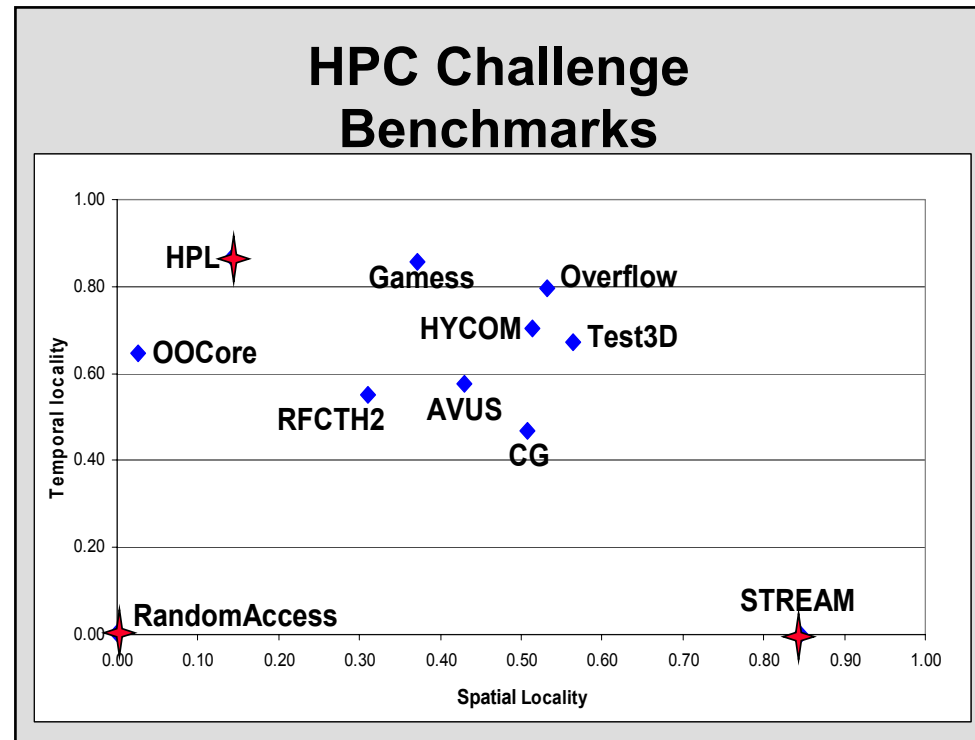
## Framework addition: Data Dependency



## Framework addition: Data Dependency



Generated by PMaC @ SDSC



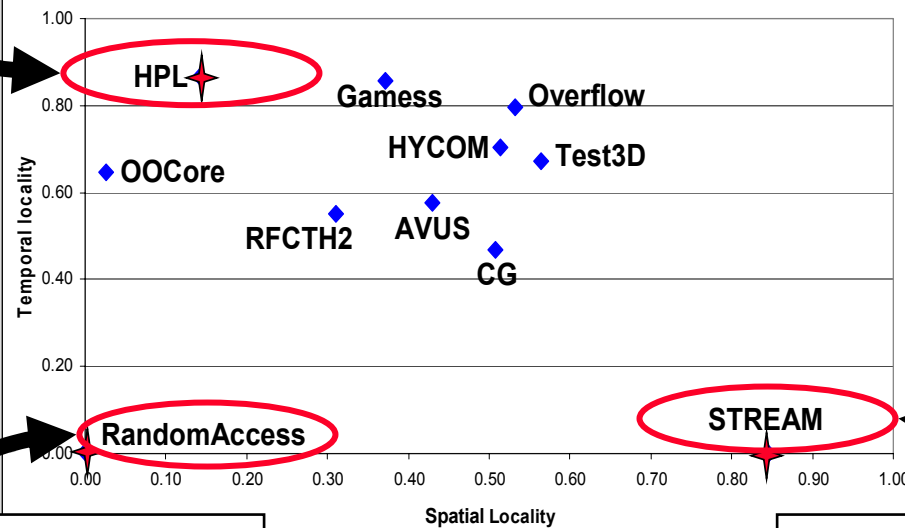
- Spatial and temporal data locality here is for one node/processor — i.e., locally or “in the small”

# Node Spatial and Temporal Locality

Generated by PMaC @ SDSC

High Temporal Locality  
Good Performance on  
Cache-based systems

## HPC Challenge Benchmarks

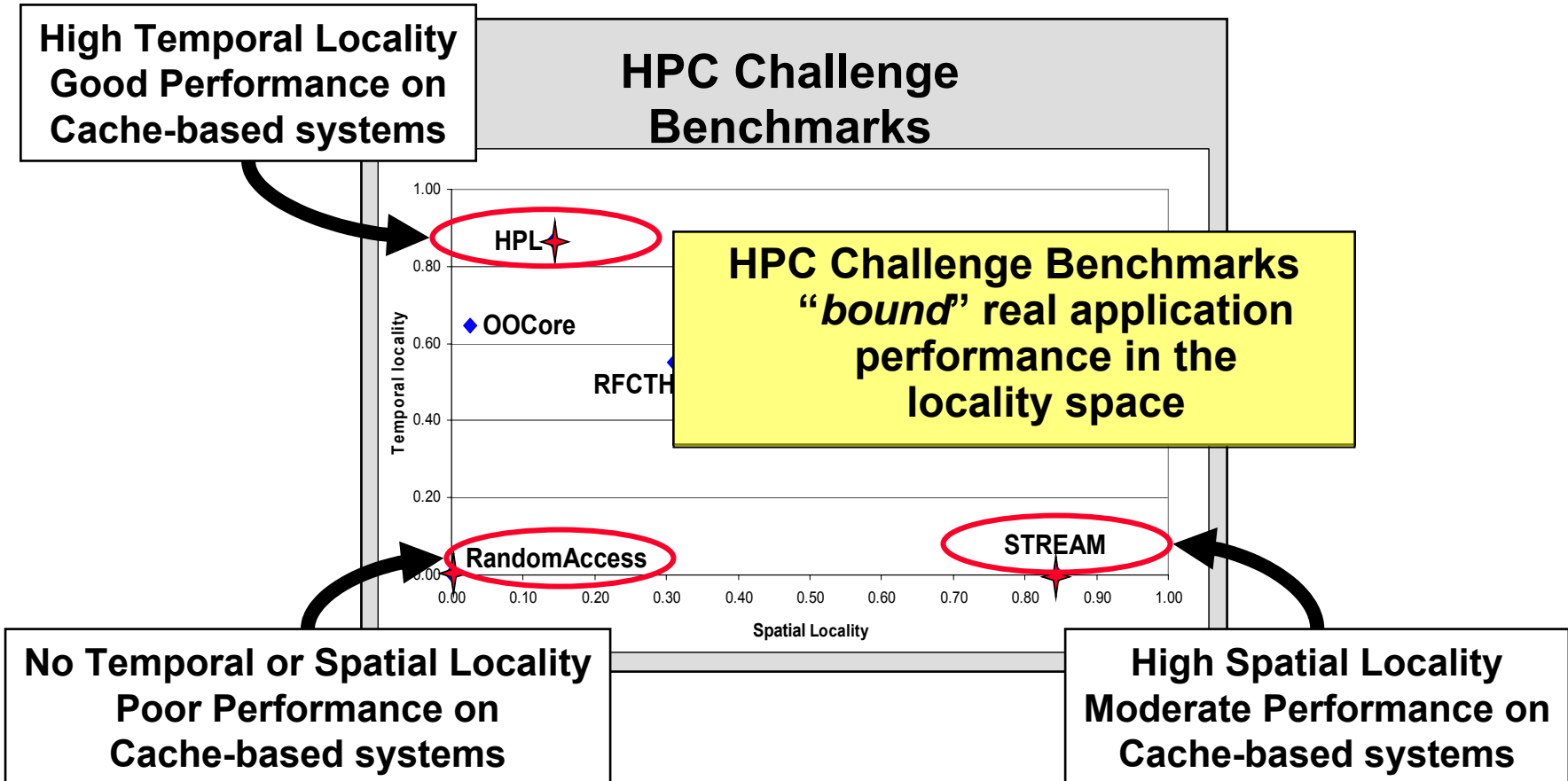


No Temporal or Spatial Locality  
Poor Performance on  
Cache-based systems

High Spatial Locality  
Moderate Performance on  
Cache-based systems

# Node Spatial and Temporal Locality

Generated by PMaC @ SDSC



# 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  $n$ :

$$Ax = b; \quad A \in \mathbf{R}^{n \times n}; \quad x, b \in \mathbf{R}^n$$

- 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:

$$Ux = y.$$

# HPC Challenge v1.x Kernels

## HPL (2 of 3)

- 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  $\varepsilon$  is machine precision for 64-bit floating-point values and  $n$  is the size of the problem

$$\frac{\|Ax - b\|_{\infty}}{\varepsilon \|A\|_1 n},$$
$$\frac{\|Ax - b\|_{\infty}}{\varepsilon \|A\|_1 \|x\|_1}, \quad \text{and}$$
$$\frac{\|Ax - b\|_{\infty}}{\varepsilon \|A\|_{\infty} \|x\|_{\infty}},$$

# 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 ( $P$ )

$$r_{\text{HPL}} \propto P$$

- It can also be shown that the time ( $t$ ) to run HPL is proportional to the square root of the number of processors

$$t_{\text{HPL}} \propto \sqrt{P}$$

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:

$$A, B, C \in \mathbf{R}^{n \times n}; \quad \alpha, \beta \in \mathbf{R}.$$

- The operation count is —  $(2n^3)$
- Correctness is ascertained by calculating the scaled residual:

$$\|C - \hat{C}\| / (\epsilon n \|C\|_F)$$

( $\hat{C}$  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:  $b \leftarrow \alpha c$

ADD:  $c \leftarrow a + b$

TRIAD:  $a \leftarrow b + \alpha c$

where:

$$a, b, c \in \mathbf{R}^m; \quad \alpha \in \mathbf{R}.$$

- **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.

# HPC Challenge v1.x Kernels

## STREAM (2 of 2)

- 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

More at <http://www.cs.virginia.edu/stream/>

# HPC Challenge v1.x Kernels

## PTRANS

- 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  $n$  by  $n$  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:

$$\|A - \hat{A}\| / (\epsilon n)$$

More at <http://www.netlib.org/parkbench/html/matrix-kernels.html>

# HPC Challenge v1.x Kernels

## RandomAccess (1 of 2)

- **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)

$$r_{RA} \propto 1/P$$

More at <http://icl.cs.utk.edu/projectsfiles/hpcc/RandomAccess/>

# HPC Challenge v1.x Kernels

## FFT

- 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 :

$$Z_k \leftarrow \sum_j^m z_j e^{-2\pi i \frac{jk}{m}}; \quad 1 \leq k \leq m$$

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:

$$\|x - \hat{x}\| / (\epsilon \log m)$$

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/](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  $\frac{1}{2}$  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

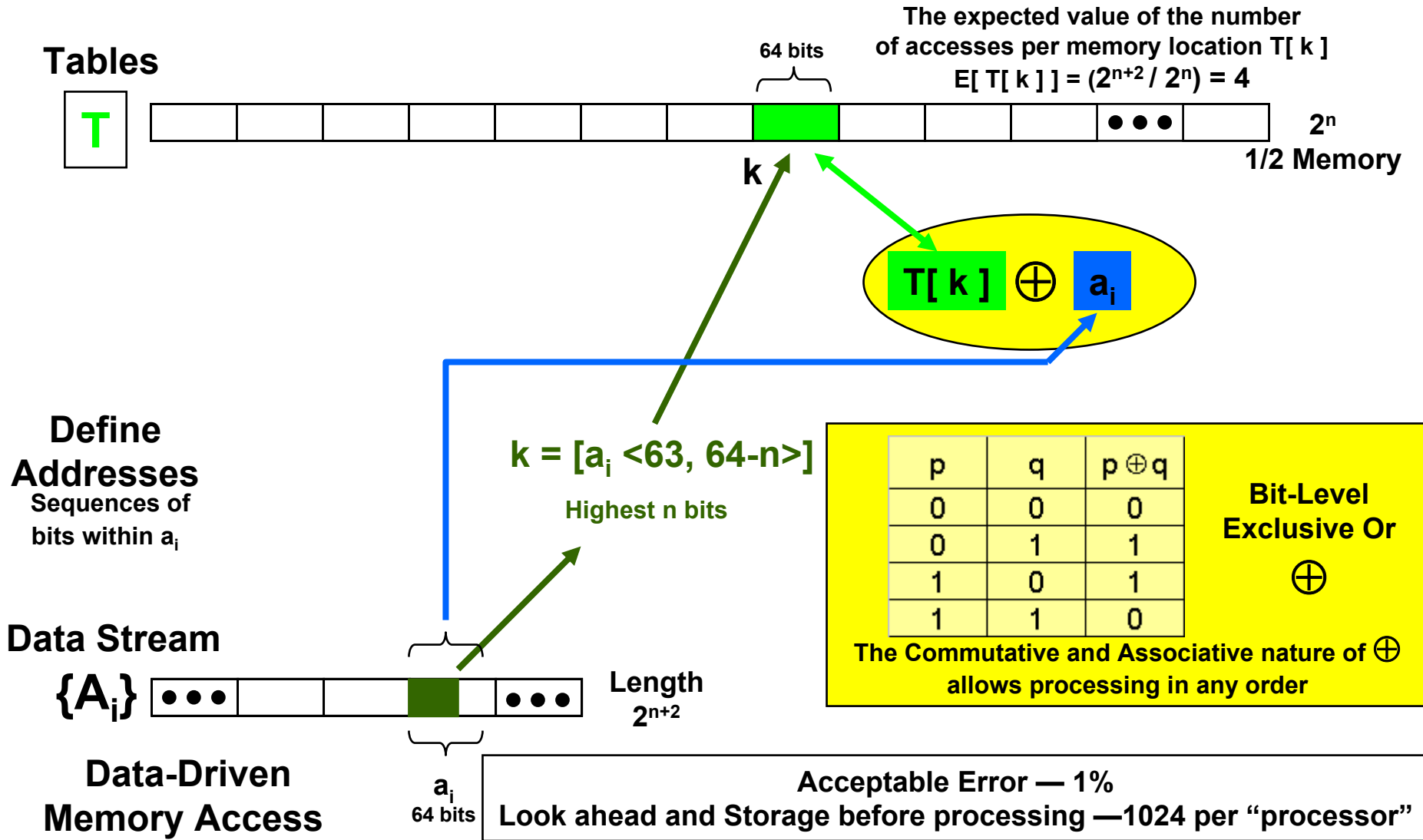
- Let  $T$  be a table of size  $2^n$  filled with random 64-bit integers
- Let  $\{A_i\}$  be a stream of 64-bit integers of length  $2^{n+2}$  generated by the primitive polynomial over  $GF(2)$ ,  $X^{63} + X^3 + X + 1$ 
  - $GF(2)$  (Galois Field of order 2)
  - The elements of  $GF(2)$  can be represented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

$p$	$q$	$p \oplus q$
0	0	0
0	1	1
1	0	1
1	1	0

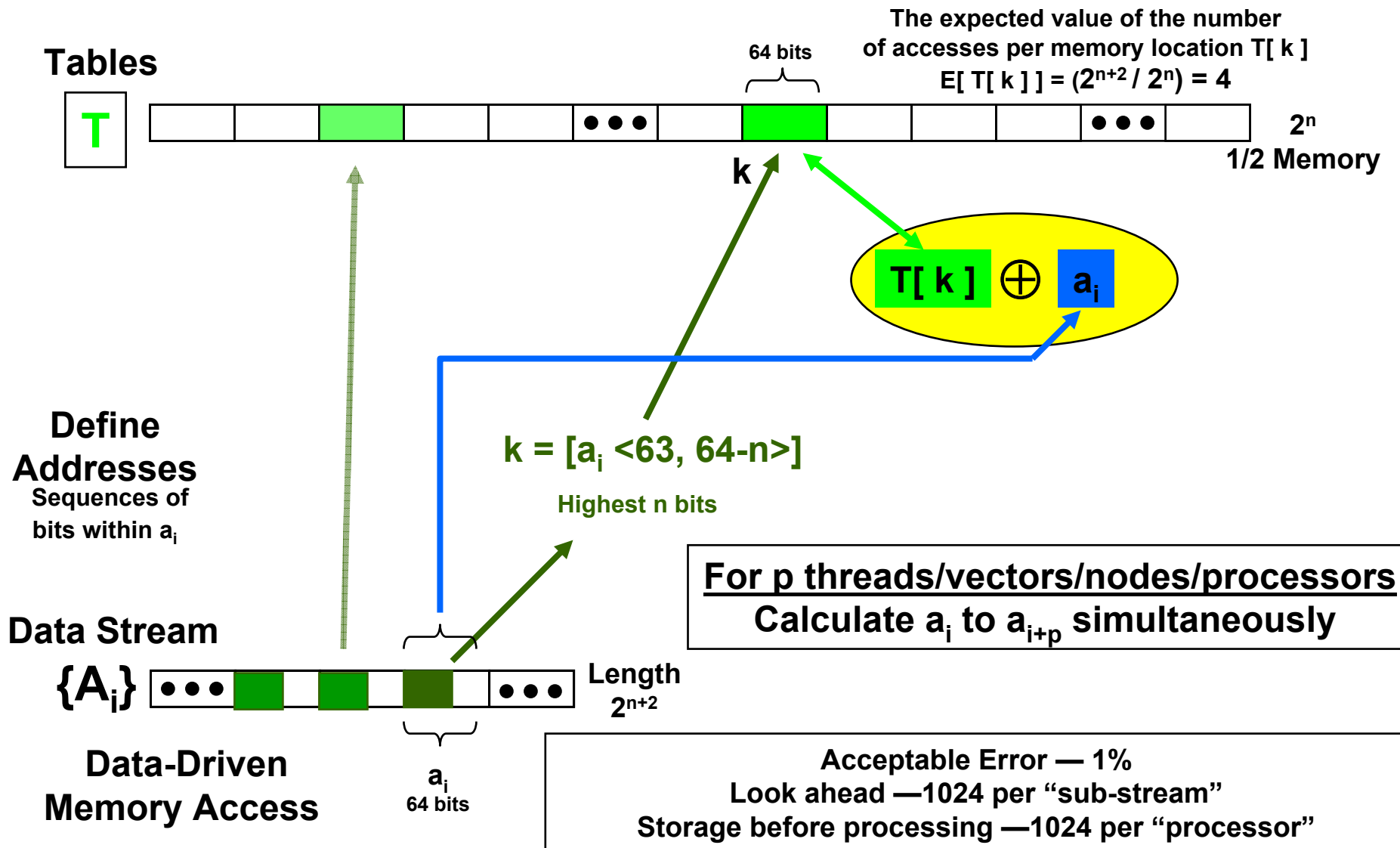
Bit-Level Exclusive Or  
 $\oplus$

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

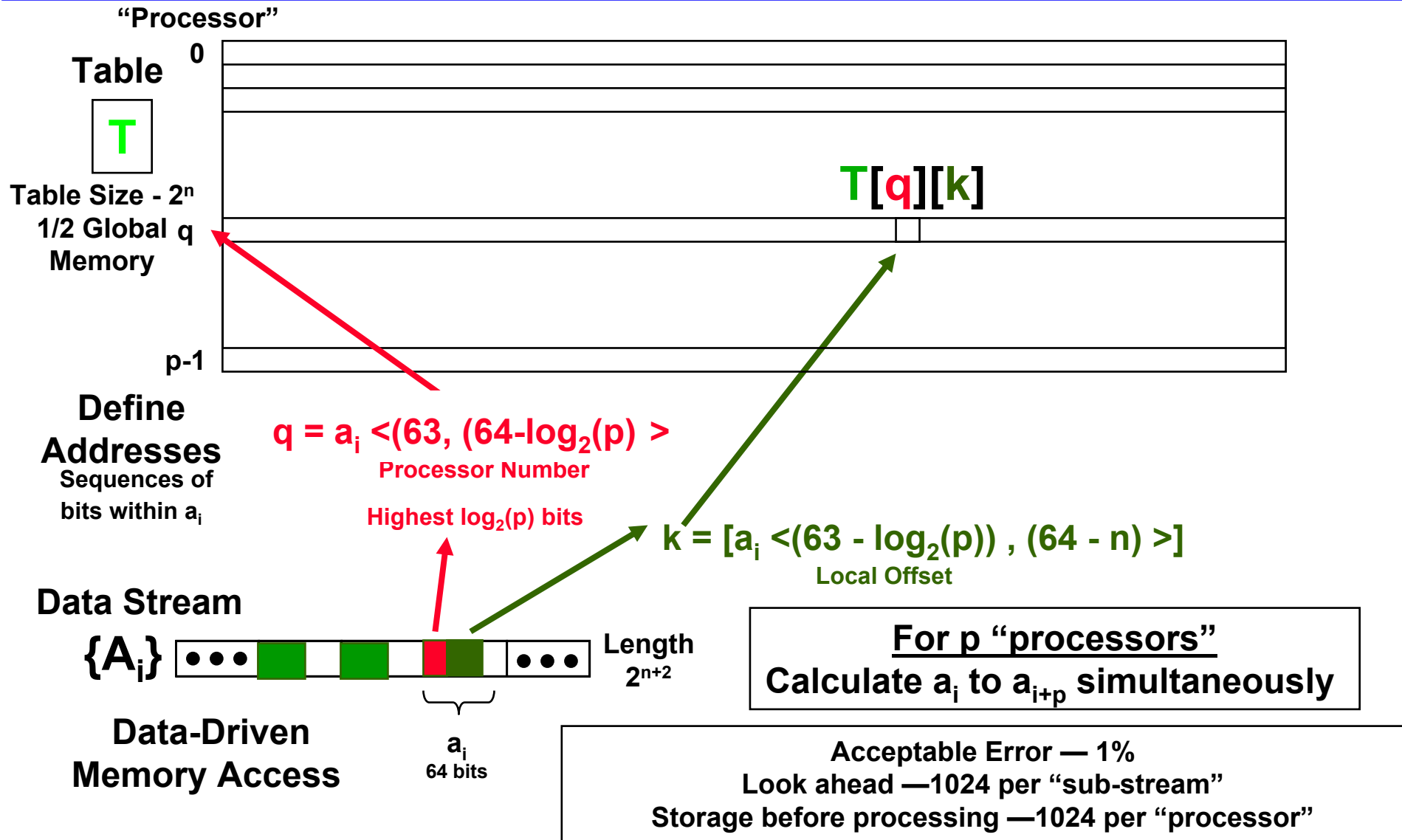
# Sequential RandomAccess Implementation



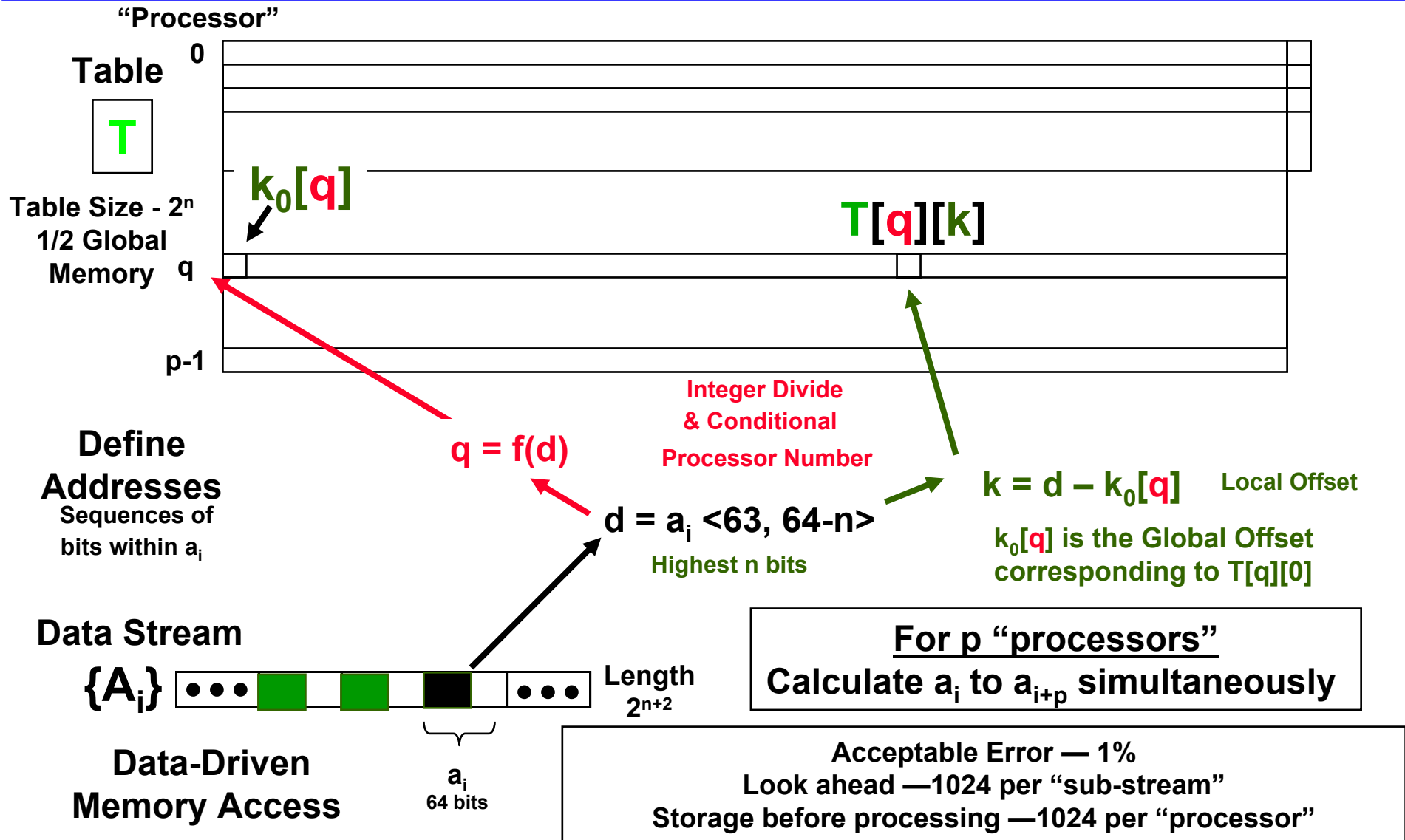
# Global Address Space (GAS) G-RandomAccess Implementation



# Distributed Memory G-RandomAccess Implementation — $p = 2^m$



# Distributed Memory G-RandomAccess Implementation — $p \neq 2^m$





# 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 Award Competition



- 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



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

- The prize will be \$500 plus a certificate for the best of each benchmark
- 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 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**

- **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**

- **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**



- **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

# Rules Baseline Runs

## 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

# Rules

## Optimizations — Code Modifications

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**

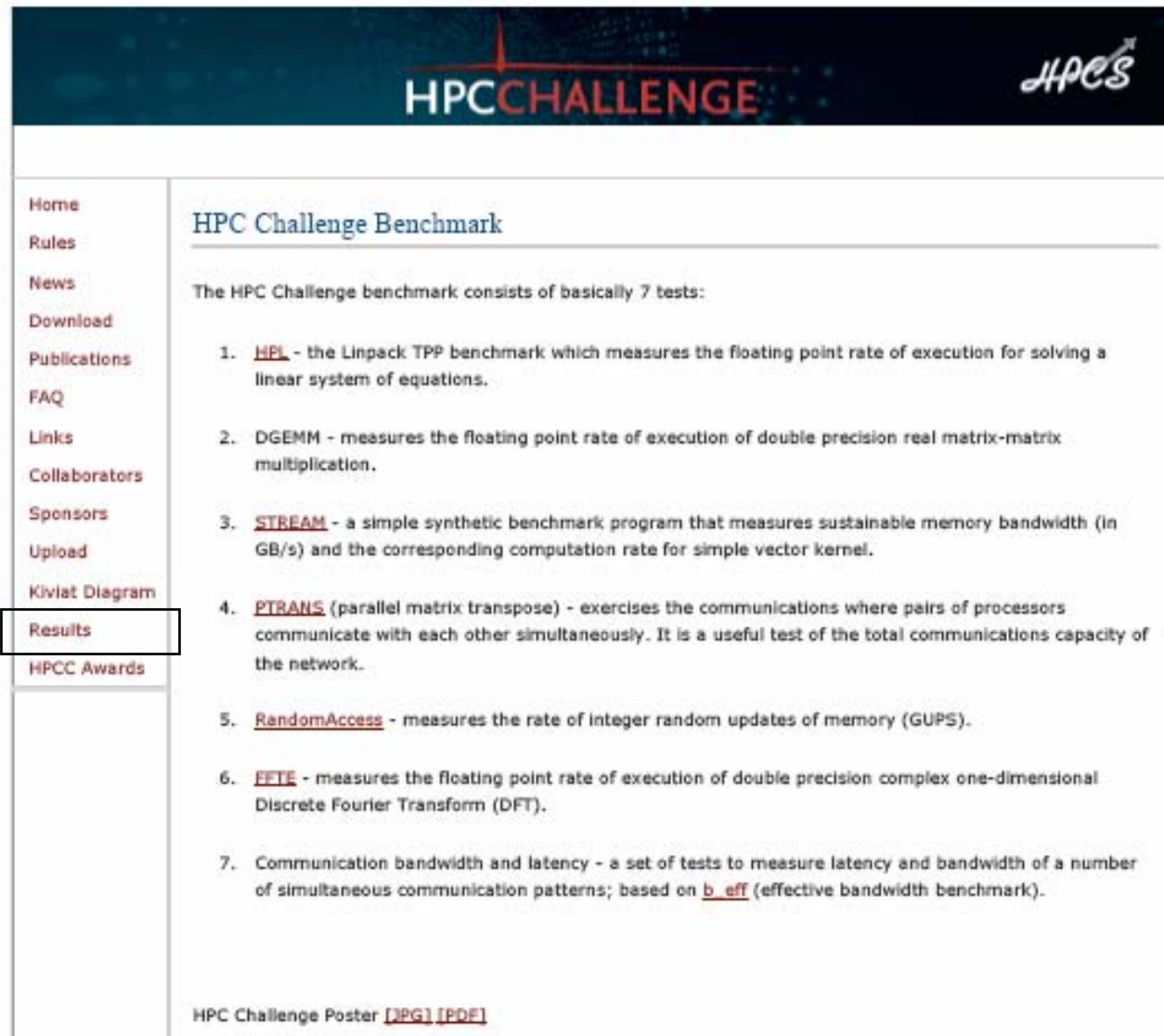
- **The HPC Challenge Benchmark suite has been designed to permit academic style usage for comparing**
  - Technologies
  - Architectures
  - Programming models
- **There is an overt attempt to keep HPC Challenge significantly different than “commercialized” benchmark suites**
  - Vendors and users can 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**





**HPCCHALLENGE**

*HPCC*

Home  
Rules  
News  
Download  
Publications  
FAQ  
Links  
Collaborators  
Sponsors  
Upload  
Kivlat Diagram  
**Results**  
HPC Awards

## 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. [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).

HPC Challenge Poster [\[JPG\]](#) [\[PDF\]](#)

- **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 kiviart 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



[Home](#) | [Rules](#) | [News](#) | [Download](#) | [FAQ](#) | [Links](#) | [Collaborators](#) | [Sponsors](#) | [Upload](#) | [Results](#)

**80 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 602-604 of the conference center. We are of course still accepting submissions to the HPC Challenge Class 1 Awards up until Sunday November 13th, 2005.

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

System Information				Run	G-HPL	G-PTRANS	G-Random Access	G-FFTE	G-STREAM Triad	EP-STREAM Triad	EP-DGEMM	RandomRing Bandwidth	RandomRing Latency
System - Processor - Speed - Count - Threads - Processes	MA/PT/PS/PC/TH/PR/CM/CS/IC/IA/SD	Type	TFlop/s	GB/s	Gup/s	GFlop/s	GB/s	GB/s	GB/s	GFlop/s	GB/s	usec	
Atipa Conquest cluster AMD Opteron	1.4GHz 128 1 128	base	0.2526110	3.2471				208.525	1.6291			0.03627	23.68
Clustervision BV Beastie AMD Opteron	2.4GHz 32 1 32	base	0.1037640	0.8159	0.0002350	2.15		106.951	3.3422	4.19493		0.02648	53.23
Cray X1 MSP	0.8GHz 64 1 64	base	0.5215600	3.2288				959.334	14.9896			0.94074	20.34
Cray X1 MSP	0.8GHz 60 1 60	base	0.5777790	30.4313				898.446	14.9741			1.03291	20.83
Cray X1 MSP	0.8GHz 120 1 120	base	1.0609700	2.4603				1019.519	8.4960			0.83014	20.12
Cray T3E Alpha 21164	0.6GHz 1024 1 1024	base	0.0481695	10.2765				529.242	0.5168			0.03174	12.09
Cray X1 MSP	0.8GHz 252 1 252	base	2.3847300	97.4076				3758.404	14.9143			0.42899	22.27
Cray X1 MSP	0.8GHz 252 1 252	opt	2.3678200	96.1372				5478.732	21.7410			0.43828	22.64
Cray X1 MSP	0.8GHz 60 1 60	opt	0.5788740	31.0723				1306.080	21.7680			1.00986	21.16
Cray X1 MSP	0.8GHz 124 1 124	base	1.2054200	39.5252				1856.664	14.9731			0.70857	20.15
Cray X1 MSP	0.8GHz 124 1 124	opt	1.1820300	39.3826				2697.260	21.7521			0.80388	20.85

# HPC 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 602-604 of the conference center. We are of course still accepting submissions to the HPC Challenge Class 1 Awards up until Sunday November 13th, 2005.

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

System Information				G-HPL	G-PTRANS	G-Random Access	G-FFTE	G-STREAM Triad	EP-STREAM Triad	EP-DGEMM	RandomRing Bandwidth	RandomRing Latency
System - Processor - Speed - Count - Threads - Processes				TFlop/s	GB/s	Gup/s	GFlop/s	GB/s	GB/s	GFlop/s	GB/s	usec
MA/PT/PS/PC/TH/PR/CM/CS/IC/IA/SD												
Atipa Conquest cluster AMD Opteron	1.4GHz	128	1 128	0.2526110	3.2471			208.525	1.6291		0.03627	23.68
Clustervision BV Beastie AMD Opteron	2.4GHz	32	1 32	0.1037640	0.8159	0.0002350	2.1470	106.951	3.3422	4.19493	0.02648	53.23
Cray X1 MSP	0.8GHz	64	1 64	0.5215600	3.2288			959.334	14.9896		0.94074	20.34
Cray X1 MSP	0.8GHz	60	1 60	0.5777790	30.4313			898.446	14.9741		1.03291	20.83
Cray X1 MSP	0.8GHz	120	1 120	1.0609700	2.4603			1019.519	8.4960		0.83014	20.12
Cray T3E Alpha 21164	0.6GHz	1024	1 1024	0.0481695	10.2765			529.242	0.5168		0.03174	12.09
Cray X1 MSP	0.8GHz	252	1 252	2.3847300	97.4076			3758.404	14.9143		0.42899	22.27
Cray X1 MSP	0.8GHz	124	1 124	1.2054200	39.5252			1856.664	14.9731		0.70857	20.15
Cray X1 MSP	0.8GHz	60	1 60	0.5087430	1.6342	0.0030750	3.1444	894.114	14.9019	10.91520	1.16779	14.66
Cray T3E Alpha 21164	0.675GHz	512	1 512	0.2231810	9.7741	0.0289464	15.4774	272.186	0.5316	0.66077	0.03571	8.14

# HPC Submissions Optimized Results



## 6 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 602-604 of the conference center. We are of course still accepting submissions to the HPC Challenge Class 1 Awards up until Sunday November 13th, 2005.

Condensed Results - Optimized Runs Only - 6 Systems - Generated on Tue Nov 1 14:57:00 2005

System Information				G-HPL	G-PTRANS	G-Random Access	G-FFTE	G-STREAM Triad	EP-STREAM Triad	EP-DGEMM	RandomRing Bandwidth	RandomRing Latency
System - Processor	Speed - Count	Threads - Processes		TFlop/s	GB/s	Gup/s	GFlop/s	GB/s	GB/s	GFlop/s	GB/s	usec
MA/PT/PS/PC/TH/PR/CM/CS/IC/IA/SD												
Cray X1 MSP	0.8GHz	252 1 252		2.368	96.1			5479	21.741		0.4383	22.64
Cray X1 MSP	0.8GHz	60 1 60		0.579	31.1			1306	21.768		1.0099	21.16
Cray X1 MSP	0.8GHz	124 1 124		1.182	39.4			2697	21.752		0.8039	20.85
Cray X1 MSP	0.8GHz	124 1 124		1.182	39.4			2697	21.752		0.8039	20.85
Cray mfg8 X1E	1.13GHz	248 1 248		3.389	66.0	1.855	-1.00	3281	13.229	13.56	0.2989	14.58
IBM Blue Gene/L PowerPC 440	0.7GHz	1024 1 1024		1.420	28.0	0.135	49.93	863	0.843	2.47	0.0346	4.83

# HPCC Submissions Display

System Information					Run	G-HPL	G-PTRANS	G-Random Access
System - Processor - Speed - Count - Threads - Processes								
MA/PT/PS/PC/TH/PR/CM/CS/IC/IA/SD					Type	TFlop/s	GB/s	Gup/s
Cray XT3 AMD Opteron		2.6GHz	1100	1 1100	base	4.7823400	217.9230	0.1370020
Cray mfeg8 X1E		1.13GHz	248	1 248	opt	3.3888700	66.0098	1.8547500
Cray XD1 AMD Opteron		2.4GHz	128	1 128	base	0.5020760	13.5155	0.0666722
Cray X1E X1E MSP		1.13GHz	252	1 252	base	3.1940900	85.2040	0.0148684
Cray XT3 AMD Opteron		2.4GHz	3744	1 3744	base	14.7040000	608.5060	0.2202960
Cray XT3 AMD Opteron		2.4GHz	5200	1 5200	base	20.5270000	874.8990	0.2685830
Cray xt3 AMD Opteron		2.4GHz	32	1 32	base	0.1387810	7.3764	0.0606017
Cray X1E		1.13GHz	32	4 32	base	0.3376360	18.9199	0.0089686
Cray XT3 AMD Opteron		2.6GHz	4096	1 4096	base	16.9752000	302.9790	0.5330720



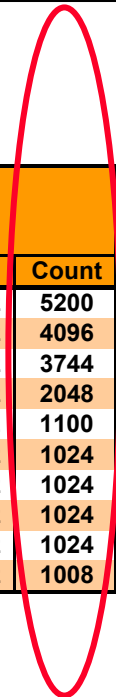
**Additional “cool” features will be discussed in the conference hands-on session!**

**Lines Depict Relative Performance**

# HPC Challenge Benchmark Suite Selected Results

## 1000+ Processor Systems

As of 1 November 2005



System Information			G-HPL	G-PTRANS	G-Random	G-FFTE	G-STREAM	EP	EP	Random	Random
System - Processor	Speed	Count	TFlop/s	GB/s	Access	GFlop/s	Triad	STREAM	DGEMM	Ring	Ring
					Gup/s		GB/s	Triad	GFlop/s	Bandwidth	Latency
								GB/s		GB/s	usec
Cray XT3 AMD Opteron	2.4GHz	5200	20.527	874.899	0.268583	644.73	26020.8	5.004	4.395	0.14682	25.8
Cray XT3 AMD Opteron	2.6GHz	4096	16.9752	302.979	0.533072	905.57	20656.5	5.043	4.782	0.16896	9.44
Cray XT3 AMD Opteron	2.4GHz	3744	14.704	608.506	0.220296	417.17	18146.4	4.847	4.413	0.16164	25.32
IBM Blue Gene PowerPC 440	0.7GHz	2048	1.4075	34.251	0.454092	96.19	1484.6	0.725	0.905	0.02089	4.98
Cray XT3 AMD Opteron	2.6GHz	1100	4.7823	217.923	0.137002	266.66	5274.7	4.795	4.811	0.28638	25.94
Cray T3E Alpha 21164	0.6GHz	1024	0.0482	10.277			529.2	0.517		0.03174	12.09
IBM Blue Gene PowerPC 440	0.7GHz	1024	0.7164	27.578	0.134994	48.99	868.4	0.848	0.919	0.03461	4.81
IBM Blue Gene/L PowerPC 440	0.7GHz	1024	1.4201	27.994	0.134729	49.93	862.9	0.843	2.467	0.03455	4.83
IBM Blue Gene PowerPC 440	0.7GHz	1024	0.7301	26.44	0.299617	70.94	765.3	0.747	0.901	0.0448	4.5
SGI Altix 3700 Intel Itanium 2	1.6GHz	1008	5.1383	105.666	0.032598	15.66	1907.5	1.892	5.884	0.20288	6.82

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

# HPC Challenge Benchmark Suite Selected Results

## Optimized G-RandomAccess

As of 1 November 2005

System Information			Run Type	G-HPL TFlop/s	G-PTRANS GB/s	G-Random Access Gup/s	G-FFTE GFlop/s	G-STREAM Triad GB/s	EP STREAM Triad GB/s	EP DGEMM GFlop/s	Random Ring Bandwidth GB/s	Random Ring Latency usec
System - Processor	Speed	Count										
Cray mfeg8 X1E	1.13GHz	248	opt	3.3889	66.01	1.85475	-1	3280.9	13.229	13.564	0.29886	14.58
Cray X1E X1E MSP	1.13GHz	252	base	3.1941	85.204	0.014868	15.54	2440	9.682	14.185	0.36024	14.93

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

**Be sure to attend the SC|05 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

System Information					G-HPL TFlop/s	G- PTRANS GB/s	G-Random Access Gup/s	G-FFTE GFlop/s	G-STREAM Triad GB/s	EP STREAM Triad GB/s	EP DGEMM GFlop/s	Random Ring Bandwidth GB/s	Random Ring Latency usec
System - Processor	Speed	Count	Tds	Proc									
Cray XT3 AMD Opteron	2.4GHz	5200	1	5200	20.527	874.899	0.268583	644.73	26020.8	5.004	4.395	0.14682	25.8
Cray mfeg8 X1E	1.13GHz	248	1	248	3.3889	66.01	1.85475	-1	3280.9	13.229	13.564	0.29886	14.58
Cray XT3 AMD Opteron	2.6GHz	4096	1	4096	16.9752	302.979	0.533072	905.57	20656.5	5.043	4.782	0.16896	9.44
NEC SX-7	0.552GHz	32	16	2	0.2174	16.34	0.000178	1.34	984.3	492.161	140.636	8.14753	4.85
NEC SX-8/6 SX-8	2GHz	6	1	6	0.0918	25.183	0.000769	3.19	370.6	61.773	15.944	13.5473	3.02
IBM pSeries 655 Power 4+	1.7GHz	256	4	64	1.0744	23.721	0.005502	10.46	411.7	6.433	17.979	0.72395	8.34
PathScale Inc. AMD Opteron	2.6GHz	32	1	32	0.1258	6.719	0.030367	10.35	134.3	4.197	4.775	0.26531	1.31

- 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

System Information					G-HPL	G-	G-Random	G-FFTE	G-STREAM	EP	EP	Random	Random
System - Processor	Speed	Count	Tds	Proc	TFlop/s	PTRANS	Access	GFlop/s	Triad	STREAM	DGEMM	Ring	Ring
						GB/s	Gup/s		GB/s	Triad	GFlop/s	Bandwidth	Latency
										GB/s		GB/s	usec
Cray XT3 AMD Opteron	2.4GHz	5200	1	5200	20.527	874.899	0.268583	644.73	26020.8	5.004	4.395	0.14682	25.8
Cray mfeg8 X1E	1.13GHz	248	1	248	3.3889	66.01	1.85475	-1	3280.9	13.229	13.564	0.29886	14.58
Cray XT3 AMD Opteron	2.6GHz	4096	1	4096	16.9752	302.979	0.533072	905.57	20656.5	5.043	4.782	0.16896	9.44
NEC SX-7	0.552GHz	32	16	2	0.2174	16.34	0.000178	1.34	984.3	492.161	140.636	8.14753	4.85
NEC SX-8/6 SX-8	2GHz	6	1	6	0.0918	25.183	0.000769	3.19	370.6	61.773	15.944	13.5473	3.02
IBM pSeries 655 Power 4+	1.7GHz	256	4	64	1.0744	23.721	0.005502	10.46	411.7	6.433	17.979	0.72395	8.34
PathScale Inc. AMD Opteron	2.6GHz	32	1	32	0.1258	6.719	0.030367	10.35	134.3	4.197	4.775	0.26531	1.31

- 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!!

# HPC Challenge Benchmark Suite

## Threads and Processes

As of 1 November 2005

System Information					G-HPL	G-PTRANS	G-Random	G-FFTE	G-STREAM	EP	EP	Random	Random
System	Speed	Count	Threads	Processes	TFlop/s	GB/s	Access Gup/s	GFlop/s	Triad GB/s	STREAM Triad GB/s	DGEMM GFlop/s	Ring Bandwidth GB/s	Ring Latency usec
NEC SX-7	0.552GHz	32	16	2	0.2174	16.34	0.000178	1.34	984.3	492.161	140.636	8.14753	4.85
NEC SX-7	0.552GHz	32	1	32	0.2553	20.546	0.000964	11.29	836.9	26.154	8.239	5.03934	14.21

- 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

Rank	Manufacturer	System	Processor Type	Processor Speed (GHz)	Processor Count	MPI Processes	HPL (TFlop/s)
1	Cray	Cray XT3	AMD Opteron	2.40	5200	5200	<b>20.53</b>
2	Cray	XT3	AMD Opteron	2.60	4096	4096	<b>16.98</b>
3	Cray	XT3	AMD Opteron	2.40	3744	3744	<b>14.70</b>
4	NEC	NEC SX-8	NEC SX-8	2.00	576	576	<b>8.01</b>
5	SGI	Altix 3700 Bx2	Intel Itanium 2	1.60	1008	1008	<b>5.14</b>
6	Cray	XT3	AMD Opteron	2.60	1100	1100	<b>4.78</b>
7	Cray	mfeg8	Cray X1E	1.13	248	248	<b>3.39</b>
8	Cray	Cray X1E	CrayX1E MSP	1.13	252	252	<b>3.19</b>
9	Cray	X1	Cray X1 MSP	0.80	252	252	<b>2.38</b>
10	Cray	X1	Cray X1 MSP	0.80	252	252	<b>2.37</b>

- **HPC Challenge Awards Class 1**

# Top 10 Performance PTRANS

As of 1 November 2005

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

# Top 10 Performance G-RandomAccess

As of 1 November 2005

Rank	Manufacturer	System	Processor Type	Processor Speed (GHz)	Processor Count	MPI Processes	Global RandomAccess (GUP/s)
1	Cray	mfg8	Cray X1E	1.13	248	248	<b>1.85475</b>
2	Rackable Systems	Emerald	AMD Opteron	2.20	256	512	<b>0.55474</b>
3	Cray	XT3	AMD Opteron	2.60	4096	4096	<b>0.53307</b>
4	IBM	Blue Gene	IBM PowerPC 440	0.70	2048	2048	<b>0.45409</b>
5	Rackable Systems	Emerald	AMD Opteron	2.20	128	256	<b>0.42255</b>
6	Rackable Systems	Emerald	AMD Opteron	2.20	64	128	<b>0.30807</b>
7	IBM	Blue Gene	IBM PowerPC 440	0.70	1024	1024	<b>0.29962</b>
8	Cray	Cray XT3	AMD Opteron	2.40	5200	5200	<b>0.26858</b>
9	Cray	XT3	AMD Opteron	2.40	3744	3744	<b>0.22030</b>
10	Cray	XT3	AMD Opteron	2.60	1100	1100	<b>0.13700</b>

- **HPC Challenge Awards Class 1**

# Top 10 Performance G-FFTE

As of 1 November 2005

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

- **HPC Challenge Awards Class 1**

# Top 10 Performance STREAM Triad (per Process)

As of 1 November 2005

Rank	Manufacturer	System	Processor Type	Processor Speed (GHz)	Processor Count	MPI Processes	EP STREAM Triad	
							per Process	per System
1	NEC	NEC SX-7	NEC SX-7	0.552	32	2	<b>492.161</b>	984.322
2	NEC	SX-8/6	NEC SX-8	2.000	6	6	<b>61.7735</b>	370.641
3	NEC	NEC SX-8	NEC SX-8	2.000	576	576	<b>40.8954</b>	23555.7504
4	NEC	NEC SX-6+	NEC SX-6	0.563	32	32	<b>28.6168</b>	915.7376
5	NEC	SX-6	NEC SX-6	0.500	64	64	<b>27.0884</b>	1733.6576
6	NEC	SX-6	NEC SX-6	0.500	128	128	<b>26.8584</b>	3437.8752
7	NEC	SX-6	NEC SX-6	0.500	32	32	<b>26.8319</b>	858.6208
8	NEC	SX-6	NEC SX-6	0.500	192	192	<b>26.3087</b>	5051.2704
9	NEC	NEC SX-7	NEC SX-7	0.552	32	32	<b>26.1539</b>	836.9248
10	Cray	X1	Cray X1 MSP	0.800	60	60	<b>21.768</b>	1306.08





# Top 10 Performance STREAM Triad (per System)

As of 1 November 2005

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



- **HPC Challenge Awards Class 1**

# Top 10 Performance DGEMM (per Process)

As of 1 November 2005

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



# Top 10 Performance RandomRing Latency

As of 1 November 2005

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



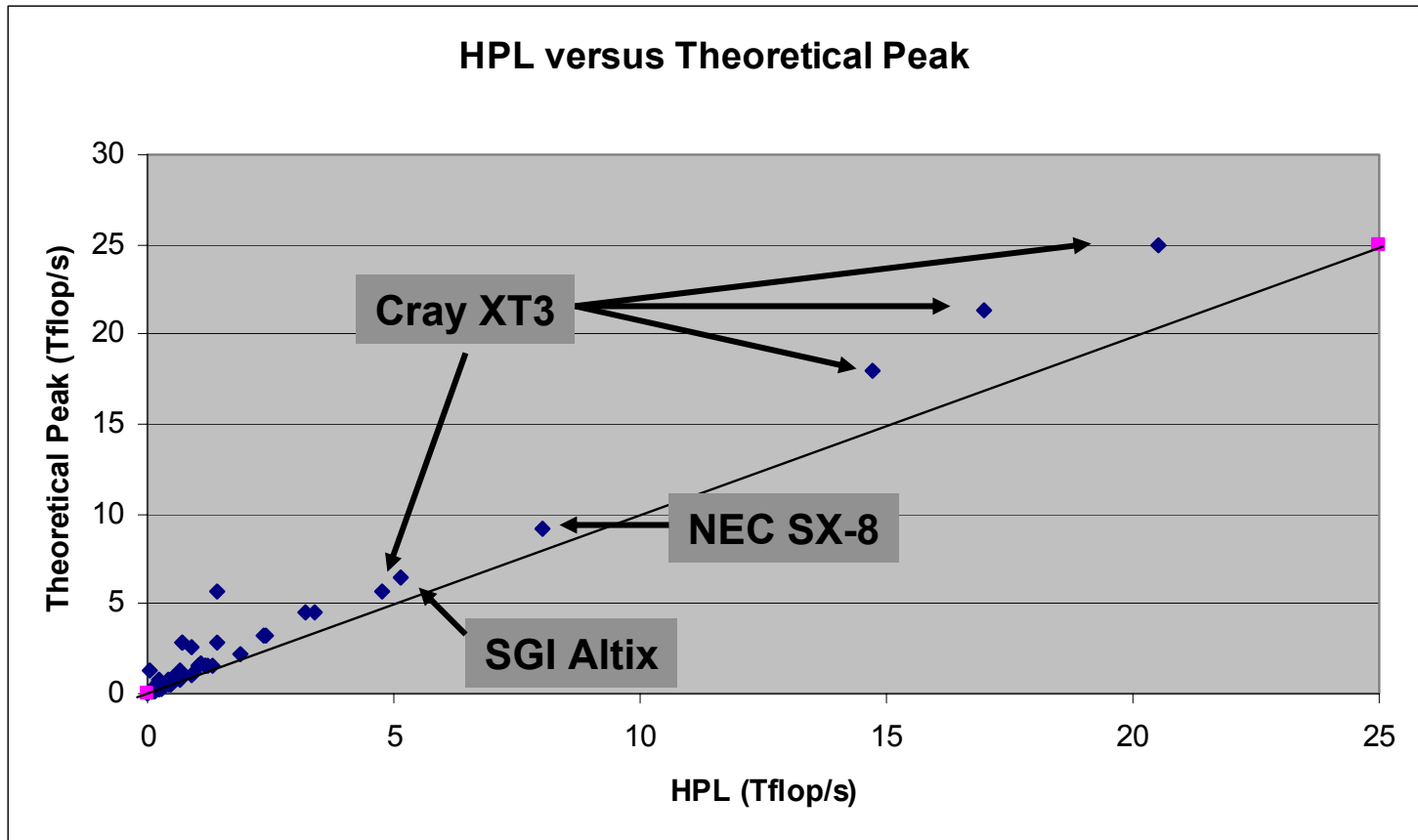
# Top 10 Performance RandomRing Bandwidth

As of 1 November 2005

Rank	Manufacturer	System	Processor Type	Processor Speed (GHz)	Interconnect	Processor Count	MPI Processes	RandomRing	
								Latency (usec)	Bandwidth (GB/s)
1	NEC	SX-8/6	NEC SX-8	2.000	Internode Crossbar Switch	6	6	3.02	<b>13.55</b>
2	NEC	NEC SX-7	NEC SX-7	0.552	non	32	2	4.85	<b>8.15</b>
3	NEC	NEC SX-7	NEC SX-7	0.552	non	32	32	14.21	<b>5.04</b>
4	SGI	Altix 3700 Bx2	Intel Itanium 2	1.600	N/A	32	32	3.26	<b>1.52</b>
5	Cray	X1	Cray X1 MSP	0.800	Cray modified 2-D Torus	32	32	14.94	<b>1.41</b>
6	Cray	X1E	Cray X1E	1.130	Cray Interconnect	32	32	12.21	<b>1.40</b>
7	Cray	X1	Cray X1 MSP	0.800	Cray modified 2-D Torus	60	60	14.66	<b>1.17</b>
8	Cray	X1	Cray X1 MSP	0.800	Cray modified 2D torus	60	60	20.83	<b>1.03</b>
9	Cray	X1	Cray X1 MSP	0.800	Cray modified 2D torus	60	60	21.16	<b>1.01</b>
10	Cray	X1	Cray X1 MSP	0.800	Cray modified 2D torus	64	64	20.34	<b>0.94</b>

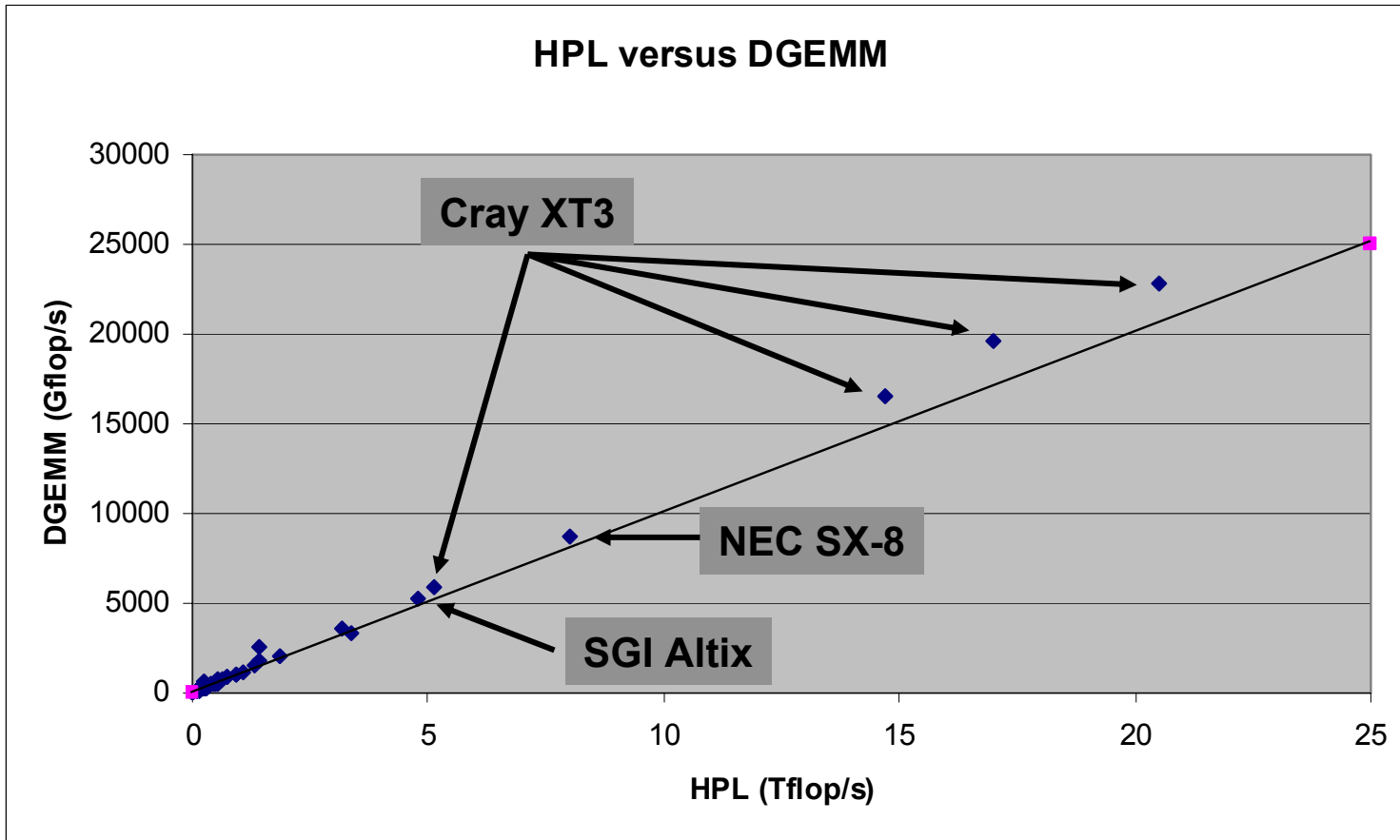


# HPL versus Theoretical Peak



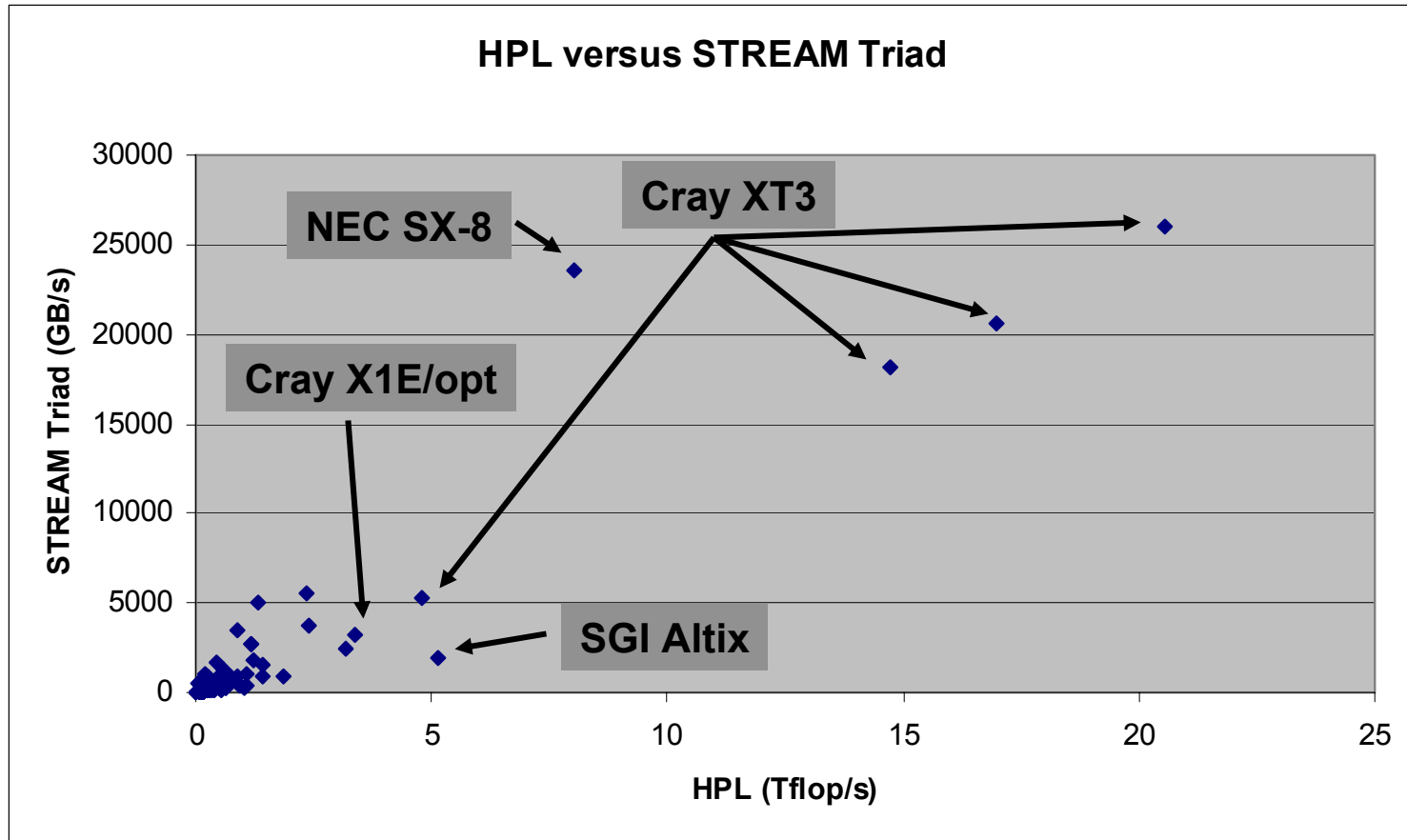
- How well does HPL data correlate with theoretical peak performance?

# HPL versus DGEMM



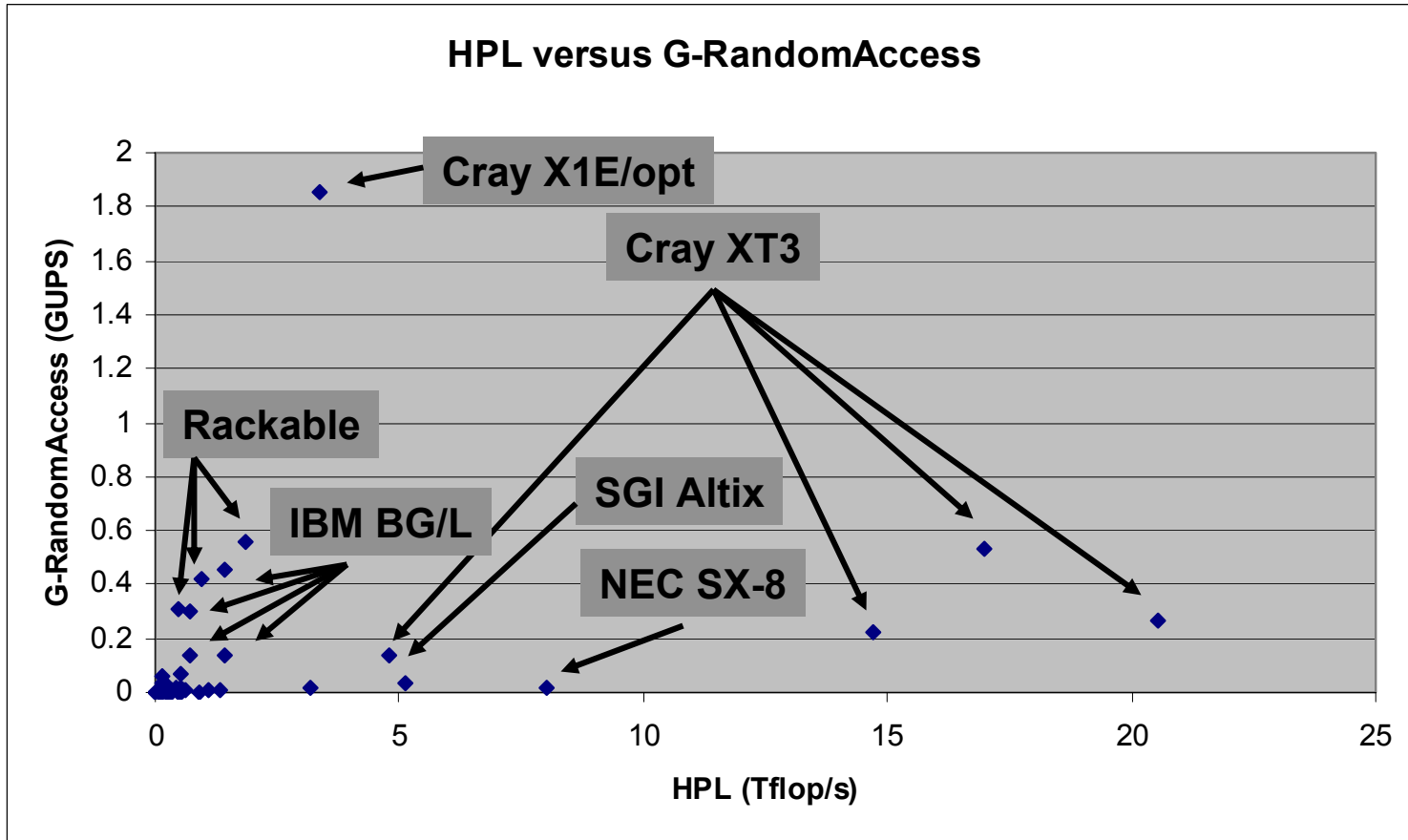
- Can I Run Just Run DGEMM Instead of HPL?
- DGEMM alone overestimates HPL performance
- Note the 1,000x difference in scales! (Tera/Giga)

# HPL versus STREAM Triad



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

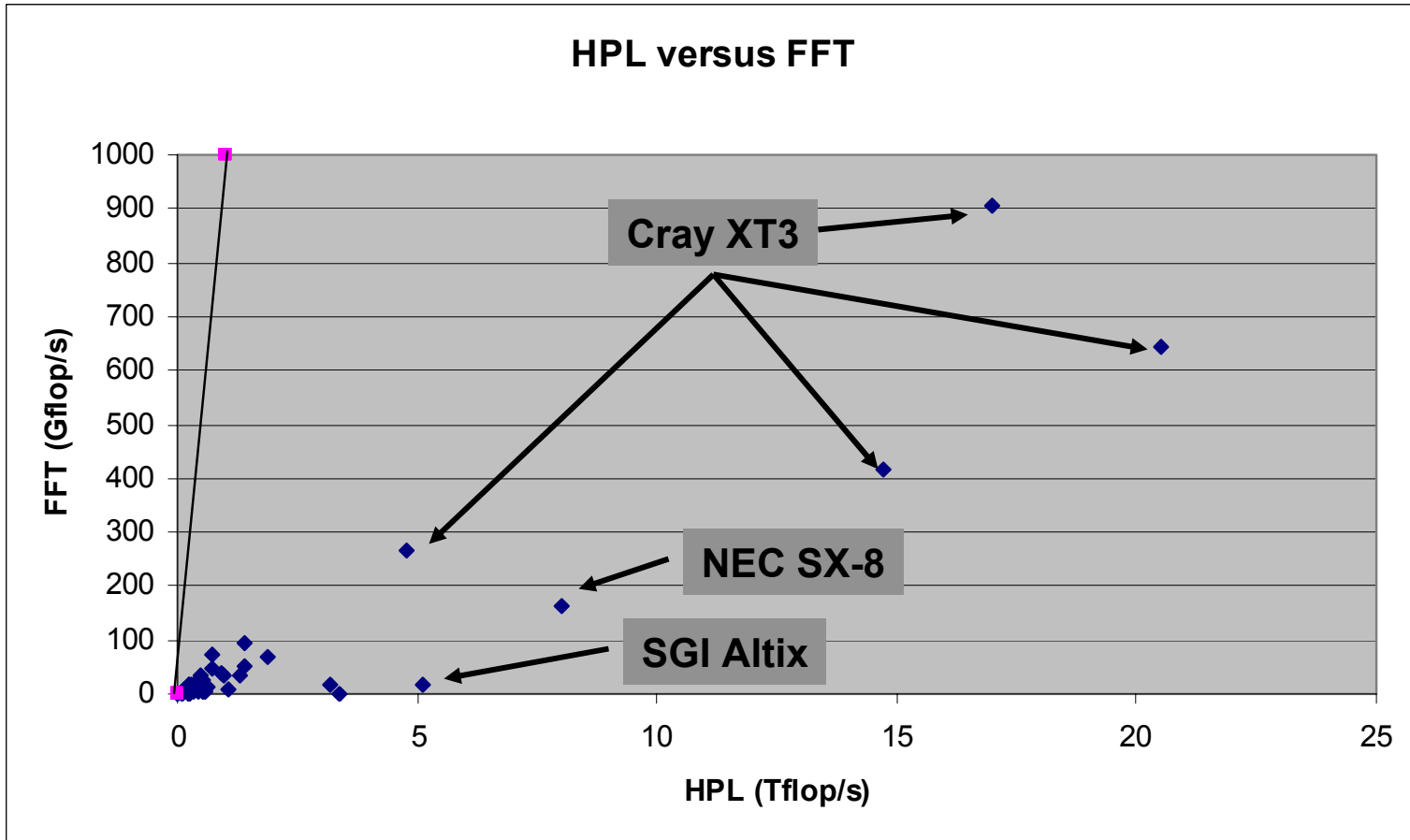
# HPL versus RandomAccess



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

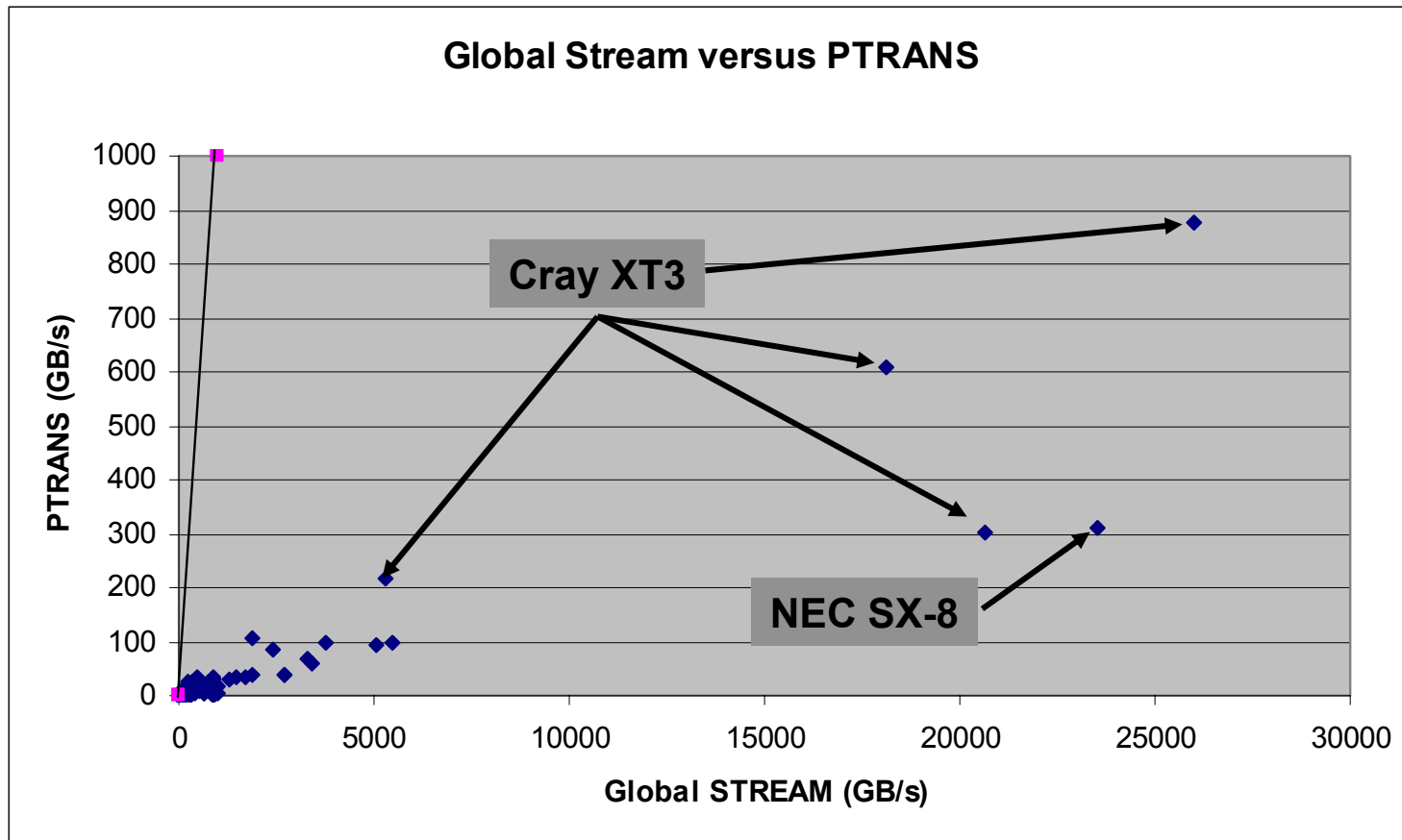


# HPL versus FFT



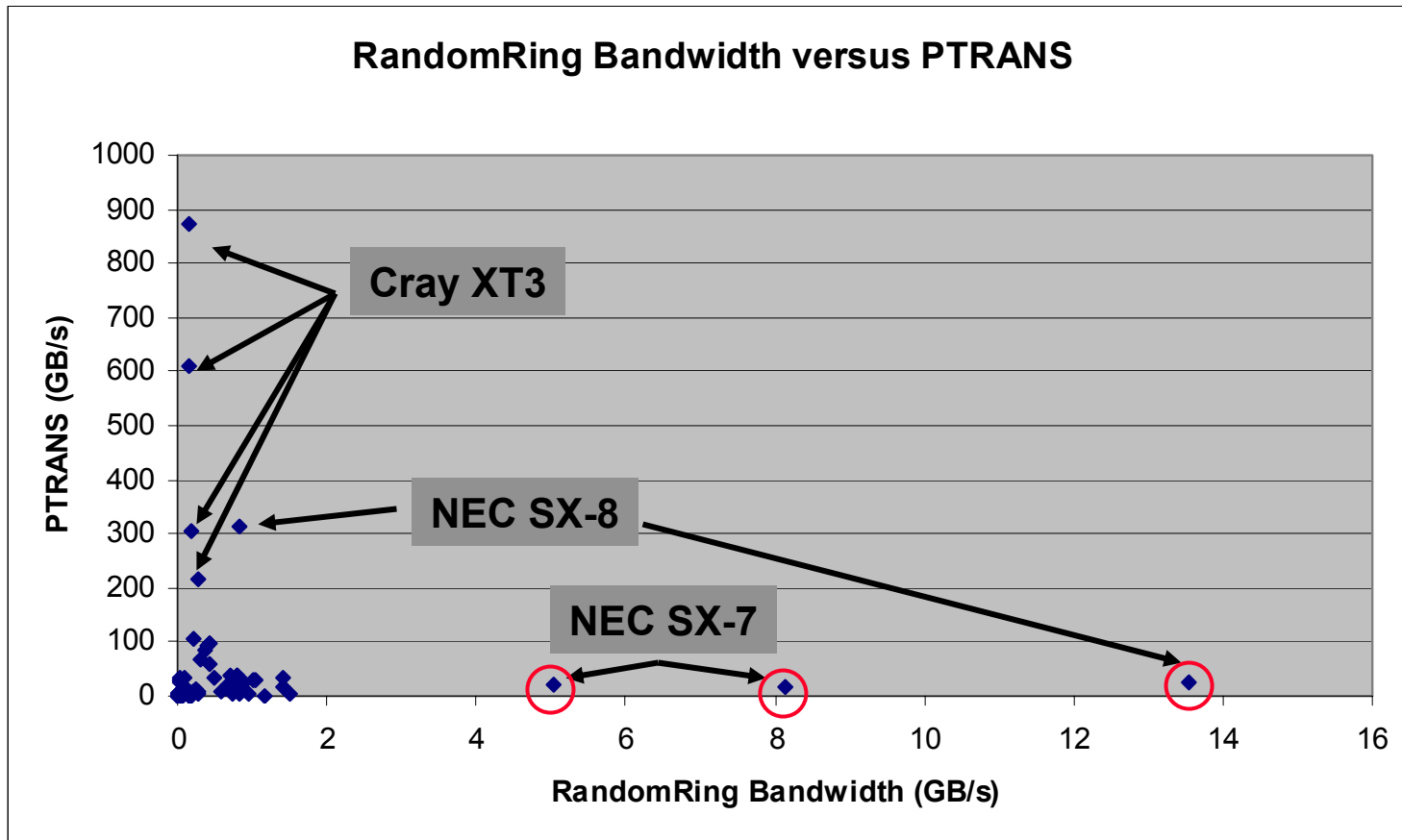
- How well does HPL correlate with FFT performance?
- Note the 1,000x difference in scales! (Tera/Giga)

# Global STREAM versus PTRANS



- How well does STREAM data correlate with PTRANS performance?

# RandomRing Bandwidth versus PTRANS

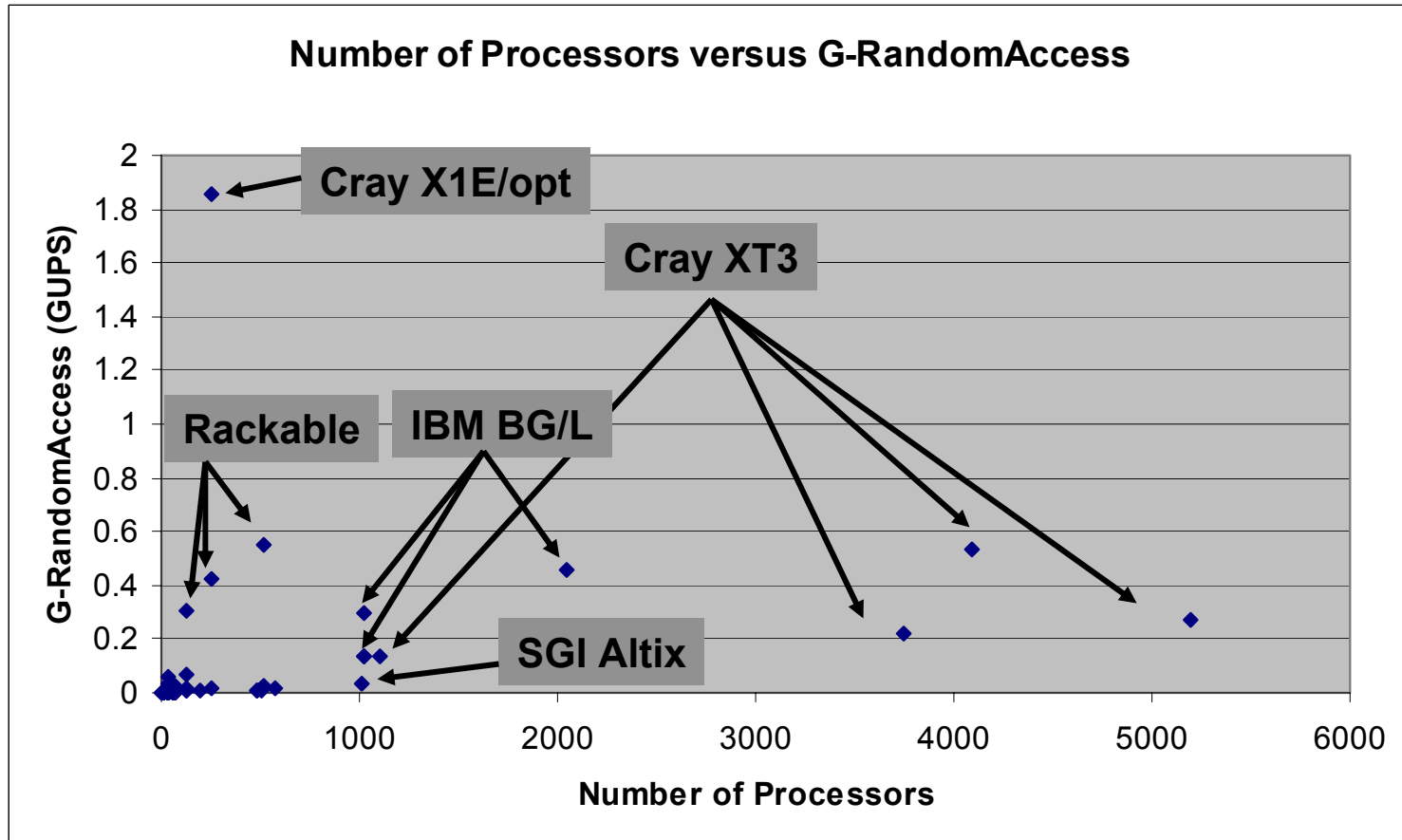


- How well does RandomRing Bandwidth data correlate with PTRANS performance
- Possible bad data?

# RandomAccess Correlations?

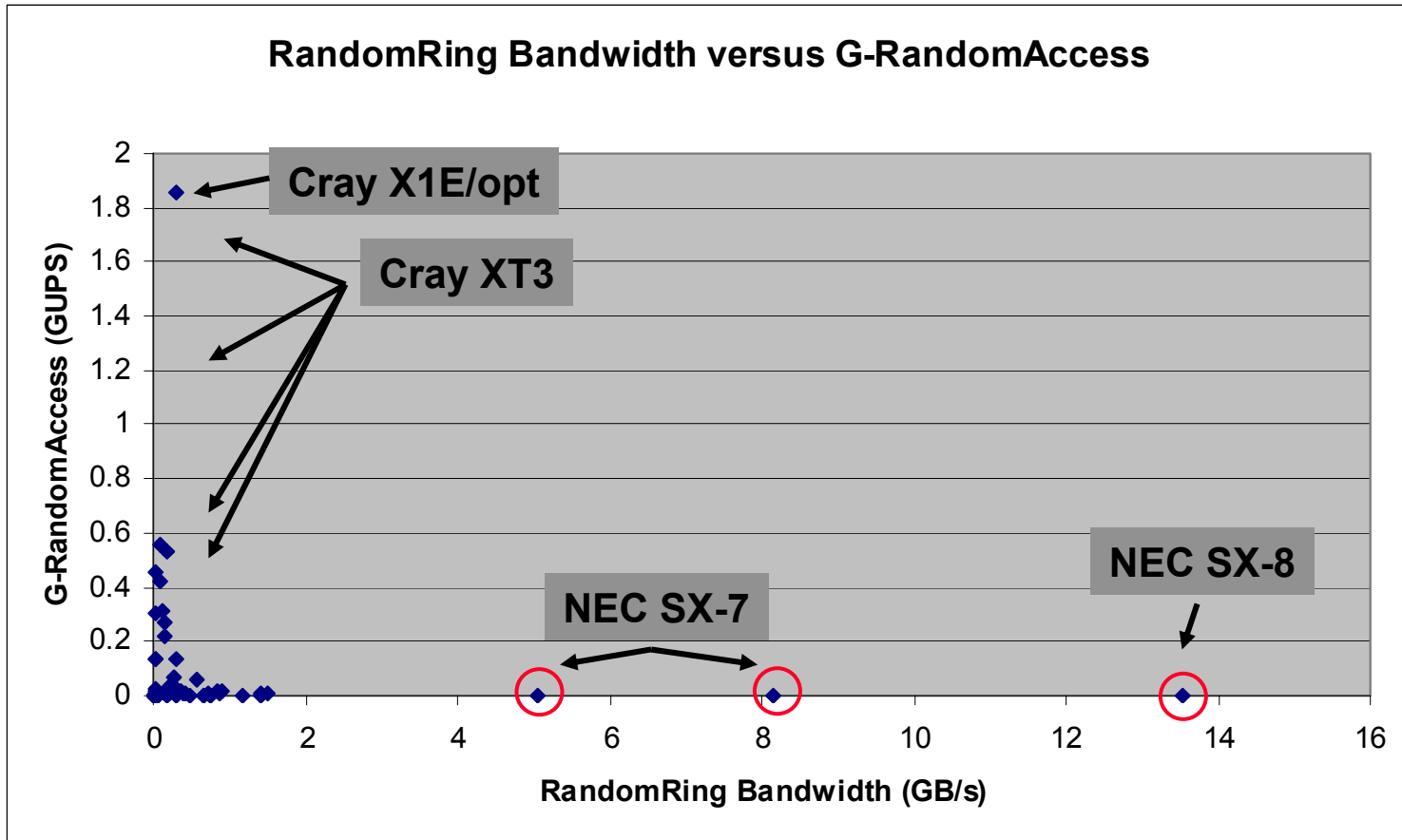
- **HPL versus G-RandomAccess**
- **Number of Processors versus G-RandomAccess**
- **RandomRing Bandwidth versus G-RandomAccess**
- **RandomRing Latency versus G-RandomAccess**
- **Single Processor RandomAccess versus G-RandomAccess**
  - per System (Single Processor)
  - per Processor (Single Processor)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

# Number of Processors versus G-RandomAccess



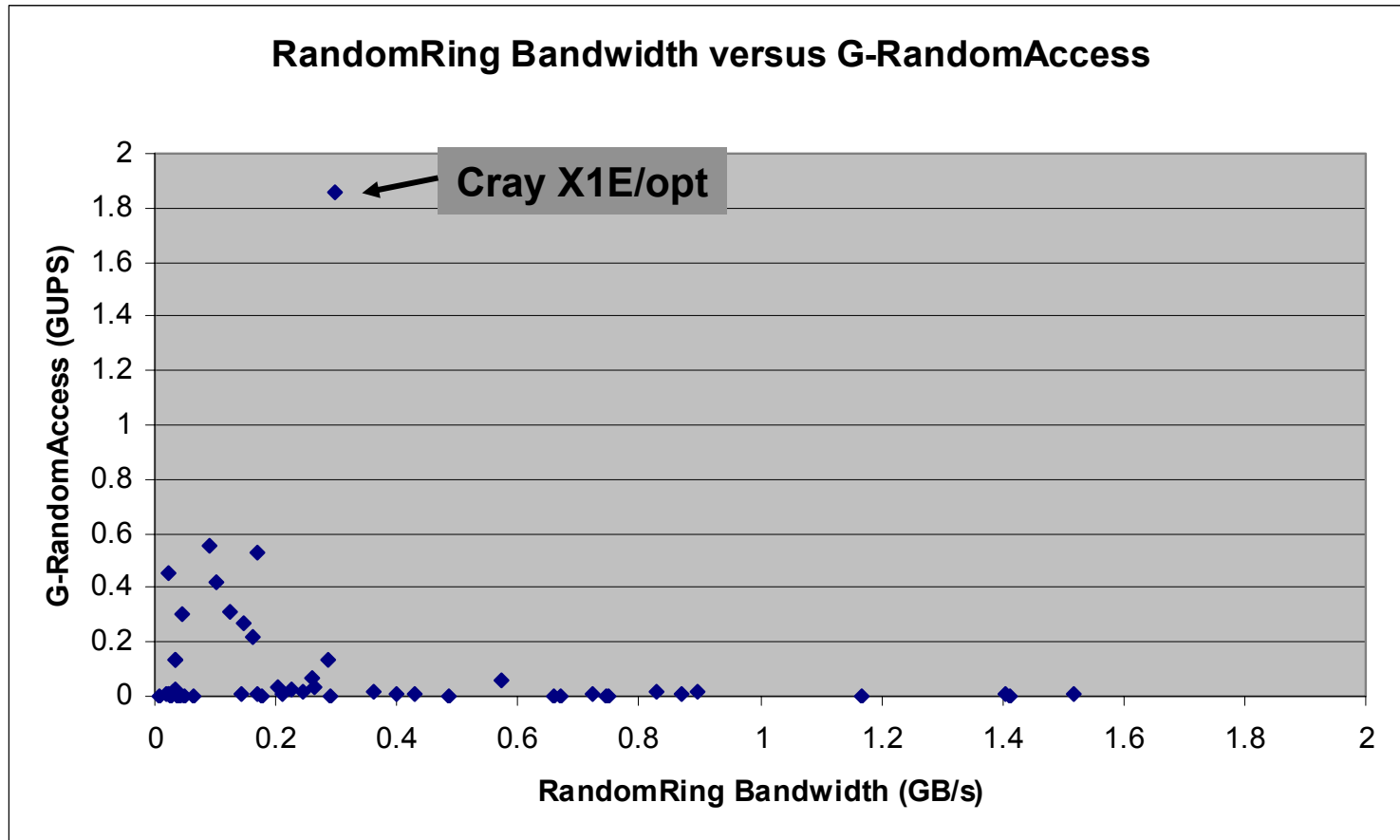
- Does G-RandomAccess scale with the number of processors?

# RandomRing Bandwidth versus G-RandomAccess



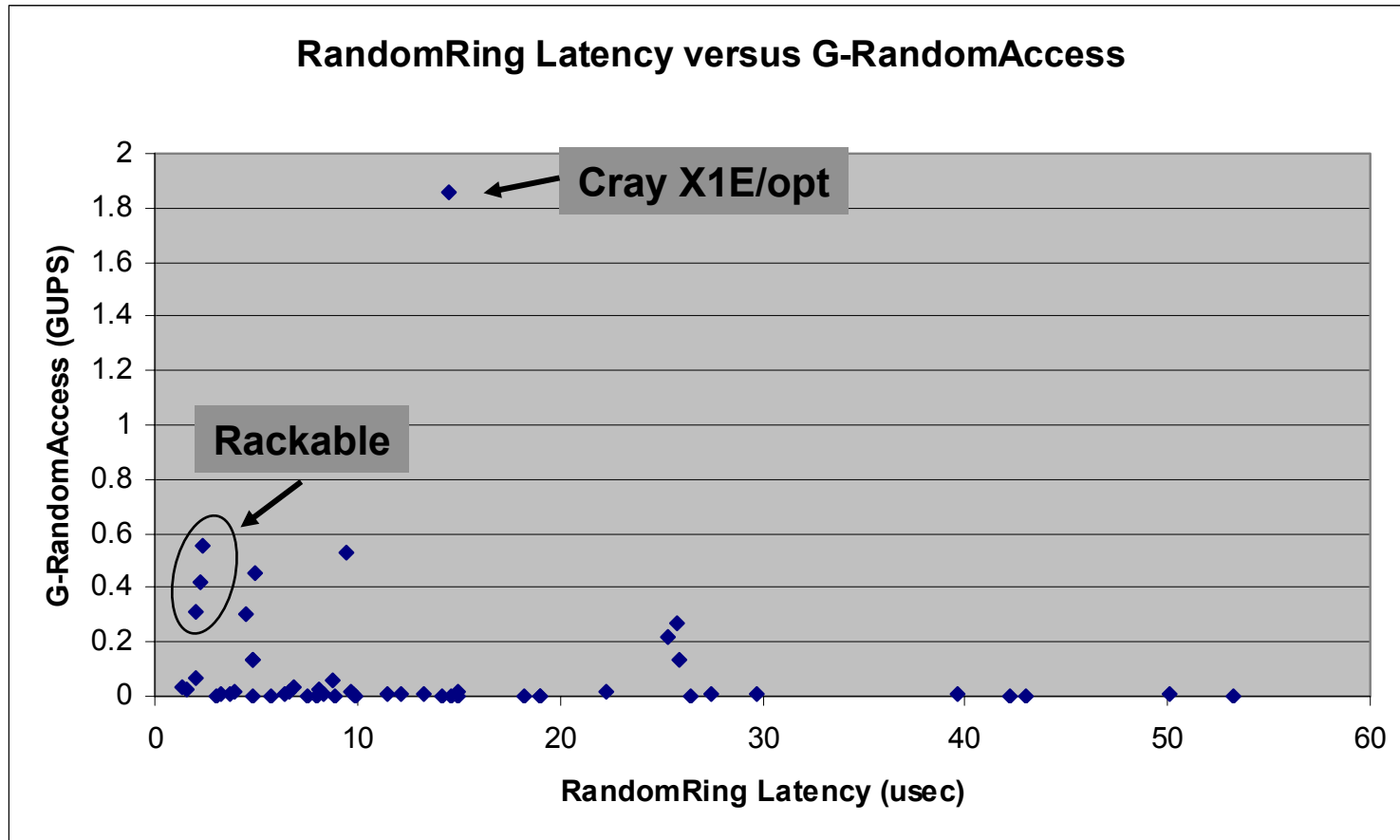
- Does G-RandomAccess scale with the RandomRing Bandwidth?
- Possible bad data?

# RandomRing Bandwidth versus G-RandomAccess



- Does G-RandomAccess scale with RandomRing Bandwidth?
- Ignoring possible bad data...

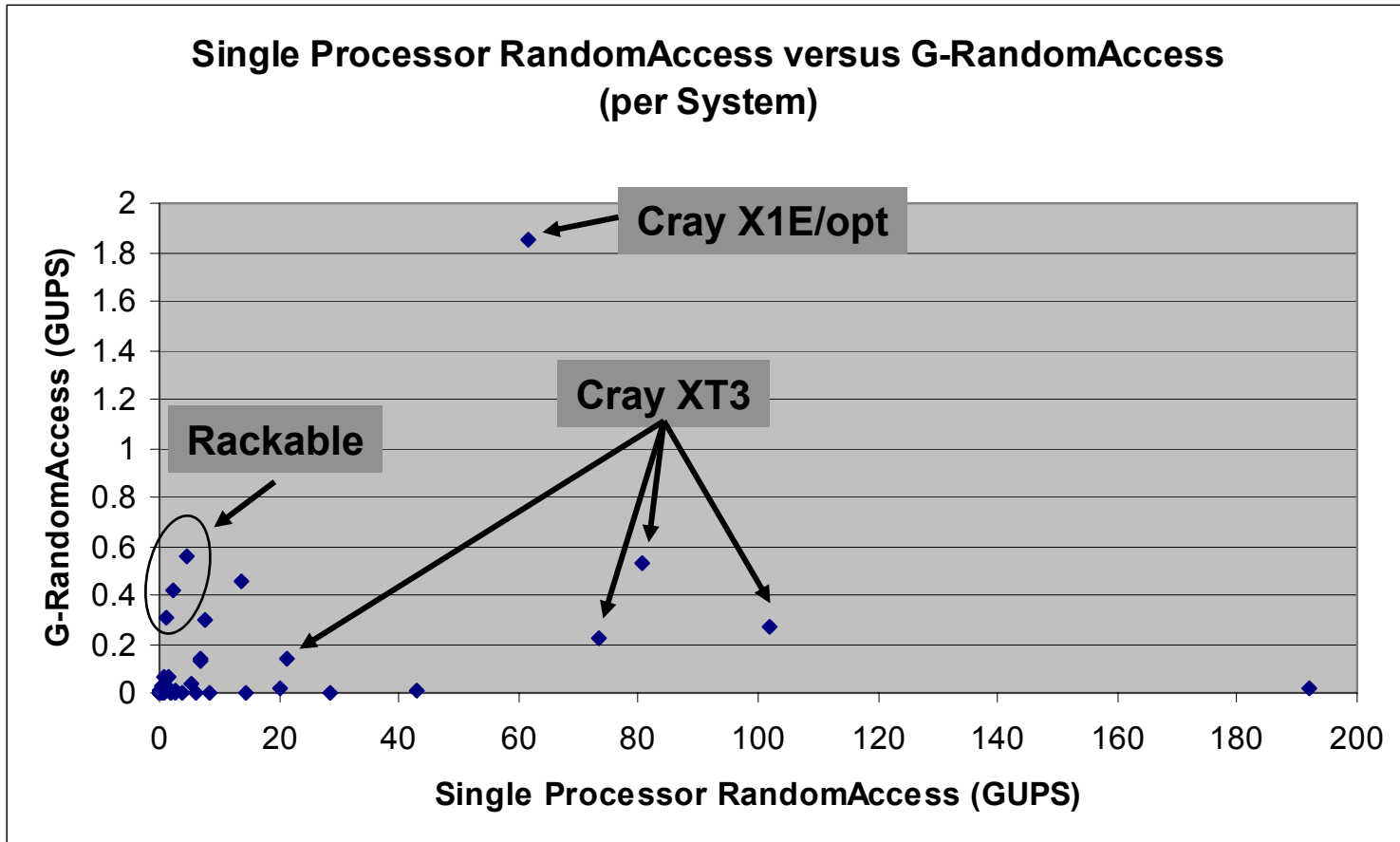
# RandomRing Latency versus G-RandomAccess



- Does G-RandomAccess scale with RandomRing Latency ?

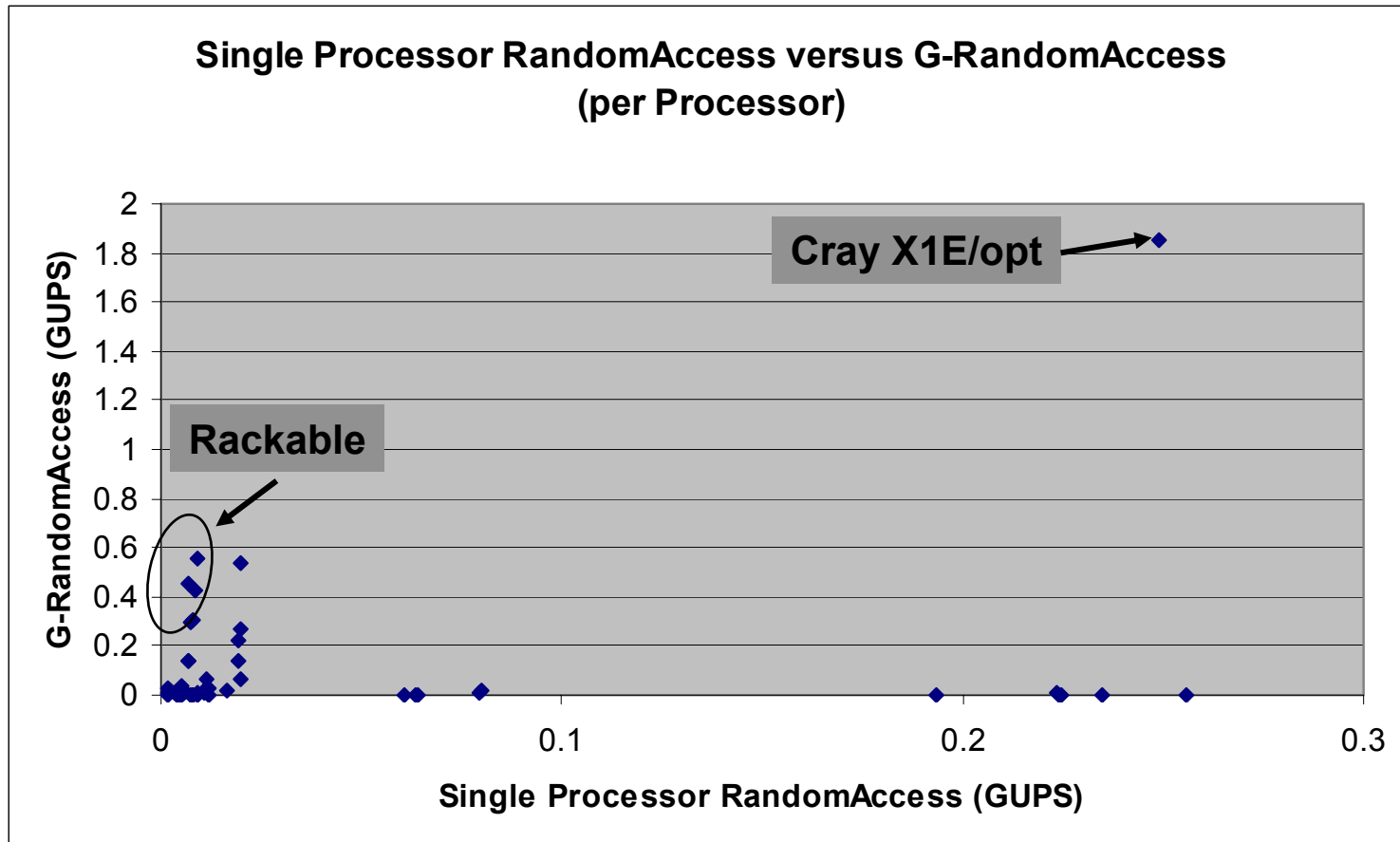


# Single Processor RandomAccess versus G-RandomAccess (per System)



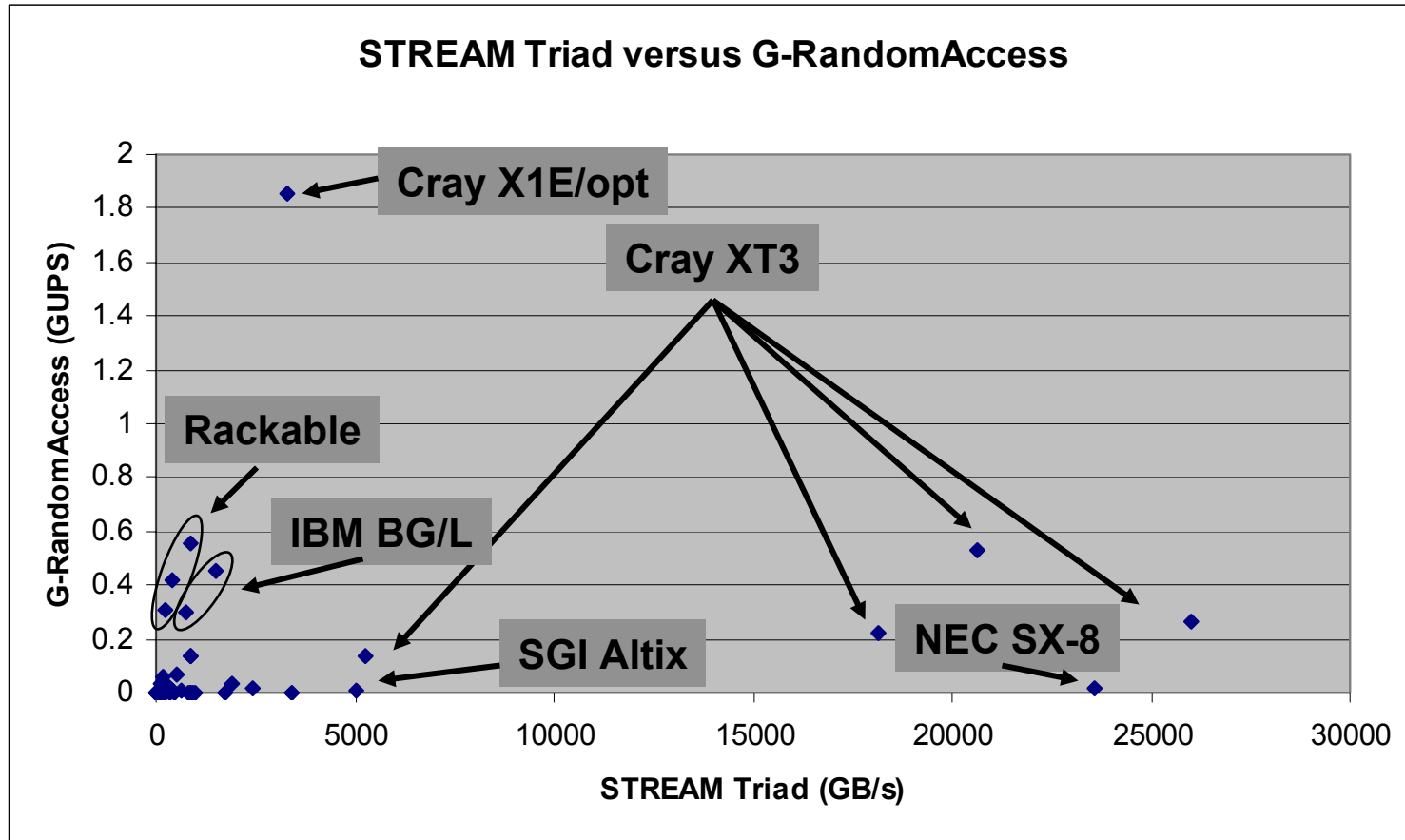
- Does G-RandomAccess scale with single processor RandomAccess performance (per system)?

# Single Processor RandomAccess versus G-RandomAccess (per Proc)



- 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

**NO!**

# 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

**NO!**

**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)

```

/*****
/* Note: UPC version of RandomAccess Benchmark */
/*
/* Date: Mon Jun 13 10:57:54 2005 */
/*
/*
/* Written By */
/* High Performance Computing Lab */
/* The George Washington University */
*****/
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;

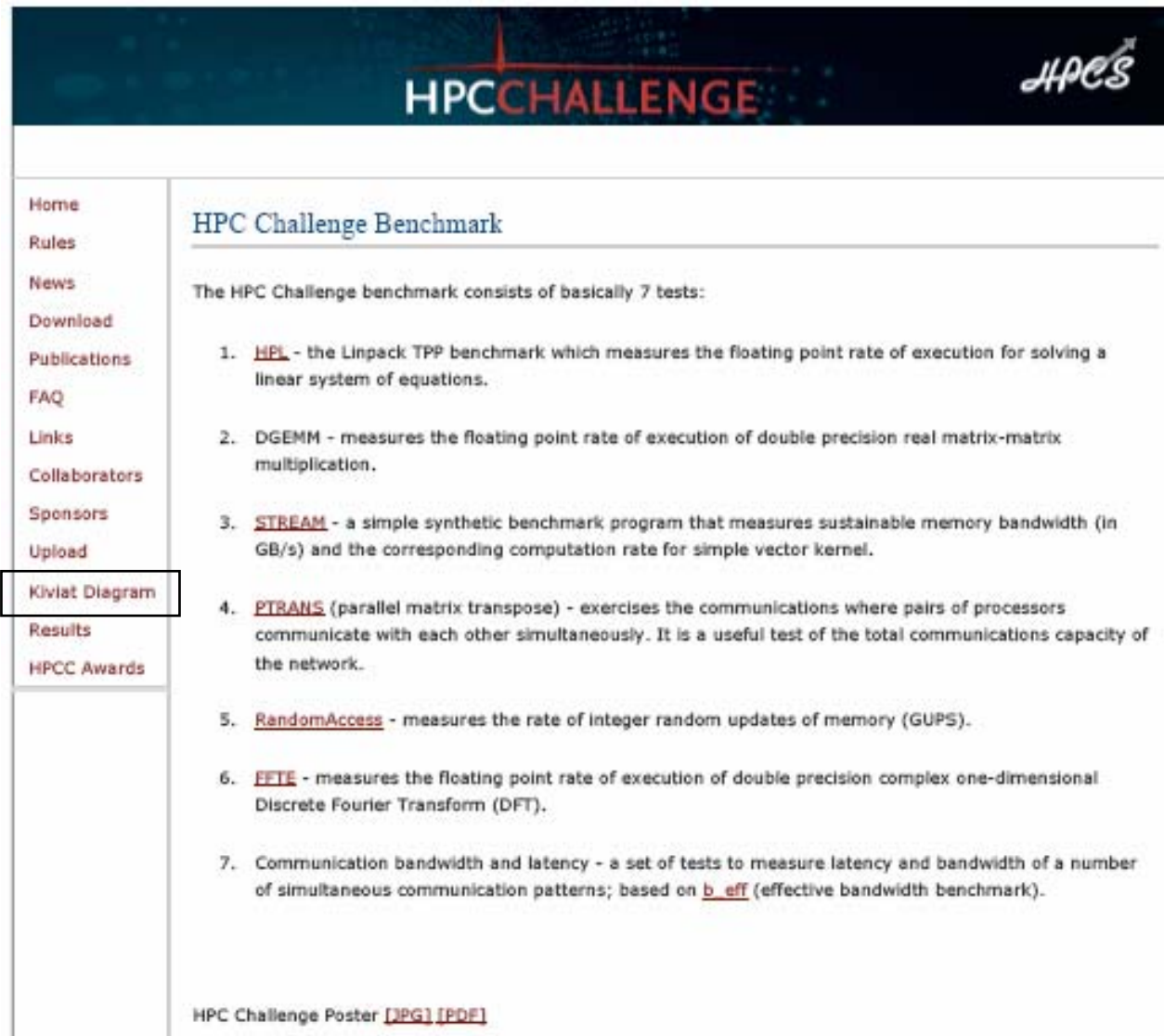
    upc_barrier;

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

```

**Initialize  
Table** →

**Update  
Table** →



Home  
Rules  
News  
Download  
Publications  
FAQ  
Links  
Collaborators  
Sponsors  
Upload  
Kivlat Diagram  
Results  
HPC Awards

## HPC Challenge Benchmark

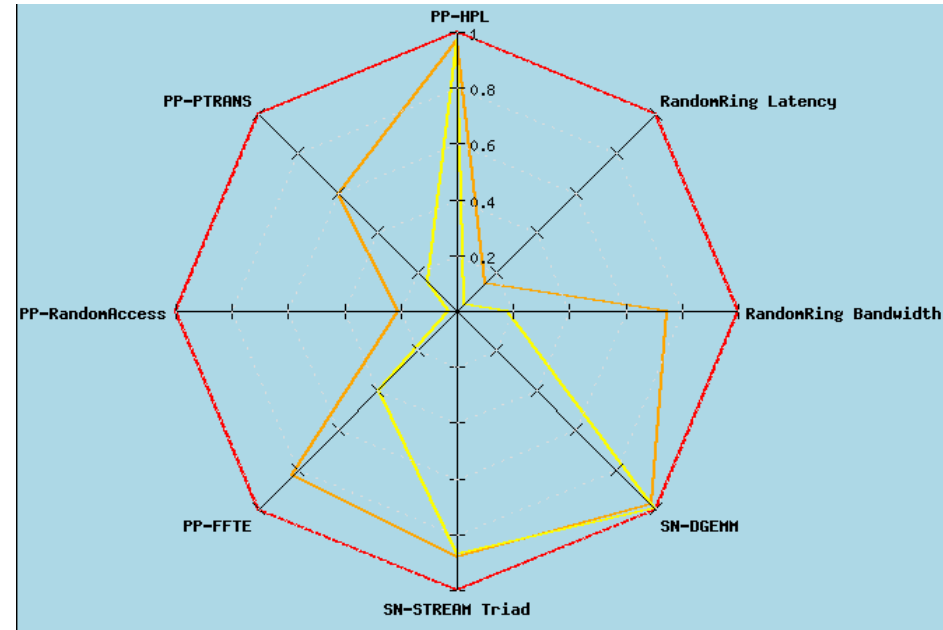
The HPC Challenge benchmark consists of basically 7 tests:

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. [FFTC](#) - 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 Poster [\[JPG\]](#) [\[PDF\]](#)

# 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*”





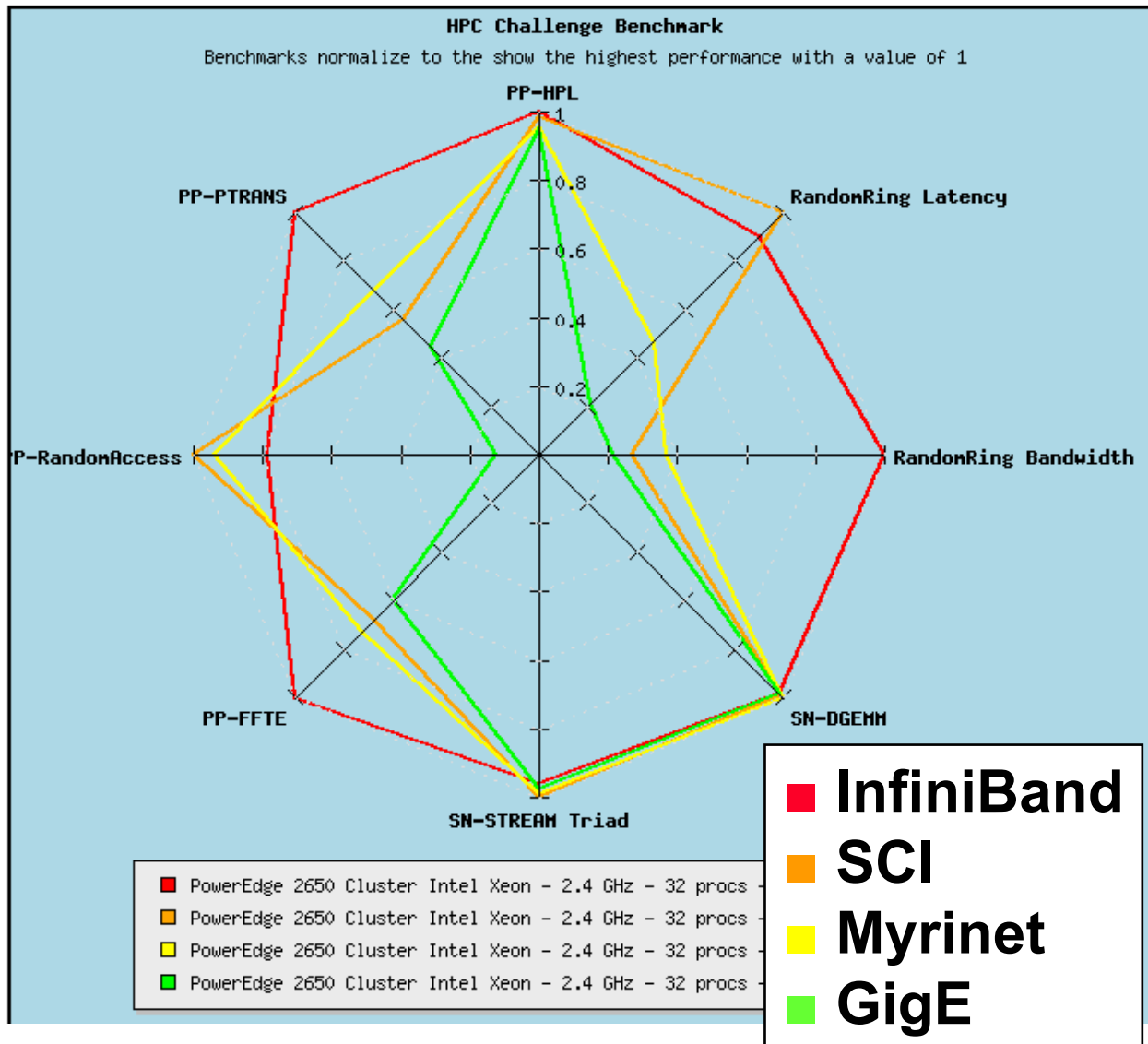
# Kiviat Chart Disclaimer

- **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>

# HPC Challenge Analysis

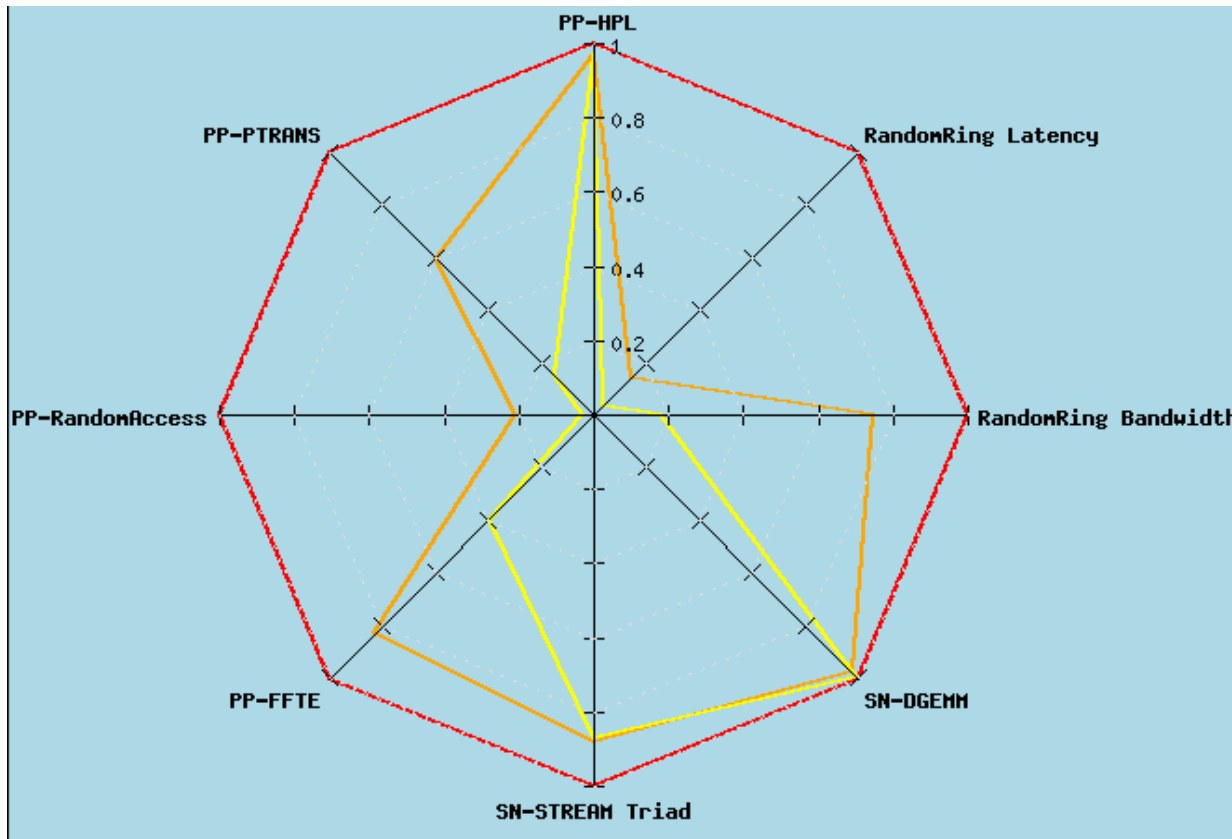
## Kiviat Diagram — Scali Cluster Comparison



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 FFTE  
Interconnect technology does matter Bandwidth sensitive

# HPC Challenge Analysis

## Kiviat Diagram — Opteron System Comparison



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 FFTE**  
Interconnect technology does matter! Bandwidth sensitive

- Cray XD1 AMD Opteron - 64 procs - 2.2 GHz  
1 thread/MPI process (64) - RapidArray Interconnect System - 11-22-2004
- Dalco Opteron/QsNet Linux Cluster AMD Opteron - 64 procs - 2.2 GHz  
1 thread/MPI process (64) - QsNetII - 11-04-2004
- Sun Fire V20z Cluster AMD Opteron - 64 procs - 2.2 GHz  
1 thread/MPI process (64) - Gigabit Ethernet, Cisco 6509 switch - 03-06-2005

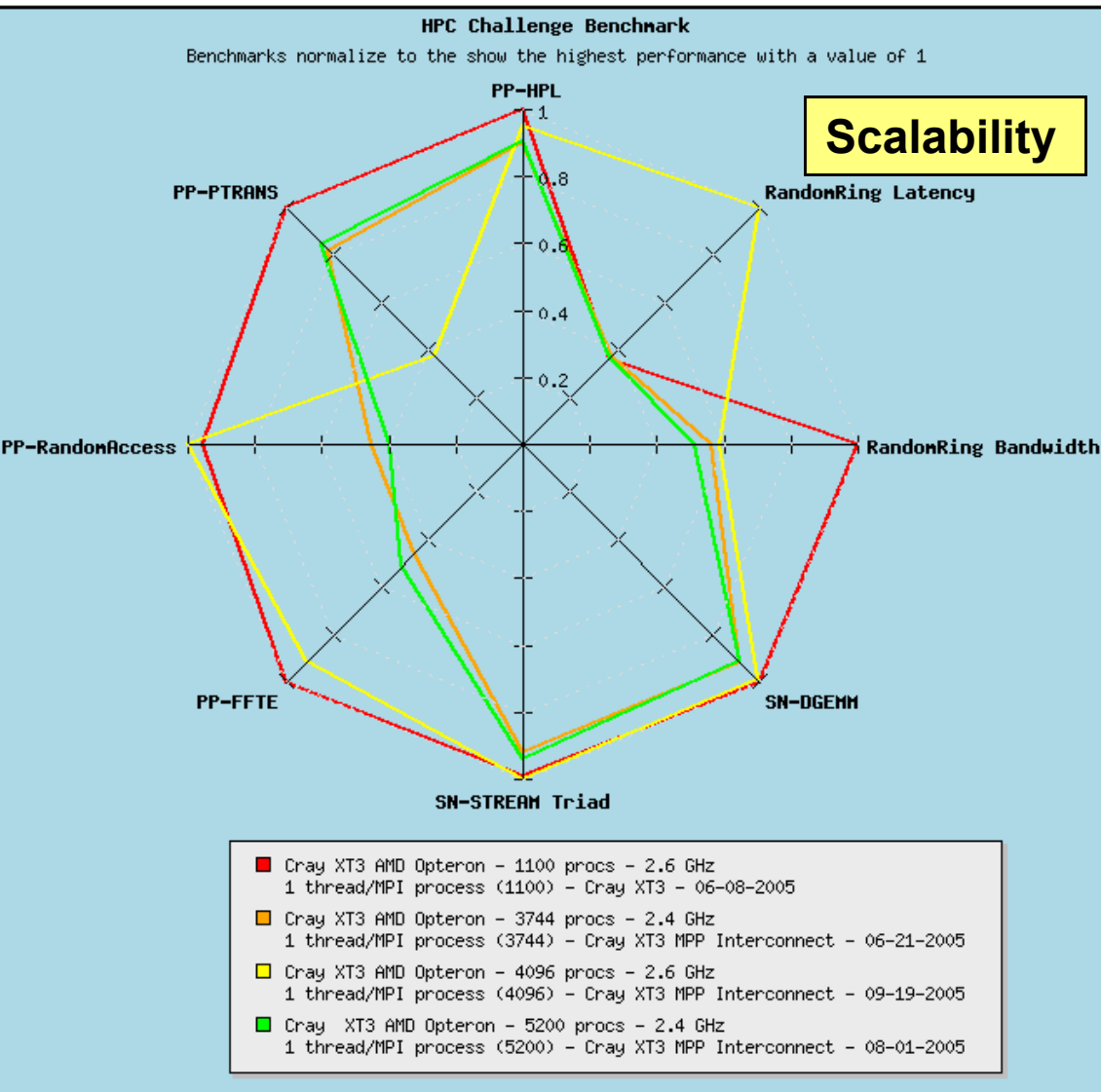
# HPC Challenge Analysis

## Kiviat Diagram — Cray XT3 Comparison

Per System — Absolute Scaling for Operations Benchmarks

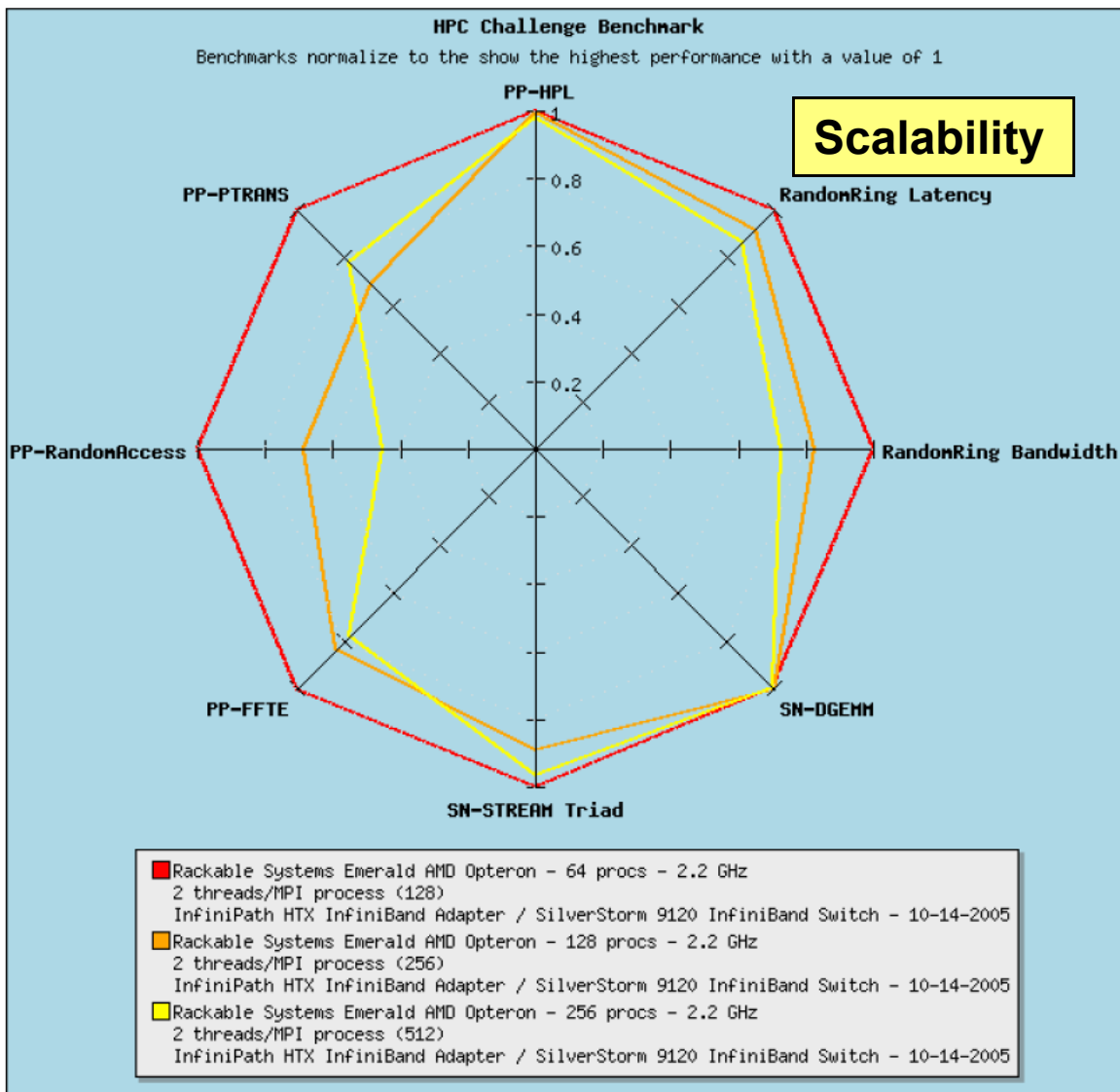
Per Processor Performance  
“Weak Scaling”

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



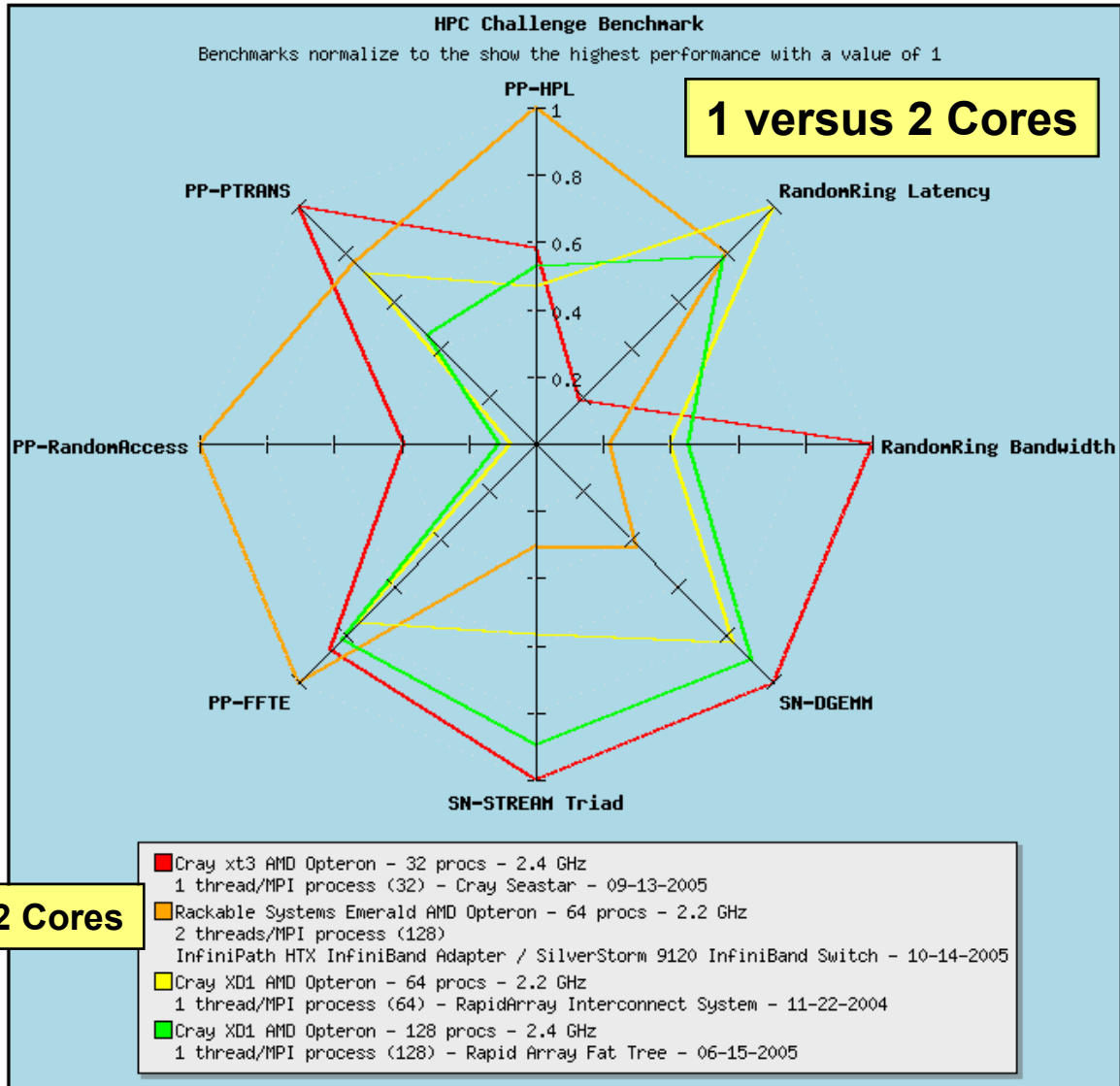
### Per System — Absolute Scaling for Operations Benchmarks

#### 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!

# HPC Challenge Analysis

## Kiviat Diagram — Opteron System Comparison



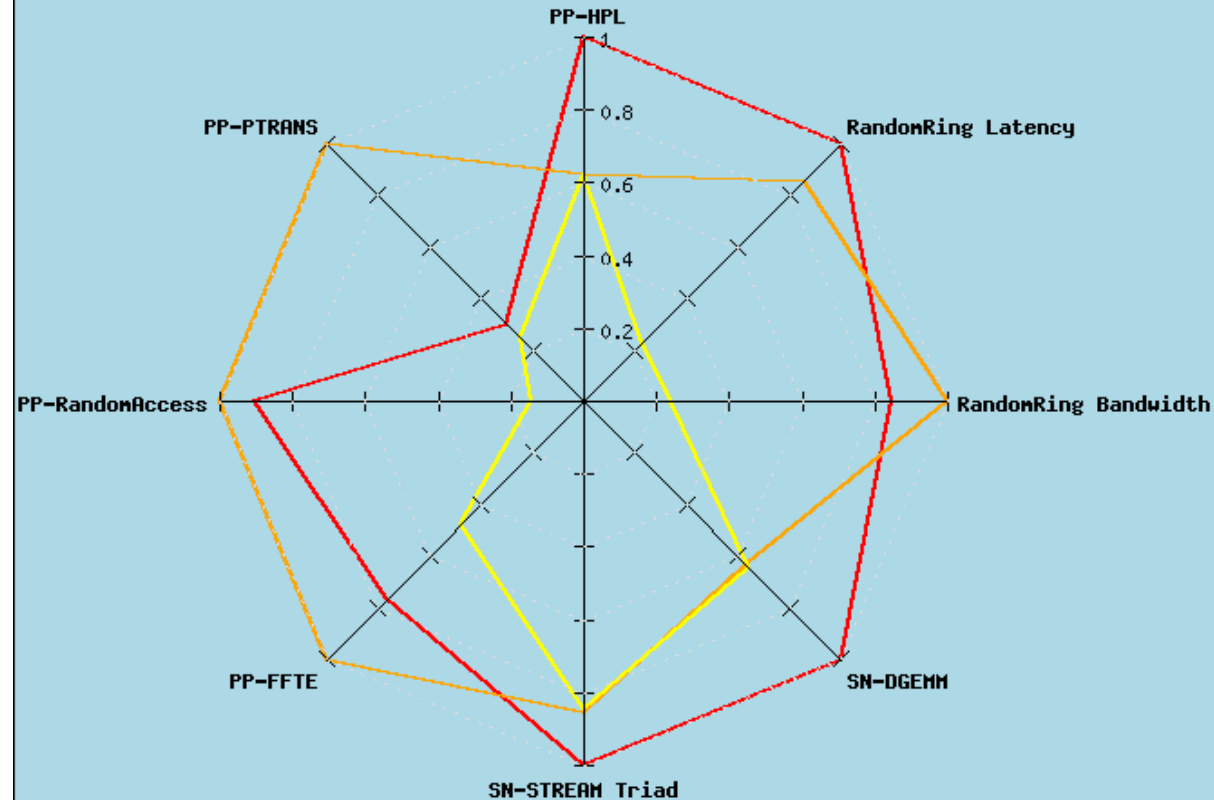
1. **RandomRing Bandwidth**  
System with two cores has significantly lower bandwidth  
⇒ cores vs interconnect?
2. **RandomRing Latency**  
System with two cores has slightly lower latency than one technology  
⇒ cores vs connect?
3. **STREAM and DGEMM**  
Significantly reduced performance for 2 cores
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

# HPC Challenge Analysis

## Kiviat Diagram — Cluster Comparison

HPC Challenge Benchmark

Benchmarks normalize to show the highest performance with a value of 1

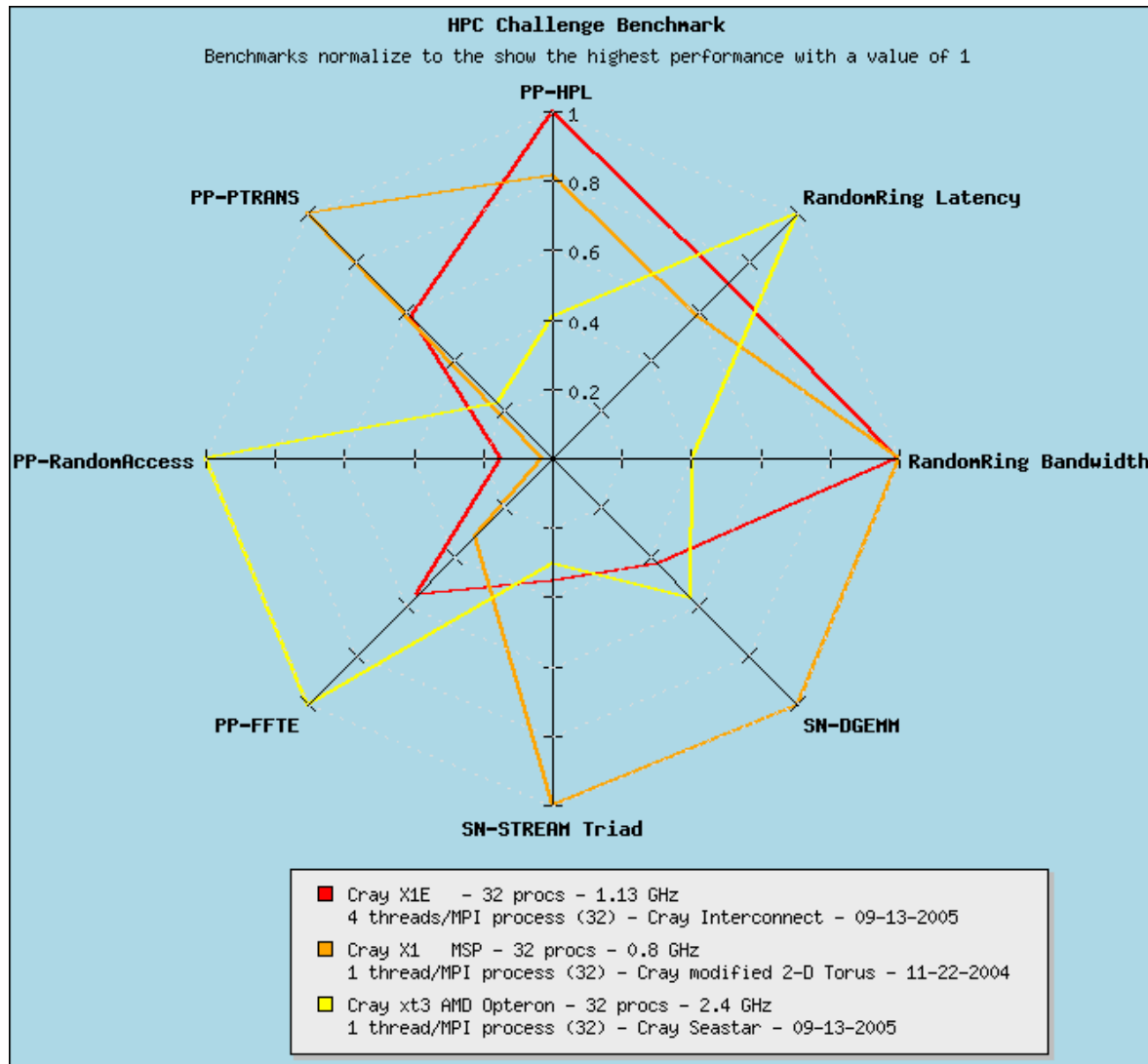


- Dell PowerEdge 1850 cluster Intel Xeon EM64T - 64 procs - 3.4 GHz  
1 thread/MPI process (64) - InfiniBand - 10-12-2004
- Dalco Opteron/QsNet Linux Cluster AMD Opteron - 64 procs - 2.2 GHz  
1 thread/MPI process (64) - QsNetII - 11-04-2004
- Sun Fire V20z Cluster AMD Opteron - 64 procs - 2.2 GHz  
1 thread/MPI process (64) - Gigabit Ethernet, Cisco 6509 switch - 03-06-2005

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 FFTE  
Interconnect technology does matter Bandwidth sensitive

# HPC Challenge Analysis

## Kiviat Diagram — Cray Architecture Comparison



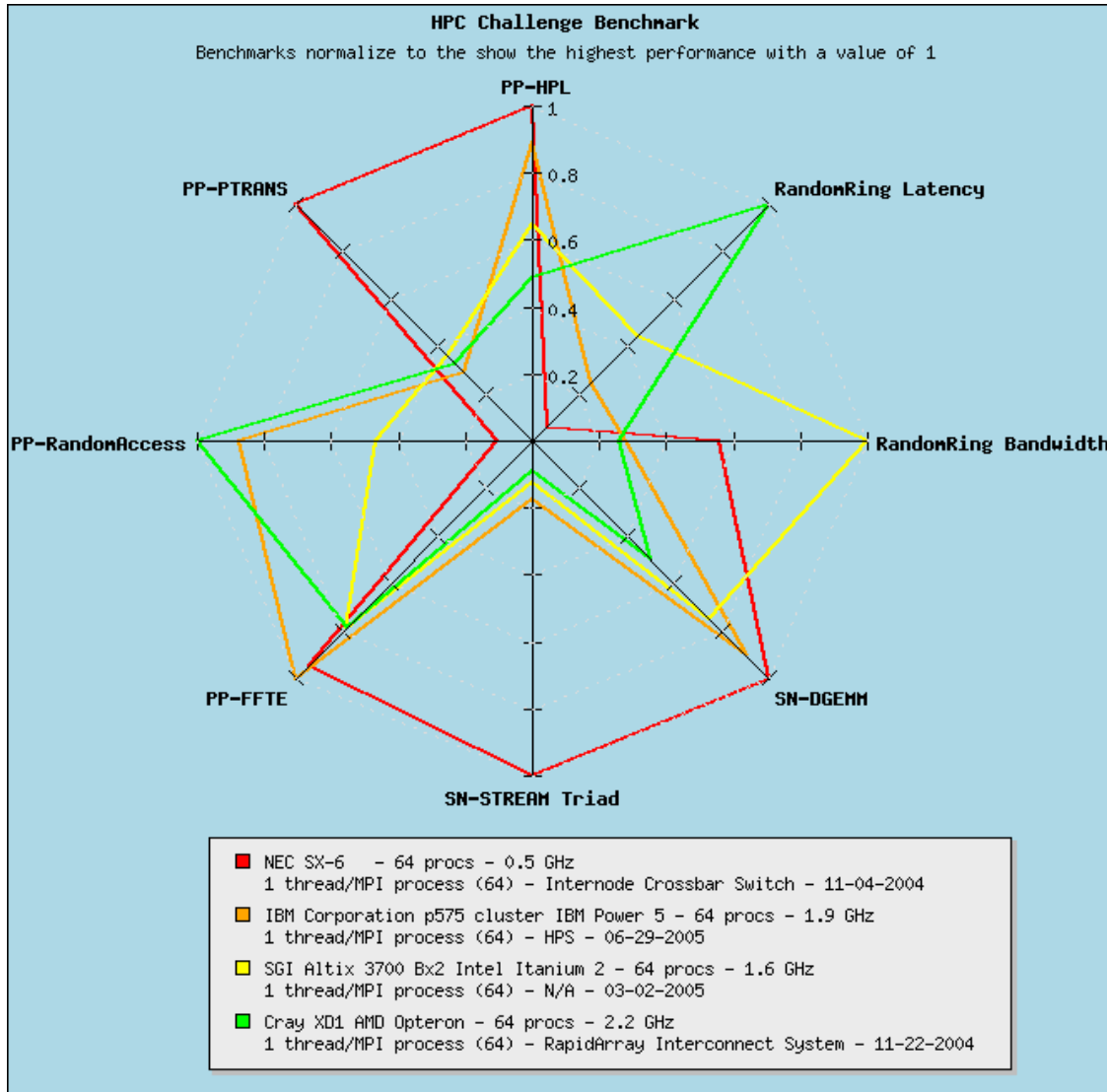
### Comparing Dissimilar Systems Can be Difficult!

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. FFTE  
NEC SX-6 vector processor slightly better  
Otherwise similar performance

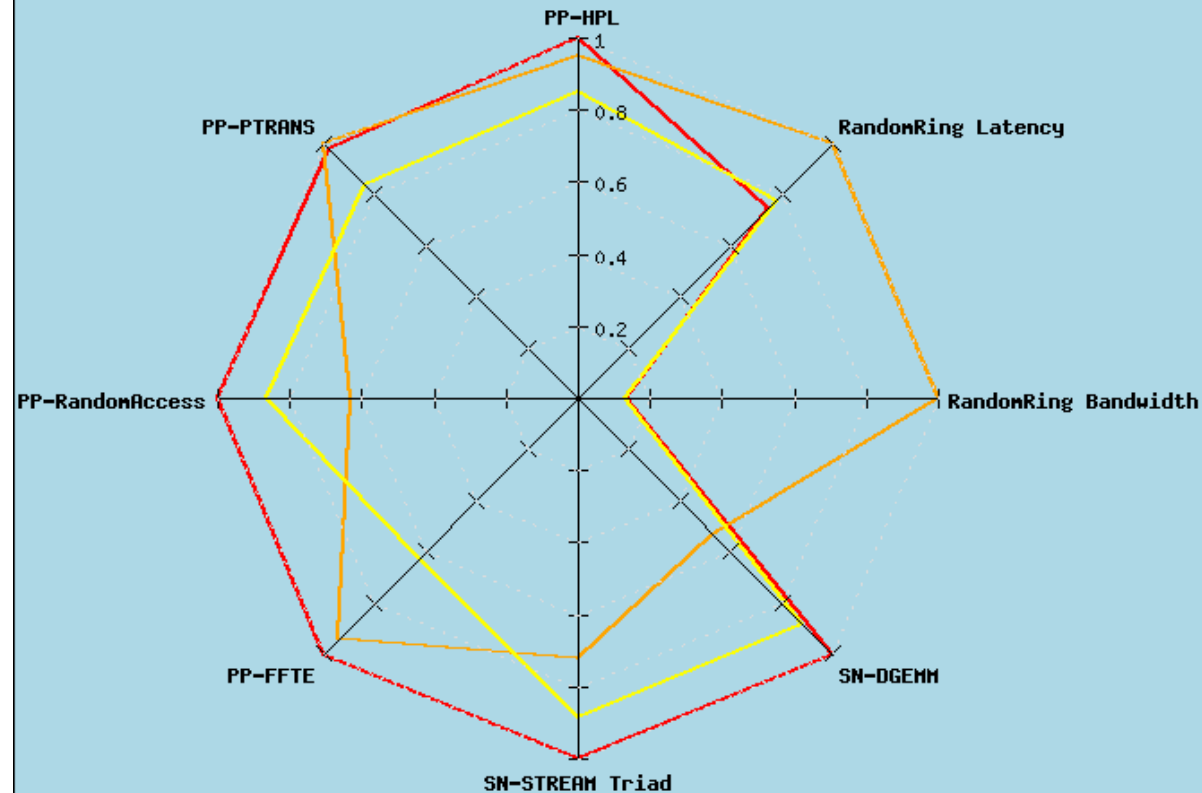
# HPC Challenge Analysis

## Kiviat Diagram — NEC SX-6/7 Comparison



### HPC Challenge Benchmark

Benchmarks normalize to show the highest performance with a value of 1



- NEC SX-6+ - 32 procs - 0.5625 GHz  
1 thread/MPI process (32) - IXS crossbar - 10-12-2004
- NEC SX-7 - 32 procs - 0.552 GHz  
1 thread/MPI process (32) - non - 11-05-2004
- NEC SX-6 - 32 procs - 0.5 GHz  
1 thread/MPI process (32) - Internode Crossbar Switch - 11-04-2004

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



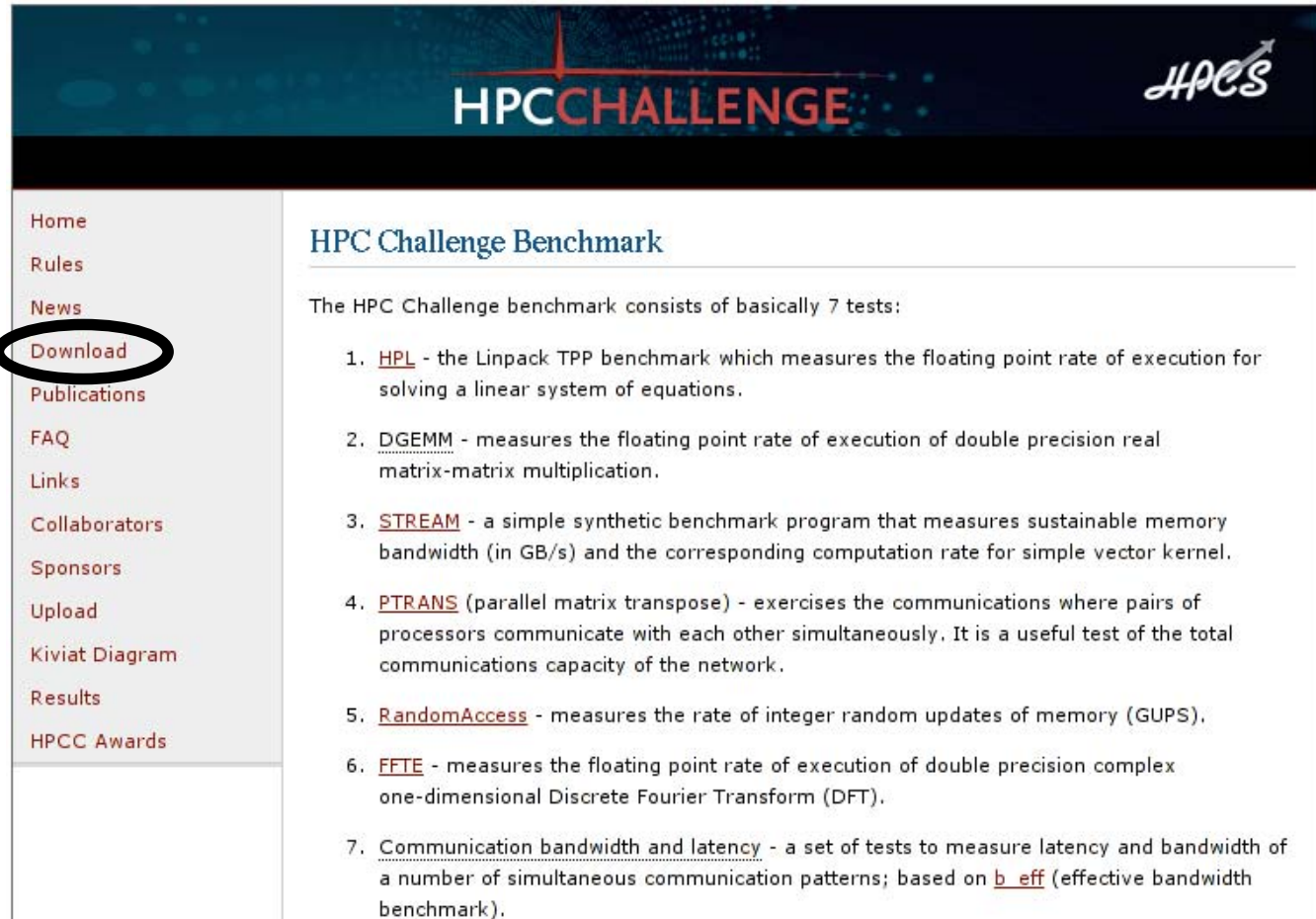
- Introduction
- Motivations
  - HPCS
  - Performance Characterization
- Component Kernels
- HPC Challenge Awards
- Unified Benchmark Framework
- Rules
  - Running HPC Challenge
  - Optimizations
  - Etiquette
- Performance Data
  - Available Benchmark Data
  - Kiviat Charts
- **Hands-on Demonstrations/Exercises**
  - 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**

- Always use the latest source code:  
<http://icl.cs.utk.edu/hpcc/>



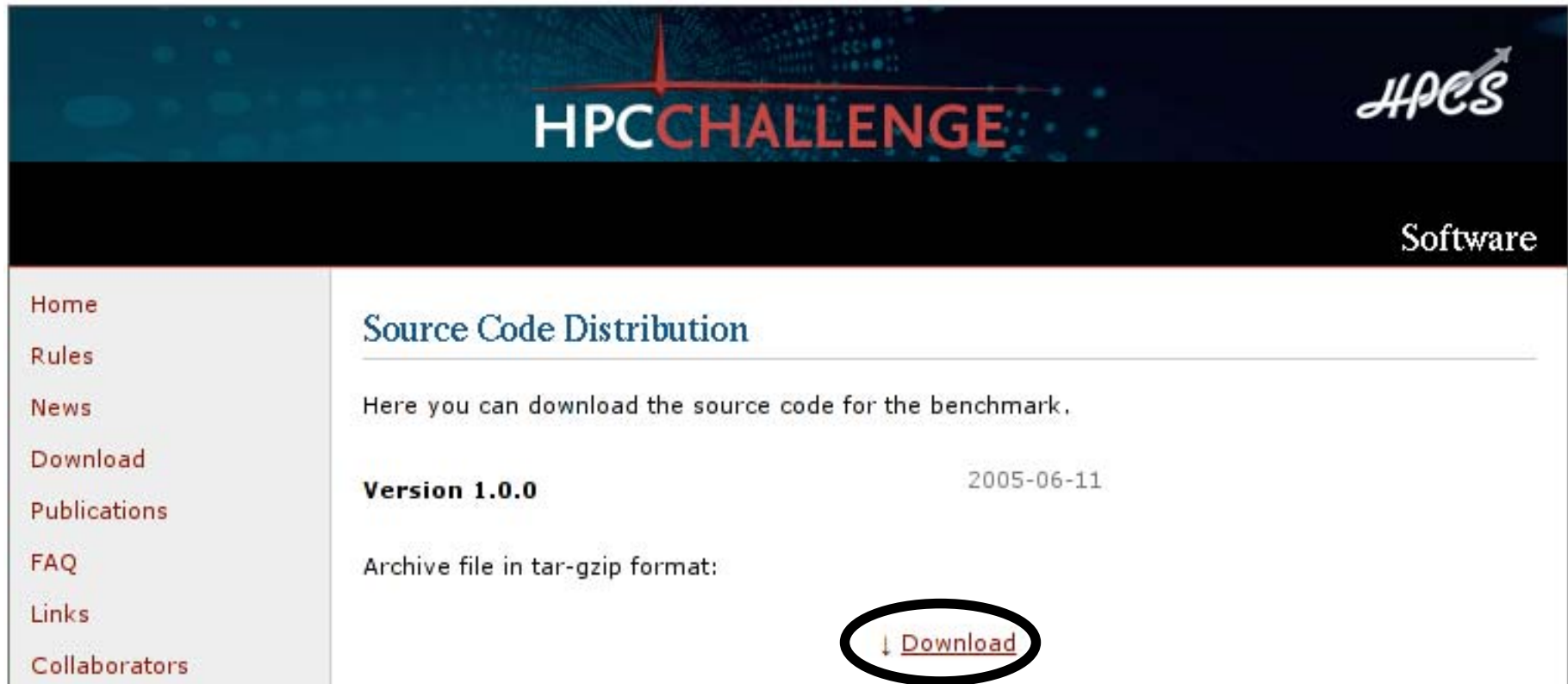
The screenshot shows the HPC Challenge Benchmark website. The header features the 'HPCCHALLENGE' logo and the 'HPCs' logo. A navigation menu on the left includes links for Home, Rules, News, Download (circled in red), Publications, FAQ, Links, Collaborators, Sponsors, Upload, Kiviat Diagram, Results, and HPC Awards. The main content area is titled 'HPC Challenge Benchmark' and lists seven tests: 1. HPL, 2. DGEMM, 3. STREAM, 4. PTRANS, 5. RandomAccess, 6. FFTE, and 7. Communication bandwidth and latency.

## 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. [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).

- <http://icl.cs.utk.edu/hpcc/software/index.html>



The screenshot shows the HPCC Challenge website's 'Software' section. The header features the 'HPCCHALLENGE' logo and a stylized 'HPCC' logo with an arrow. The main content area is titled 'Source Code Distribution' and contains the text: 'Here you can download the source code for the benchmark.' Below this, it lists 'Version 1.0.0' with a date of '2005-06-11'. Underneath, it says 'Archive file in tar-gzip format:' followed by a 'Download' link, which is circled in black. A left-hand navigation menu includes links for Home, Rules, News, Download, Publications, FAQ, Links, and Collaborators.

- Latest version is always at the top

- **Sample Makefiles live in**  
**hpl/setup**
- **BLAS**
  - **LAdir** – BLAS top directory for other **LA**-variables
  - **LAinc** – where BLAS headers live (if needed)
  - **ALib** – 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**



- **Vendor**
  - Cray (MPT)
  - IBM (POE)
  - SGI (MPT)
  - Dolphin, Infiniband (Mellanox, Voltaire, ...), Myricom (GM, MX), Quadrics, PathScale, Scali, ...
- **Open Source**
  - MPICH1, MPICH2 (<http://www-unix.mcs.anl.gov/mpi/mpich/>)
  - Lam MPI (<http://www.lam-mpi.org/>)
  - OpenMPI (<http://www.open-mpi.org/>)
  - LA-MPI (<http://public.lanl.gov/lampi/>)
- **MPI implementation components**
  - Compiler (adds MPI header directories)
  - Linker (need to link in Fortran I/O)
  - Exe (poe, mprun, mpirun, aprun, mpiexec, ...)

- **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

- **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

- **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**

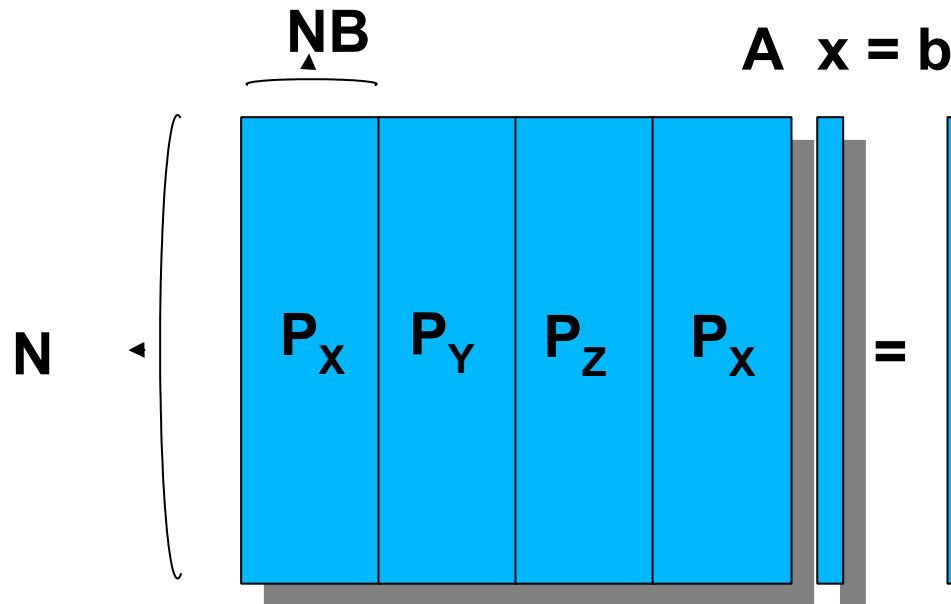
# Parallelism Examples

- **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 !)

- **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...

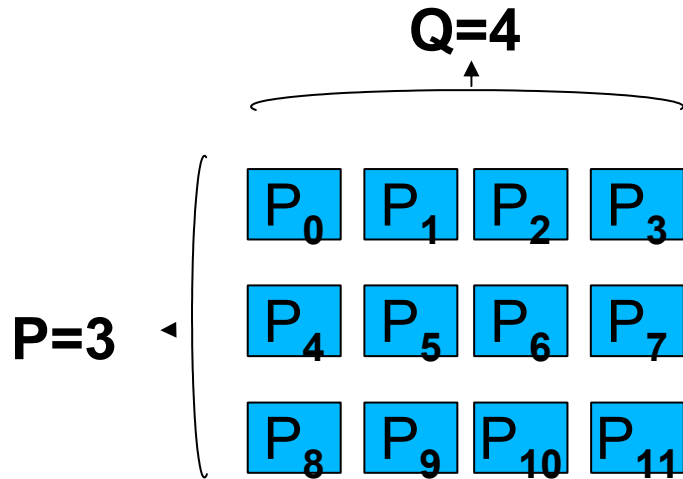
# Tuning HPL - Introduction

- 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

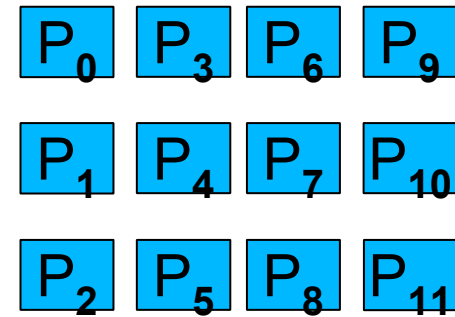


# Tuning HPL – More Definitions

- Process grid parameters: P, Q, and PMAP



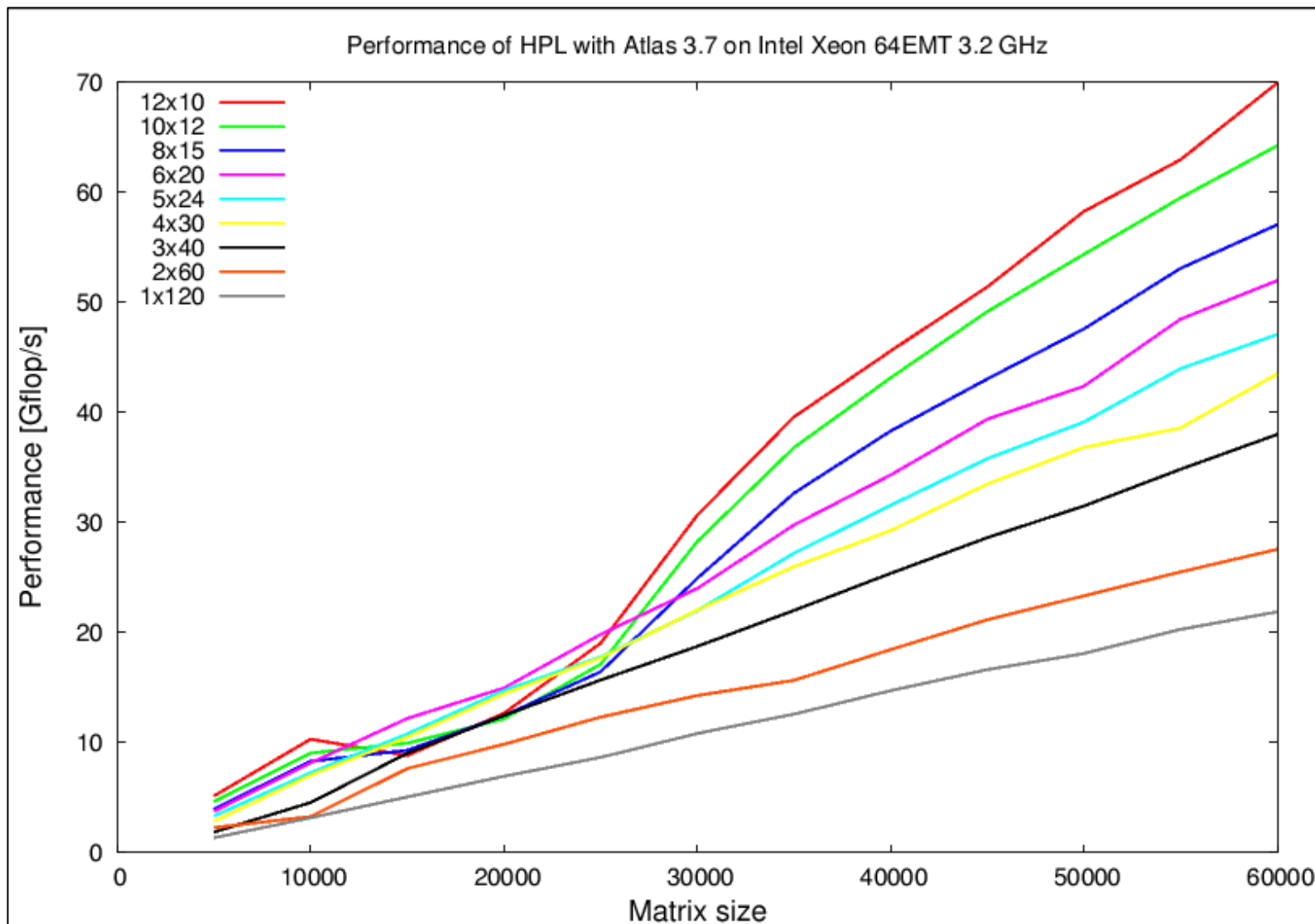
PMAP=C



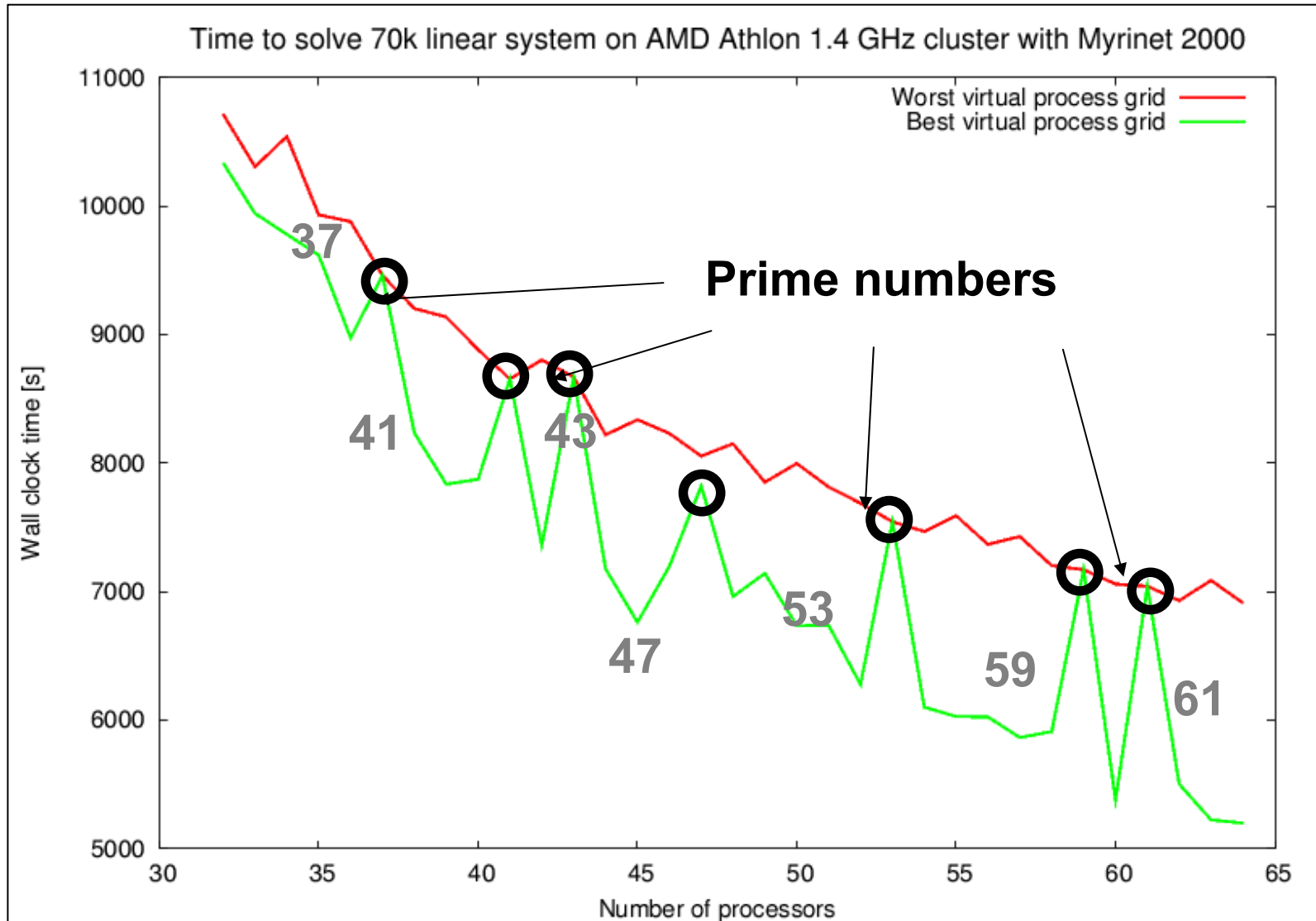
PMAP=R



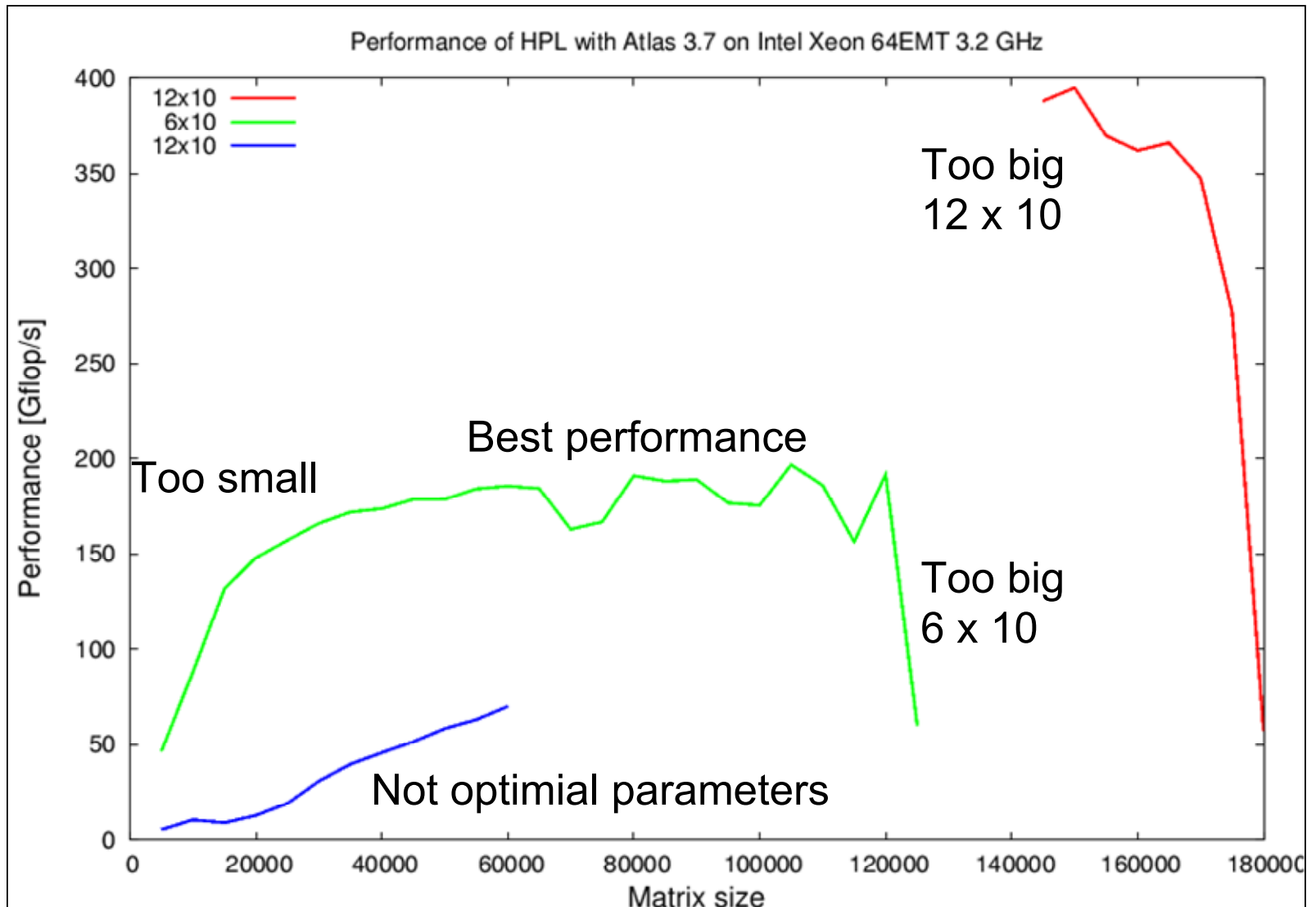
# Tuning HPL – Selecting Process Grid



# Tuning HPL – Number of Processors



# Tuning HPL – Matrix Size



- <http://www.netlib.org/benchmark/hpl/>
- Much more details from HPL's author:
- Antoine Petitet

- **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

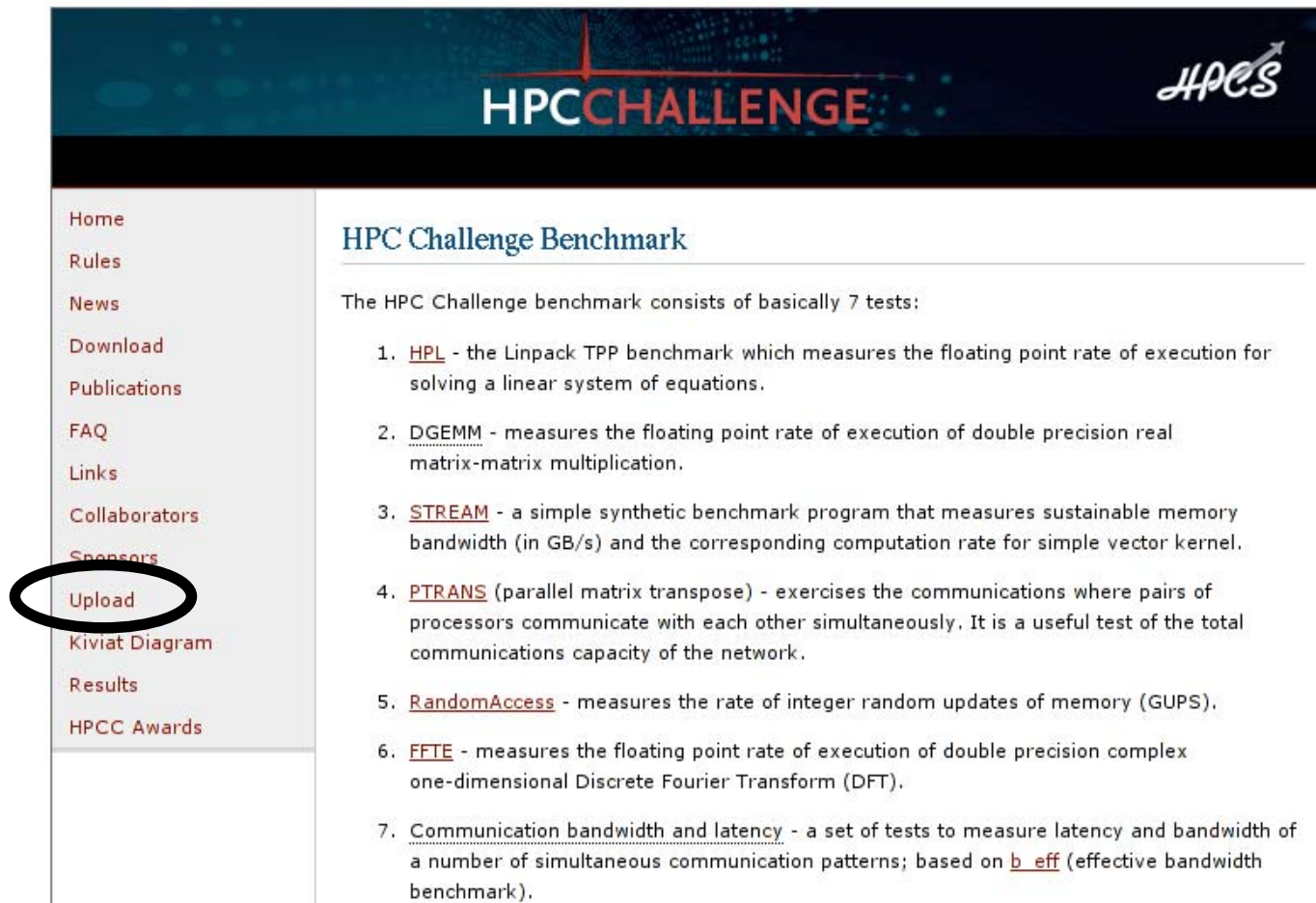
- **b\_eff (Effective bandwidth and latency) test can also be tuned**
- **Tuning must use only standard MPI calls**
- **Examples**
  - **Persistent communication**
  - **One-sided communication**



- 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 **hpcconf.txt** should be submitted along with system info




**HPCCHALLENGE**


Home  
Rules  
News  
Download  
Publications  
FAQ  
Links  
Collaborators  
Sponsors  
**Upload**  
Kiviat Diagram  
Results  
HPC Awards

## 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. [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).





[Upload](#)

[Home](#)  
[Rules](#)  
[News](#)  
[Download](#)  
[Publications](#)  
[FAQ](#)  
[Links](#)  
[Collaborators](#)  
[Sponsors](#)  
[Upload](#)  
[Kiviat Diagram](#)  
[Results](#)  
[HPCC Awards](#)

## 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:	<input type="text"/>	<a href="#">help?</a>
Last name:	<input type="text"/>	<a href="#">help?</a>
Email address:	<input type="text"/>	<a href="#">help?</a>
Machine Location:	<input type="text" value="Select here:"/> <a href="#">help?</a> (country/state)	
City:	<input type="text"/> ( optional )	<a href="#">help?</a>
Institution/Affiliation:	<input type="text"/> ( eg. University of Tennessee )	<a href="#">help?</a>
Institution/Affiliation URL:	<input type="text"/> ( eg. http://icl.cs.utk.edu/ )	<a href="#">help?</a>

# 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



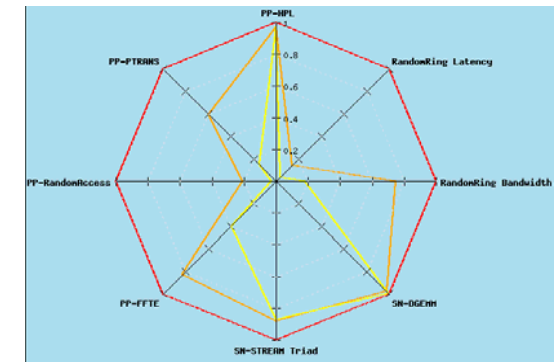
- Introduction
- Motivations
  - HPCS
  - Performance Characterization
- Component Kernels
- HPC Challenge Awards
- Unified Benchmark Framework
- Rules
  - Running HPC Challenge
  - Optimizations
  - Etiquette
- Performance Data
  - Available Benchmark Data
  - Kiviat Charts
- Hands-on Demonstrations/Exercises
  - 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

System Information					G-HPL	G-PTRANS	G-Random	G-FFTE	G-STREAM	EP	EP	Random	Random
System - Processor	Speed	Count	Tds	Proc	TFlop/s	GB/s	Access Gup/s	GFlop/s	Triad GB/s	STREAM Triad GB/s	DGEMM GFlop/s	Ring Bandwidth GB/s	Ring Latency usec
Cray XT3 AMD Opteron	2.4GHz	5200	1	5200	20.527	874.899	0.268583	644.73	26020.8	5.004	4.395	0.14682	25.8
Cray mfg8 X1E	1.13GHz	248	1	248	3.3889	66.01	1.85475	-1	3280.9	13.229	13.564	0.29886	14.58
Cray XT3 AMD Opteron	2.6GHz	4096	1	4096	16.9752	302.979	0.533072	905.57	20656.5	5.043	4.782	0.16896	9.44
NEC SX-7	0.552GHz	32	16	2	0.2174	16.34	0.000178	1.34	984.3	492.161	140.636	8.14753	4.85
NEC SX-8/6 SX-8	2GHz	6	1	6	0.0918	25.183	0.000769	3.19	370.6	61.773	15.944	13.5473	3.02
IBM pSeries 655 Power 4+	1.7GHz	256	4	64	1.0744	23.721	0.005502	10.46	411.7	6.433	17.979	0.72395	8.34
PathScale Inc. AMD Opteron	2.6GHz	32	1	32	0.1258	6.719	0.030367	10.35	134.3	4.197	4.775	0.26531	1.31



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