The HPC Challenge (HPCC) Benchmark Suite

Characterizing a system with several specialized kernels

Rolf Rabenseifner
rabenseifner@hlrs.de

High Performance Computing Center Stuttgart (HLRS)
University of Stuttgart
www.hlrs.de

SPEC Benchmarking Joint US/Europe Colloquium
June 22, 2007
Technical University of Dresden, Dresden, Germany

http://www.hlrs.de/people/rabenseifner/publ/publications.html#SPEC2007

Acknowledgements

- Update and extract from **SC'06 Tutorial S12**:

  Piotr Luszczek, David Bailey, Jack Dongarra, Jeremy Kepner, Robert Lucas, Rolf Rabenseifner, Daisuke Takahashi:

  "The HPC Challenge (HPCC) Benchmark Suite"

  SC06, Tampa, Florida, Sunday, November 12, 2006

- 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

Outline

- Overview
- The HPCC kernels
- Database & output formats
- HPCC award
- Augmenting TOP500
- Balance Analysis
- Conclusions

Introduction

- **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**

TOP500 and HPCC

- TOP500
  - Performance is represented by only a single metric
  - Data is available for an extended time period (1993-2006)
- 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 (2003-2007)
- Problem:
  There cannot be one "*winner*"
- We avoid "*composite*" benchmarks
  - Perform trend analysis
    - HPCC can be used to show complicated kernel/ architecture performance characterizations
  - Select some numbers for comparison
  - Use of kiviat charts
    - Best when showing the differences due to a single independent "variable"
  - Compute balance ratios
- Over time — also focus on growth trends

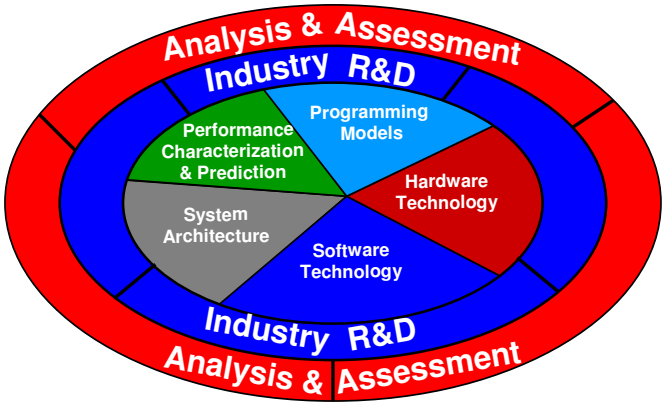
High Productivity Computing Systems (HPCS)

Goal:

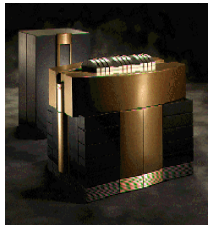
- **Provide a new generation of economically viable high productivity computing systems for the national security and industrial user community (2010)**

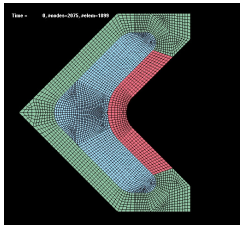
Impact:

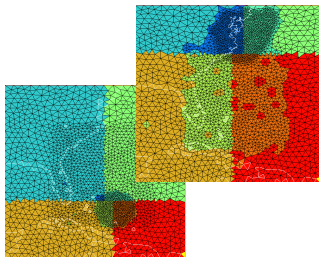
- **Performance** (time-to-solution): speedup critical national security applications by a factor of 10X to 40X
- **Programmability** (idea-to-first-solution): reduce cost and time of developing application solutions
- **Portability** (transparency): insulate research and operational application software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

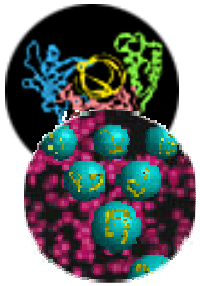


HPCS Program Focus Areas









Applications:

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

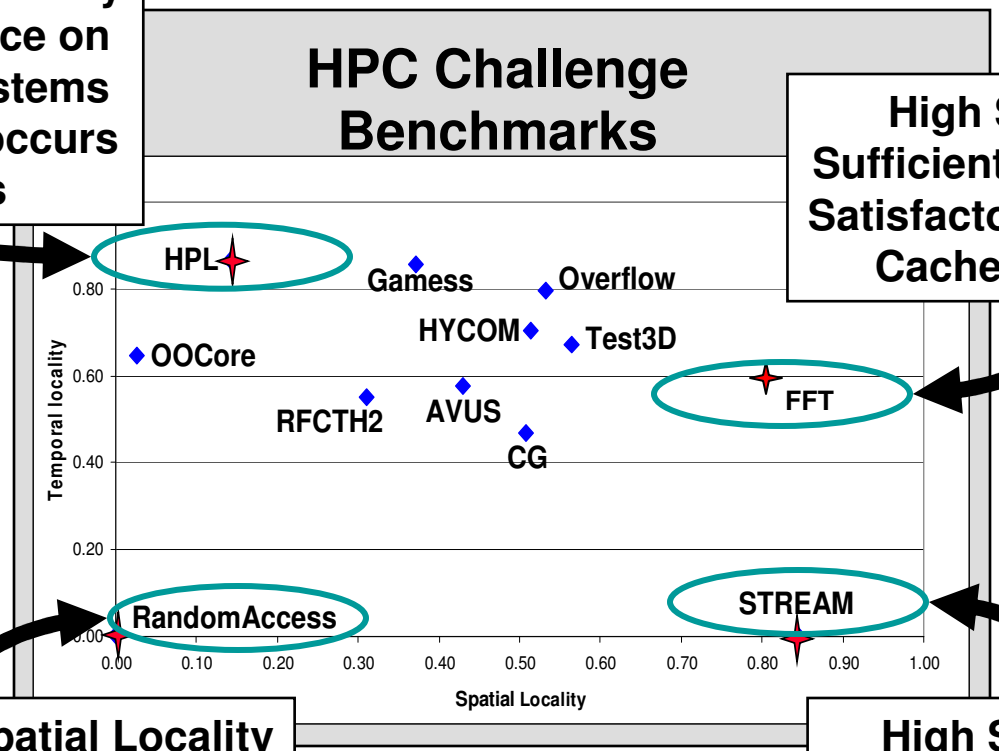
Fill the Critical Technology and Capability Gap

Today (late 80's HPC technology)…..*to*…..Future (Quantum/Bio Computing)

Motivation of the HPCC Design

High Temporal Locality
Good Performance on
Cache-based systems
Spatial Locality occurs
in registers

High Spatial Locality
Sufficient Temporal Locality
Satisfactory Performance on
Cache-based systems



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

Generated by
PMaC @ SDSC

High Spatial Locality
Moderate Performance on
Cache-based systems

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

Further information:
„Performance Modeling and Characterization"
@ San Diego Supercomputer Center
http://www.sdsc.edu/PMaC/

HPCS Performance Targets

HPC Challenge Benchmark	Corresponding Memory Hierarchy	HPCS Targets (improvement)
• Top500: solves a system $Ax = b$	Registers	
	Instr. Operands	2 Petaflops (8x)
• STREAM: vector operations $A = B + s \times C$	Cache	
	Blocks	6.5 Petabyte/s (40x)
• FFT: 1D Fast Fourier Transform $Z = FFT(X)$	Local Memory (bandwidth, latency)	0.5 Petaflops (200x)
	Messages	
• RandomAccess: random updates $T(i) = XOR(T(i), r)$	Remote Memory	64,000 GUPS (2000x)
	Pages	
	Disk	

Compared to BlueGene/L, Nov. 2006

- HPCS program has developed a new suite of benchmarks (HPC Challenge)
- Each benchmark focuses on a different part of the memory hierarchy
- HPCS program performance targets will flatten the memory hierarchy, improve real application performance, and make programming easier

HPCC as a Framework (1/2)

- **Many of the component benchmarks were widely used before**
  - **HPCC is more than a packaging effort**
  - **E.g., provides consistent verification and reporting**

- **Important:**
  Running these benchmarks on a single machine —
  with a single configuration and options
  - **The benchmark components are still 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 under the same conditions
      allows for a variety of analyses
      depending on end user needs

HPCC as a Framework (2/2)

- 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

- 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

HPCC Tests at a Glance

1. HPL

- High Performance Linpack
- Solving $Ax = b \quad A \in \mathbf{R}^{n \times n} \quad x,b \in \mathbf{R}$

2. DGEMM

- Double-precision General Matrix-matrix Multiply
- Computing $C \leftarrow \alpha AB + \beta C \quad A,B,C \in \mathbf{R}^{n \times n} \quad \alpha,\beta \in \mathbf{R}$
- Temporal/spatial locality: similar to HPL

3. STREAM

- measures sustainable memory bandwidth with vector operations
- COPY: $c=a$ SCALE: $b=\alpha c$
  ADD: $c=a+b$ TRIAD: $a=b+\alpha c$

4. PTRANS

- Parallel matrix TRANSpose
- Computing $A=A^T+B$
- Temporal/spatial locality: similar to EP-STREAM, but includes global communication

5. RandomAccess

- calculates a series of integer updates to random locations in memory
-
```
Ran = 1;
for (i=0; i<4*N; ++i) {
 Ran= (Ran<<1) ^
      (((int64_t)Ran < 0) ? 7:0);
 Table[Ran & (N-1)] ^= Ran;
}
```
- Use at least 64-bit integers
- About half of memory used for 'Table'
- Parallel look-ahead limited to 1024

6. FFT

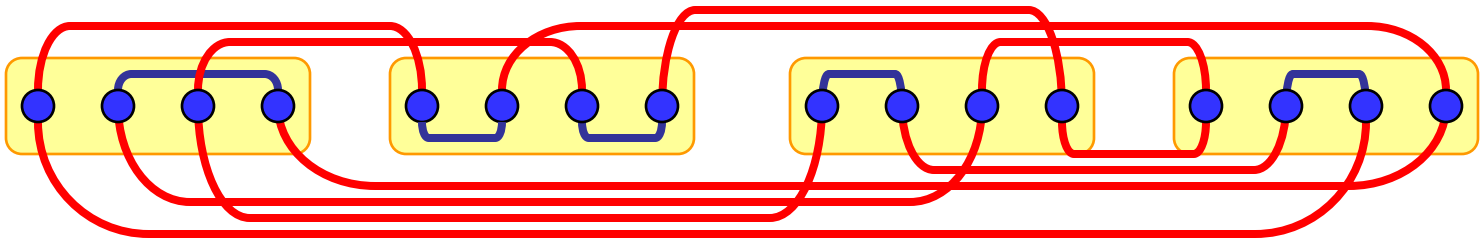
- Fast Fourier Transform
- Computing $z_k=\Sigma x_j$ exp$(-2\pi\sqrt{-1}\, jk/n) \quad x,z \in \mathbf{C}^n$

7. b_eff

- Patterns:
  - ping-pong,
  - natural ring, and
  - random ring patterns
- Bandwidth (w 2,000,000 bytes messages)
- Latency (with 8 bytes messages)

Random Ring Bandwidth

- Reflects communication patterns in unstructured grids
- And 2nd & 3rd dimension of a Cartesian domain decomposition
- On clusters of SMP nodes:
  - Some connections are inside of the nodes
  - Most connections are inter-node

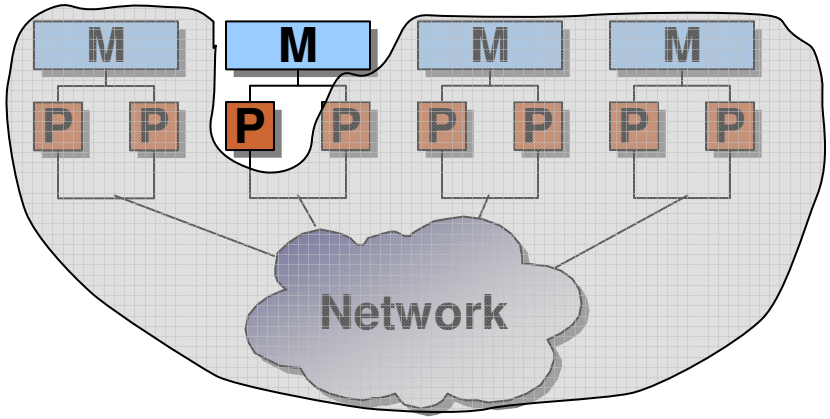


- Global benchmark → all processes participate
- Reported: bandwidth per process
- Accumulated bandwidth
  := bandwidth per process x #processes

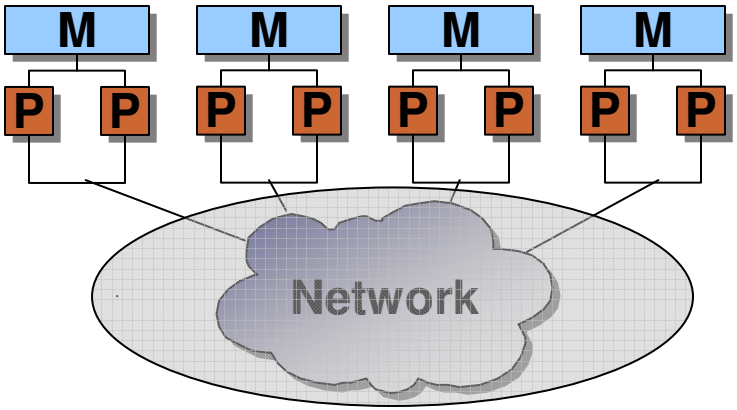
} similar to bi-section bandwidth

HPCC Testing Scenarios

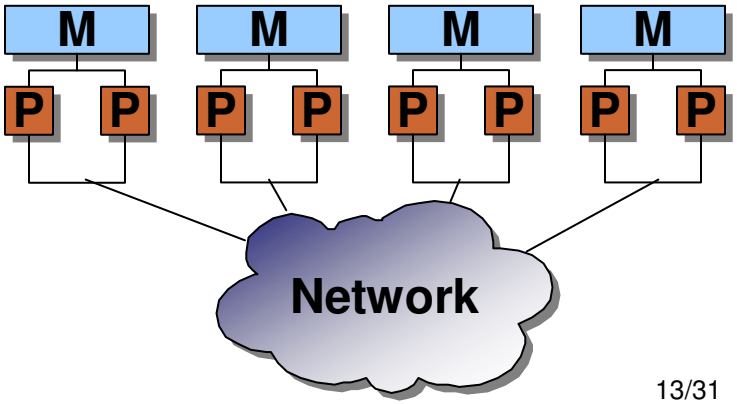
1. Local
   1. Only single process computes



2. Embarrassingly parallel
   1. All processes compute and do not communicate (explicitly)



3. Global
   1. All processes compute and communicate



4. Network only

Base vs. Optimized Submission

- **G-RandomAccess**

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

- **Base code: <u>Latency</u> based execution**
- **Optimization I: <u>UPC based code – only a few lines</u>**
  - **Optimization inside of UPC compiler / library**
  - **~125x improvement**
- **Optimization II: <u>Butterfly (MPI-based) algorithm</u>**
  - **<u>Bandwidth</u> based (packet size ~ 4 kB)**
  - **On BlueGene/L with special communication library: 537 x faster than "base"**

Results

- **HPCC Database**
  - **←upload of HPCC results**
  - **→Output through several interfaces**
    - **Web-output**
      - **Table with several subsets of kernels**
      - **Base / optimized / base+optimized**

Can be sorted by any column

Condensed Results - Base Runs Only - 132 Systems - Generated on Tue Jun 19 08:54:47 2007

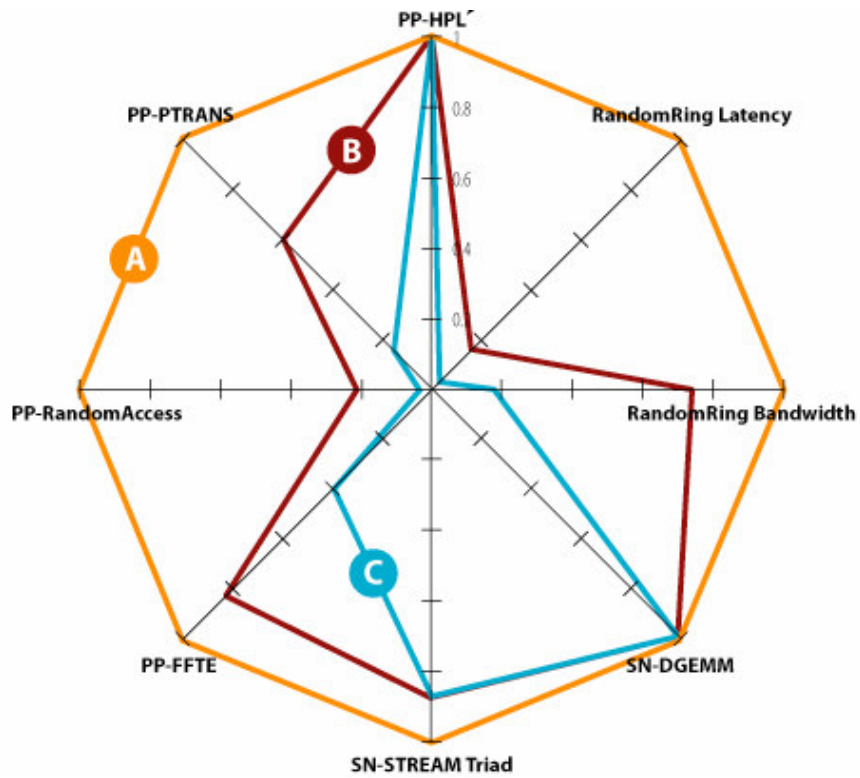
System Information System - Processor - Speed - Count - Threads - Processes				G-HPL	G-PTRANS	G-Random Access	G-FFTE	EP-STREAM Sys	EP-STREA Triad	EP-DGEMM	RandomRing Bandwidth	RandomRing Latency
MA/PT/PS/PC/TH/PR/CM/CS/IC/IA/SD				TFlop/s	GB/s	Gup/s	GFlop/s	GB/s	GB/s	GFlop/s	GB/s	usec
Cray Inc. Red Storm/XT3 AMD Opteron	2.4GHz	12960	125920	91.0350000	2356.9700	1.7401500	1554.0700	54840.499	2.1158	4.39939	0.05911	16.29
IBM Blue Gene/L PowerPC 440	0.7GHz	65536	165536	80.6830000	339.2840	0.0657312	2178.1100	53555.888	0.8172	1.85619	0.01084	8.84
IBM p5-575 Power5	1.9GHz	10240	110240	**57.8670000**	553.0090	0.1693440	842.5000	**55184.179**	5.3891	7.08562	0.11015	118.59
IBM p5-575 Power5	1.9GHz	8192	1 8192	45.7019000	2626.1700	0.3239760	908.6920	44455.936	5.4268	7.06423	0.08871	11.05
Cray Inc. XT3 Dual-Core AMD Opteron	2.6GHz	10404	110404	43.4033000	778.3850	0.8235630	1107.2100	25774.557	2.4774	4.78995	0.06937	14.32
IBM Blue Gene/L PowerPC 440	0.7GHz	65536	165536	37.3540000	4665.9100	0.1648600	1762.8200	62889.787	0.9596	2.47017	0.01039	8.62
Cray Inc. XT3 AMD Opteron	2.6GHz	8190	1 8190	35.1985000	603.1050	0.7308180	882.4230	17998.835	2.1977	4.79150	0.08599	14.22

- **As Excel or XML (all results)**
- **Comparing up to 6 platforms with a Kiviat diagram**

Kiviat Charts: Comparing Interconnects

- Comparing per-process values
- 8 fixed benchmark kernels
- Up to 6 systems
- Normalized:
  - 1 = best system at each kernel
- Example:
  - AMD Opteron clusters
    - 2.2 GHz
    - 64-processor cluster
  - Interconnects
    1. GigE
    2. Commodity
    3. Vendor
  - Cannot be differentiated based on:
    - HPL
    - Matrix-matrix multiply
- Available on HPCC website

Kiviat chart (radar plot)



HPCC Awards Overview

- **Goals**
  - **Increase awareness of HPCC benchmarks**
  - **Increase awareness of HPCS program and its goals**
  - **Increase number of HPCC submissions**
    - **Expanded view of largest supercomputing installations**
- **Means**
  - **HPCwire sponsorships and press coverage**
  - **HPCS mission partners' contribution**
  - **HPCS vendors' contribution**
- **Awards are presented at the SCxx HPC Challenge BOF**

HPCC Awards Rules

- Class 1: Best Performance
  - Figure of merit: raw system performance
  - Submission must be valid HPCC database entry
    - Side effect: populate HPCC database
  - 4 categories: HPCC components
    - HPL
    - STREAM-system
    - RandomAccess
    - FFT
  - Award certificates
    - 4x $500 from HPCwire

- Class 2: Most Productivity
  - Figure of merit: performance (50%) and elegance (50%)
    - Highly subjective
    - Based on committee vote
  - Submission must implement at least 3 out of 4 Class 1 tests
    - The more tests the better
  - Performance numbers are a plus
  - The submission process:
    - Source code
    - “Marketing brochure”
    - SC06 BOF presentation
  - Award certificate
    - $1500 from HPCwire

HPCwire

SC|06 HPCC Award – Class 1

G-HPL	Achieved	System	Affiliation	Submitter
1st place	**259 Tflop/s**	IBM BG/L	DOE/NNSA/LLNL	Tom Spelce
1st runner up	67 Tflop/s	IBM BG/L	IBM T.J. Watson	John Gunnels
2nd runner up	57 Tflop/s	IBM p5-575	LLNL	Charles Grassl
HPCS goal:	**2000 Tflop/s**	**= current 1st place x 8**		

EP-STREAM-Triad	Achieved	System	Affiliation	Submitter
1st place	**160 TB/s**	IBM BG/L	DOE/NNSA/LLNL	Tom Spelce
1st runner up	55 TB/s	IBM p5-575	LLNL	Charles Grassl
2nd runner up	43 TB/s	Cray XT3	SNL	Courtenay Vaughan
HPCS goal:	**6500 TB/s**	**= current 1st place x 40**		

G-FFT	Achieved	System	Affiliation	Submitter
1st place	**2.311 Tflop/s**	IBM BG/L	DOE/NNSA/LLNL	Tom Spelce
1st runner up	1.122 Tflop/s	Cray XT3 Dual	ORNL	Jeff Larkin
2nd runner up	1.118 Tflop/s	Cray XT3	SNL	Courtenay Vaughan
HPCS goal:	**500.0 Tflop/s**	**= current 1st place x 200**		

G-RandomAccess	Achieved	System	Affiliation	Submitter
1st place	**35 GUPS**	IBM BG/L	DOE/NNSA/LLNL	Tom Spelce
1st runner up	17 GUPS	IBM BG/L	IBM T.J. Watson	John Gunnels
2nd runner up	10 GUPS	Cray XT3 Dual	ORNL	Jeff Larkin
HPCS goal:	**65000 Tflop/s**	**= current 1st place x 2000**		

SC|06 HPCC Awards Class 2

Language	HPL	Random Access	STREAM	FFT	PTRANS	DGEMM	
Cilk	√	√	√	√	√	√	Best Overall Productivity
x|UPC	√	√		√			Best Productivity in Performance
Parallel Matlab	√		√	√			Best Productivity and Elegance
MC#	√		√	√			Best Student Paper
Chapel		√	√	√			Honorable Mention
X10		√	√	√			

Augmenting TOP500's 26th Edition

Nov. 2005

	Computer	Rmax	HPL	PTRANS	STREAM	FFT	GUPS	Latency	B/W
1	BlueGene/L	281	259	374	160	2311	35.5	6	0.2
2	BGW (**)	91	84	172	50	1235	21.6	5	0.2
3	ASC Purple	63	58	576	44	967	0.2	5	3.2
4	Columbia (**)	52	47	91	21	230	0.2	4	1.4
5	Thunderbird	38							
6	Red Storm	36	33	1813	44	1118	1.0	8	1.2
7	Earth Simulator	36							
8	MareNostrum	28							
9	Stella	27							
10	Jaguar	21	20	944	29	855	0.7	7	1.2

Augmenting TOP500's 28th Edition with HPCC

Nov. 2006

	Computer	Rmax TFlop/s	G-HPL TFlop/s	G-PTRANS GB/s	EP-STREAM Triad TB/s	G-FFT GFlop/s	G-Random Access GUPS	PingPong Latency µs	PingPong Bandwidth GB/s
1	BlueGene/L	280.6	259.2 80.7	4665.9 339.3	160 57	2311 2178	35.47 0.066	5.92 µs 7.07 µs	0.158 0.157
2	Red Storm Cray XT3 Opteron dual-core	101.4	91.0	2357.0	55	1554	29.81 1.74	7.16 µs	2.024
3	BGW BlueGene/L IBM/Watson (** 32768→ 40960)	91	83.9 ** 39 **	171.55 ** 109 **	50 ** 37 **	1235** 1391**	21.61** 0.348**	4.95 µs	0.159
4	ASC Purple IBM p5 (** 10240→ 12208)	75.8	69 **	659 **	66 **	1004**	1.02(*) 0.202**	5.10 µs	3.184
5	MareNostrum	62.63							
6	Thunderbird	53.00							
7	Tera-10	52.84							
8	Columbia SGI Altix Infiniband (** 2024→ 10160)	51.87	47 **	91.31 **	20 **	229 **	0.25 **	4.23 µs	0.896
9	TSUBAME	47.38							
10	Jaguar Cray XT3 Opteron dual-core	43.48	43.40	2039 778	27	1127 1107	10.67 0.82	6.69 µs	1.15

Upper values = optimized
Bottom values = base
(*) = not published in HPCC database
(**) = extrapolated: #CPUs HPCC → #CPUs Linpack

Augmenting TOP500's 28th Edition with HPCC

Nov. 2006

	Computer	Rmax TFlop/s	HPL TFlop/s	STREAM Triad TB/s	Random Ring BW global TB/s	Random Ring BW per proc. GB/s	Ping Pong GB/s	Random Ring Latency µs	Ping Pong Latency µs
1	BlueGene/L	280.6	259.2 80.7	160 57	0.727 0.710	0.011 0.011	0.158 0.157	7.78 8.84	5.92 7.07
2	Red Storm Cray XT3 Opteron dual-core	101.4	91.0	55	1.532	0.059	2.024	16.29	7.16
3	BGW BlueGene/L IBM/Watson (** 32768→ 40960)	91	83.9 39 **	171.55 109 **	0.490 **	0.012	0.159	9.51	4.95
4	ASC Purple IBM p5 HPS (** 10240→ 12208)	75.8	69 **	66 **	1.345 **	0.110	3.154	118.59	5.10
5	MareNostrum	62.63							
6	Thunderbird	53.00							
7	Tera-10	52.84							
8	Columbia SGI Altix Infiniband (** 2024→ 10160)	51.87	47 **	20 **	1.247 **	0.122	0.896	6.98	4.23
9	TSUBAME	47.38							
10	Jaguar Cray XT3 Opteron dual-core	43.48	43.40	27	0.722	0.069	1.15	14.32	6.69

Global values (i.e., accumulated per system)

B/W ratio PingPing / Random 7 - 34

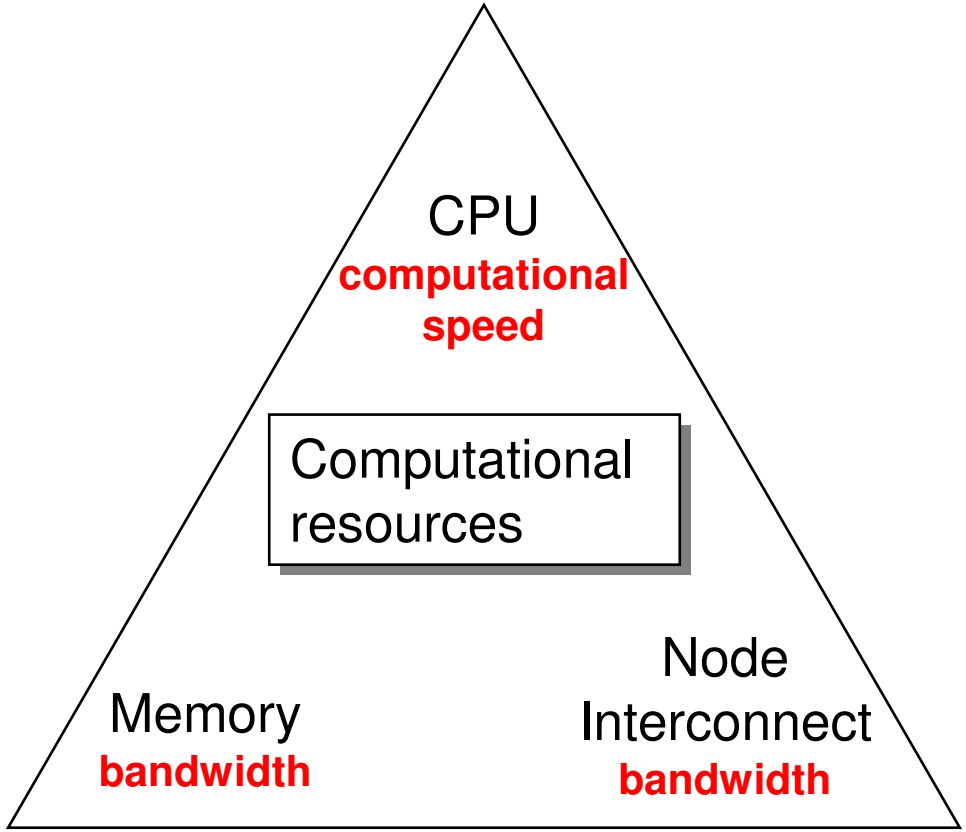
Latency Random / PingPong ratio: 1.3 – 2.3; 23 (Purple)

Upper values = optimized
Bottom values = base

(*) = not published in HPCC DB
(**) = extrapolated

HPCC and Computational Resources

HPL
(Jack Dongarra)



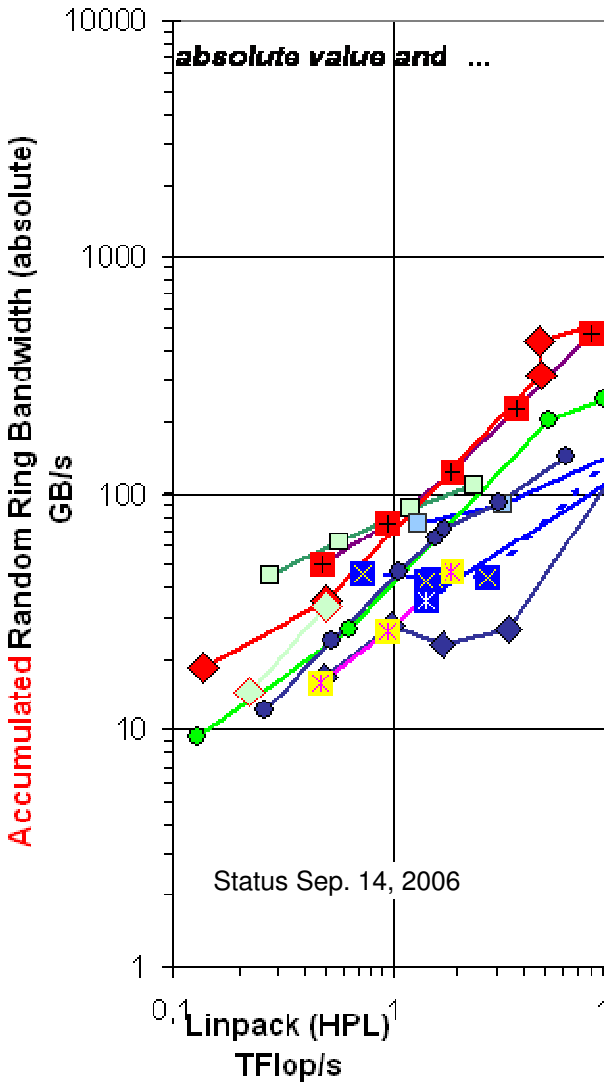
STREAM
(John McCalpin)

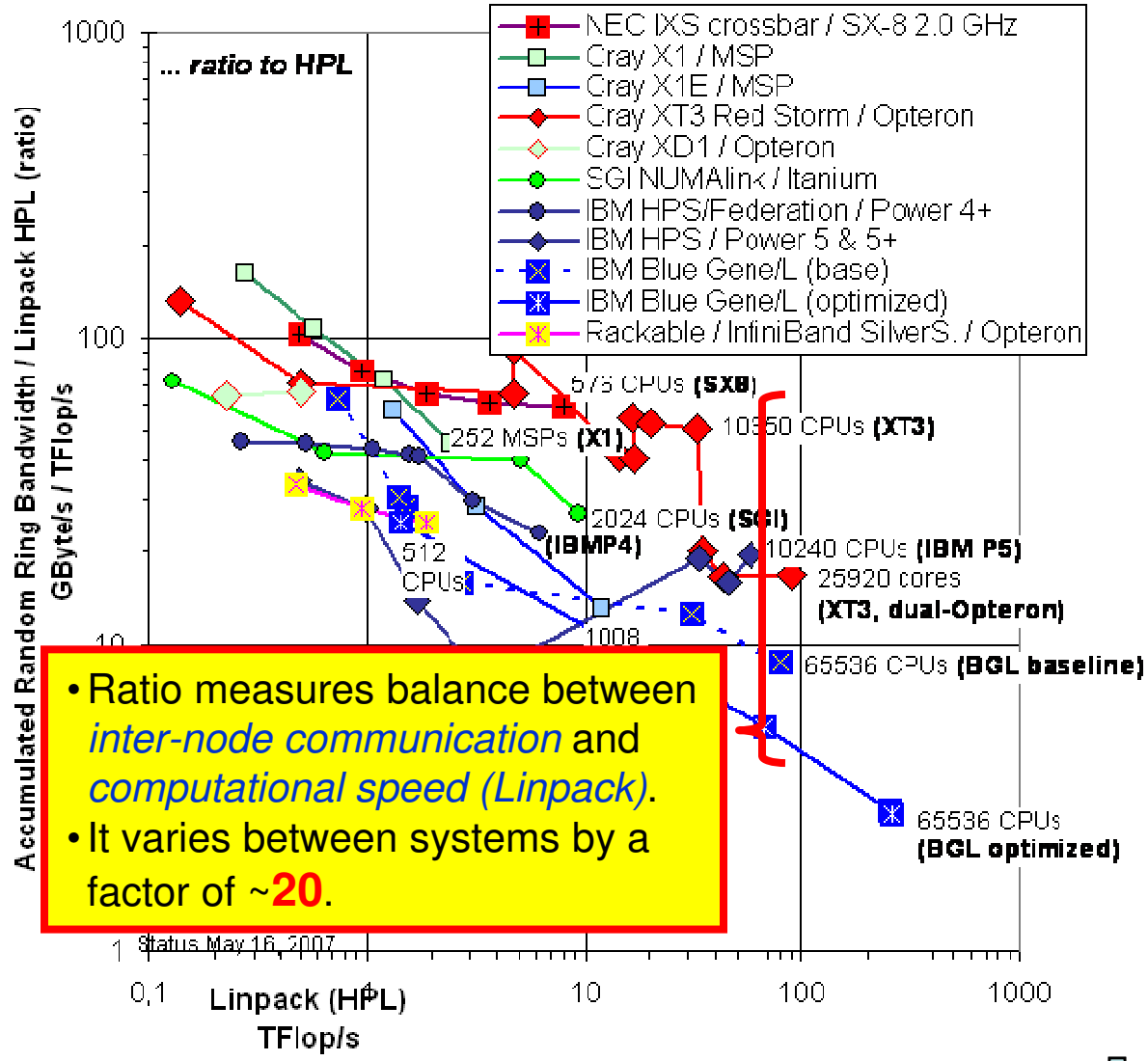
Random & Natural
Ring
Bandwidth & Latency
(my part of the HPCC Benchmark Suite)

Balance Analysis with HPCC Data

- **Balance can be expressed as a set of ratios**
  - **e.g., accumulated memory bandwidth / accumulated Tflop/s rate**
- **Basis**
  - Linpack (HPL) → Computational Speed
  - Random Ring Bandwidth → Inter-node communication
  - Parallel STREAM Copy or Triad → Memory bandwidth
- **Be careful:**
  - **Balance calculation always with accumulated data on the total system (Global or EP)**
  - **Random Ring B/W: *per process value* must be multiplied by *#processes***

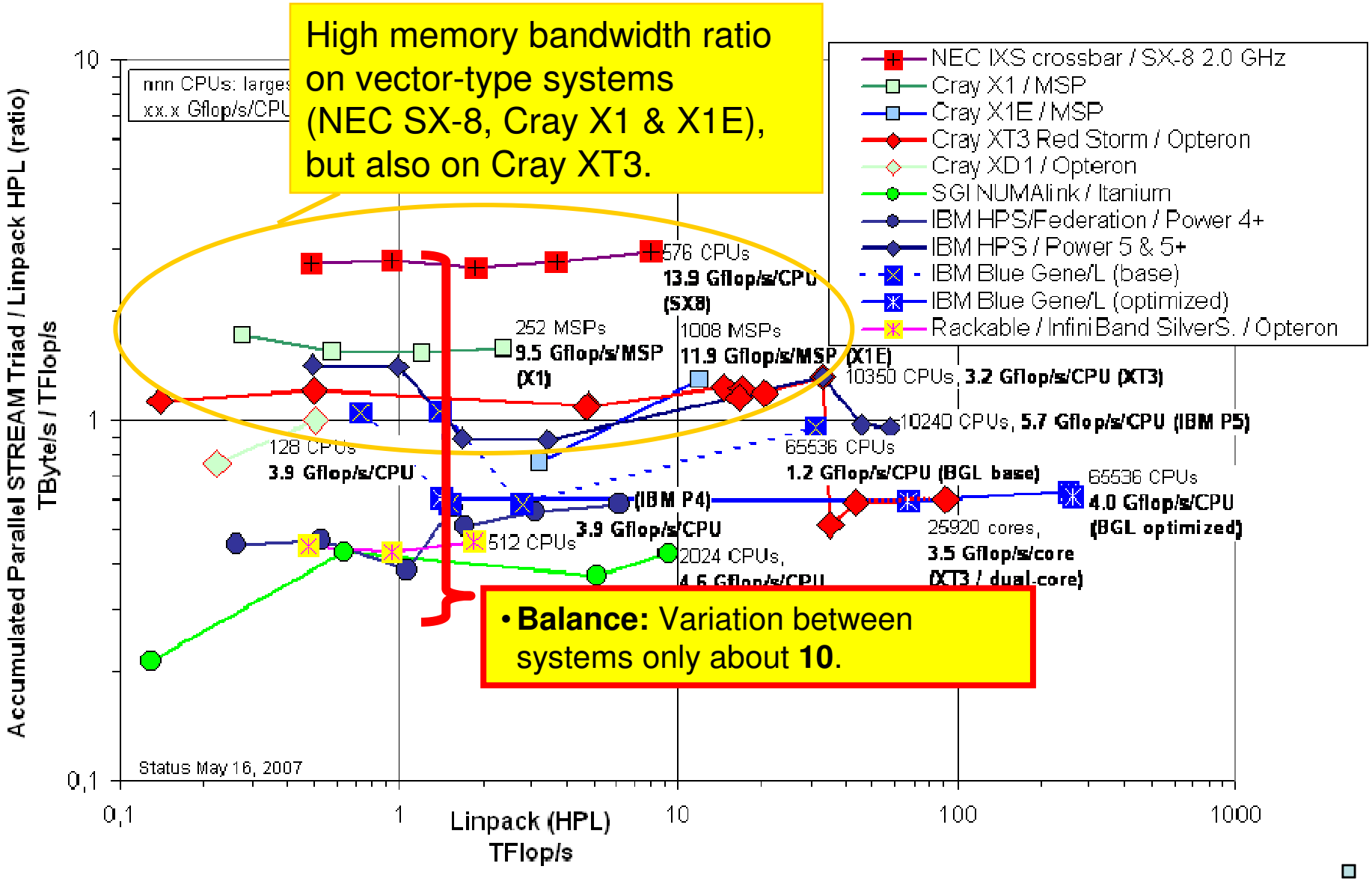
Balance: Random Ring B/W and HPL



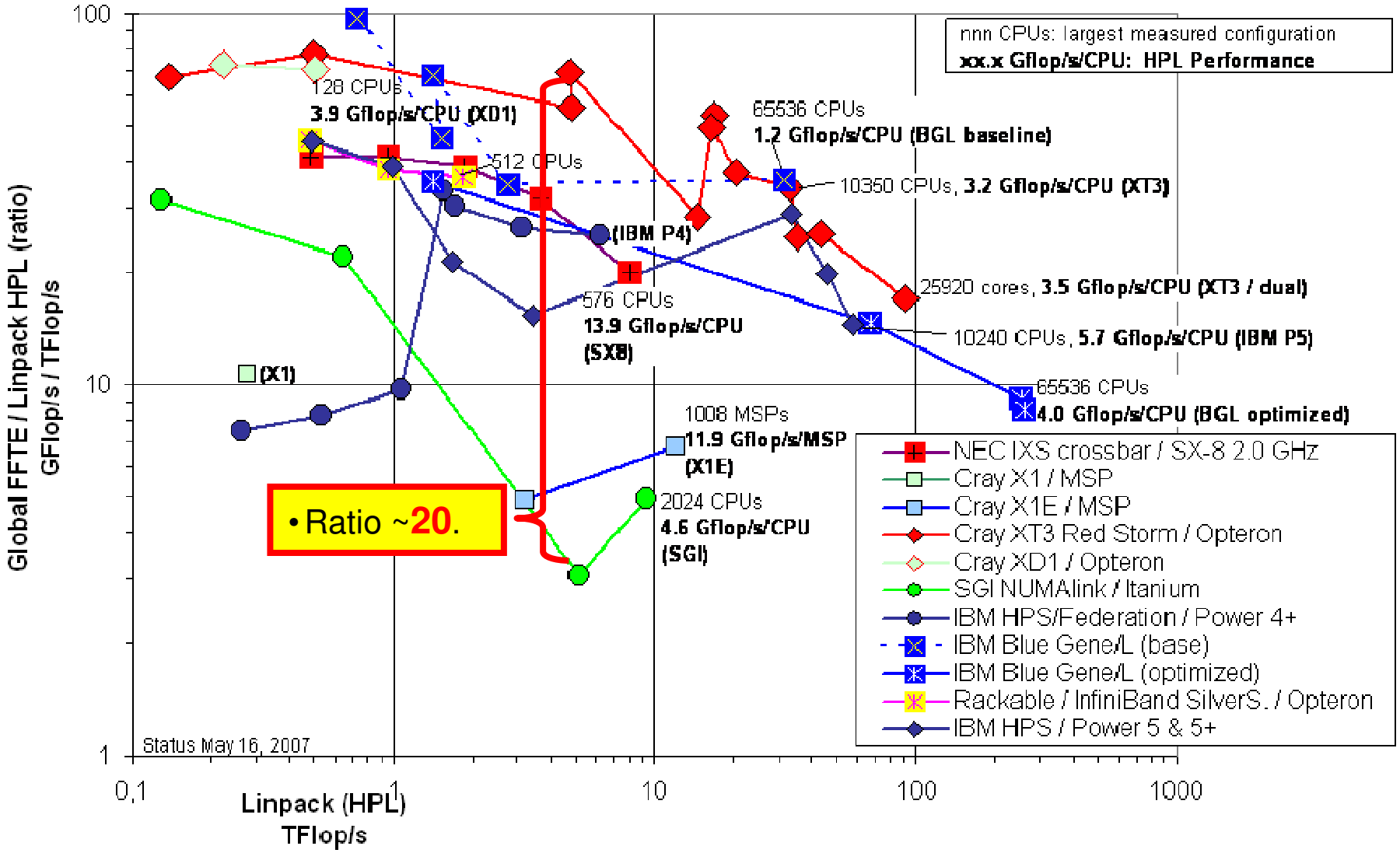


- Ratio measures balance between *inter-node communication* and *computational speed (Linpack)*.
- It varies between systems by a factor of ~**20**.

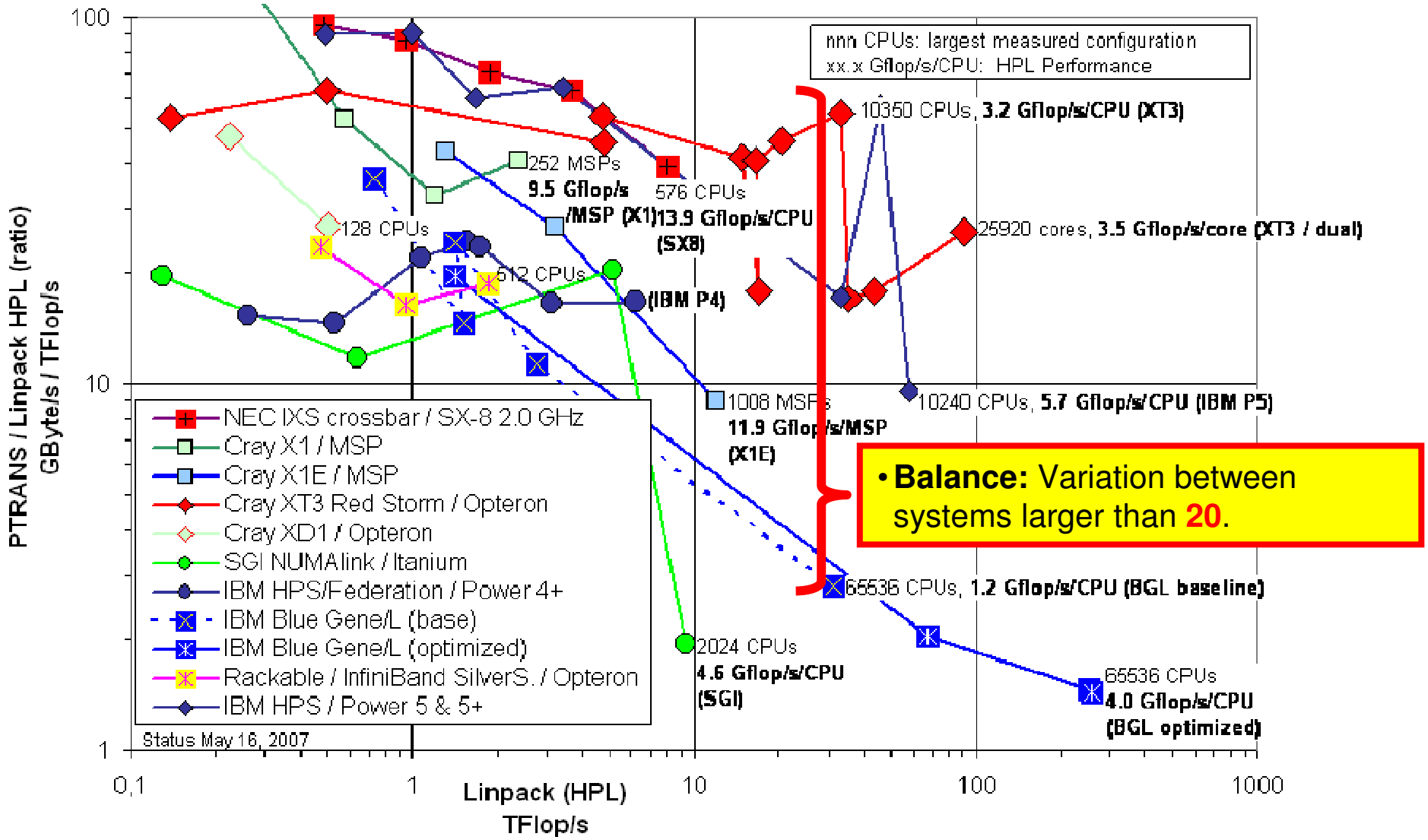
Balance: Memory and CPU Speed



Balance: FFT and CPU



Balance: PTRANS and CPU



Acknowledgments

- **Thanks to**
  - **all persons and institutions that have uploaded HPCC results.**
  - **Jack Dongarra and Piotr Luszczek for inviting me into the HPCC development team.**
  - **Matthias Müller, Sunil Tiyyagura and Holger Berger for benchmarking on the SX-8 and SX-6 and discussions on HPCC.**
  - **Nathan Wichmann from Cray for additional Cray XT3 and X1E data.**

- **References**

  - **Piotr Luszczek, David Bailey, Jack Dongarra, Jeremy Kepner, Robert Lucas, Rolf Rabenseifner, Daisuke Takahashi**: The HPC Challenge (HPCC) Benchmark Suite. **Tutorial at SC|06.**
  - **S. Saini, R. Ciotti,B. Gunney, Th. Spelce, A. Koniges, D. Dossa, P. Adamidis, R. Rabenseifner, S. Tiyyagura, M, Müller, and R. Fatoohi:** Performance Evaluation of Supercomputers using HPCC and IMB Benchmarks. **In the proceedings of the IPDPS 2006 Conference.**
  - **R. Rabenseifner, S. Tiyyagurra, M. Müller:** Network Bandwidth Measurements and Ratio Analysis with the HPC Challenge Benchmark Suite (HPCC). **Proceedings of the 12th European PVM/MPI Users' Group Meeting, EuroPVM/MPI 2005**
  - **http://icl.cs.utk.edu/hpcc/**

Conclusions

- **HPCC is an interesting basis for**
  - **benchmarking computational resources**
  - **Augmenting TOP500**
  - **analyzing the balance of a system**
  - **scaling with the number of processors**
  - **with respect to applications' needs (e.g., locality characteristics)**
- **HPCC helps to show the strength and weakness of super-computers**
- **Future super computing should not focus only on Pflop/s in the TOP500**
  - **Memory and network bandwidth are as same as important to predict real application performance**

Copy of the slides:

http://www.hlrs.de/people/rabenseifner/publ/publications.html#SPEC2007

Appendix

HPCC Tests - HPL

- **HPL = High Performance Linpack**
- **<u>Objective</u>: solve system of linear equations**

$$Ax=b \qquad A \in \boldsymbol{R}^{n \times n} \qquad x,b \in \boldsymbol{R}$$

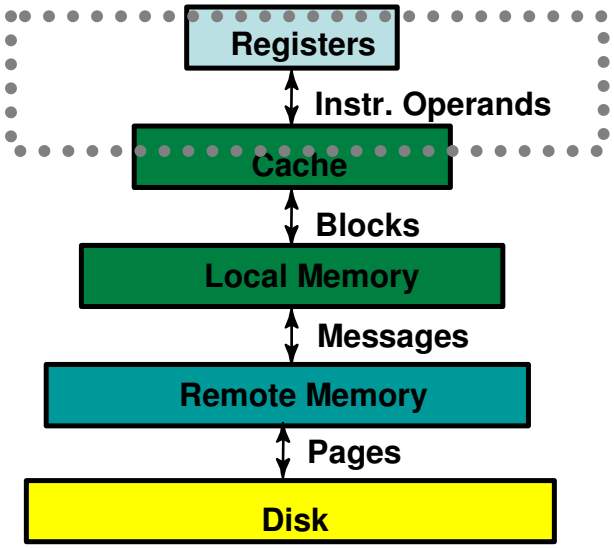
- **<u>Method</u>: LU factorization with partial row pivoting**
- **<u>Performance</u>:** $({}^{2}/_{3} n^3 + {}^{3}/_{2} n^2) / t$
- **<u>Verification</u>: scaled residuals must be small**

$$\| Ax-b \| / (\varepsilon \, \|A\| \, \|x\| \, n)$$

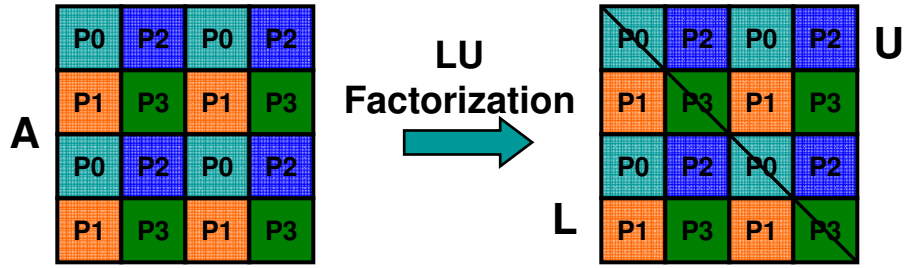
- **<u>Restrictions</u>:**
  - **No complexity reducing matrix-multiply**
    - **(Strassen, Winograd, etc.)**
  - **64-bit precision arithmetic through-out**
    - **(no mixed precision with iterative refinement)**

HPCC HPL: Further Details

- High Performance Linpack (HPL) solves a system Ax = b
- Core operation is a LU factorization of a large MxM matrix
- Results are reported in floating point operations per second (flop/s)



Parallel Algorithm



2D block cyclic distribution
is used for load balancing

- Linear system solver (requires all-to-all communication)
- Stresses local matrix multiply performance
- DARPA HPCS goal: 2 Pflop/s (8x over current best)

HPCC Tests - DGEMM

- **DGEMM = Double-precision General Matrix-matrix Multiply**
- **<u>Objective</u>: compute matrix**

  $C \leftarrow \alpha AB + \beta C \qquad A,B,C \in \boldsymbol{R}^{n \times n} \qquad \alpha,\beta \in \boldsymbol{R}$
- **<u>Method</u>: standard multiply (maybe optimized)**
- **<u>Performance</u>:** $2n^3/t$
- **<u>Verification</u>: Scaled residual has to be small**

  $\| x - y \| / (\varepsilon\, n \| y \|)$

  where x **and** y **are vectors resulting from multiplication by a random vector of left and right hand size of the objective expression**
- **<u>Restrictions</u>:**
  - **No complexity reducing matrix-multiply**
    - **(Strassen, Winograd, etc.)**
  - **Use only 64-bit precision arithmetic**
- **<u>Temporal/spatial Locality</u>: similar to HPL**

HPCC Tests - STREAM

- **STREAM is a test that measures sustainable memory bandwidth (in Gbyte/s) and the corresponding computation rate for four simple vector kernels**
- **<u>Objective</u>: set a vector to a combination of other vectors**

  COPY: $c = a$

  SCALE: $b = \alpha c$

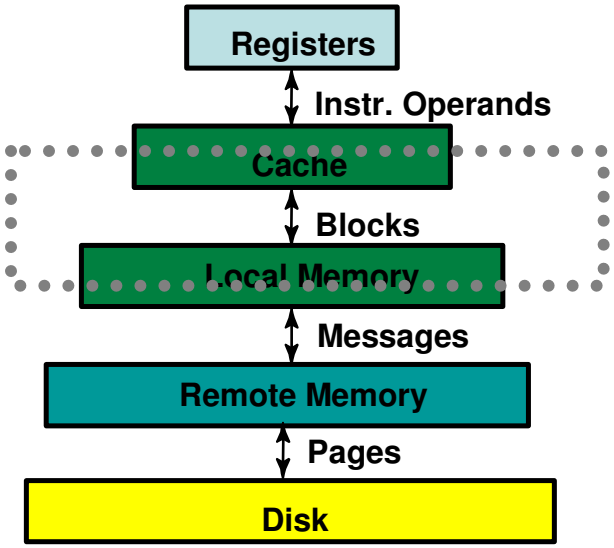
  ADD: $c = a + b$

  TRIAD: $a = b + \alpha c$
- **<u>Method</u>: simple loop that preserves the above order of operations**
- **<u>Performance</u>:** $2n/t$ **or** $3n/t$
- **<u>Verification</u>: scalre residual of computed and reference vector needs to be small**

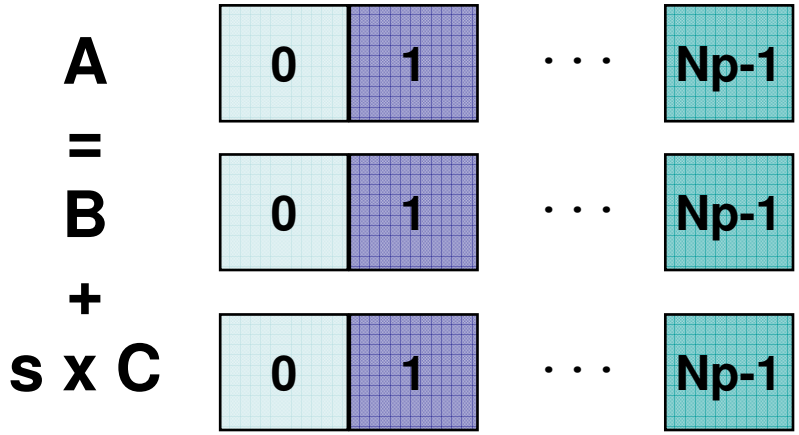
  $\| x - y \| / (\varepsilon n \| y \|)$
- **<u>Restrictions</u>:**
  - **Use only 64-bit precision arithmetic**

HPCC STREAM: Further Details

- Performs scalar multiply and add
- Results are reported in bytes/second



Parallel Algorithm



- Basic operations on large vectors (requires no communication)
- Stresses local processor to memory bandwidth
- DARPA HPCS goal: 6.5 Pbyte/s (40x over current best)

HPCC Tests - PTRANS

- **PTRANS = Parallel TRANSpose**
- **<u>Objective</u>: update matrix with sum of its transpose and another matrix**

  $A = A^T + B \qquad A, B \in \mathbf{R}^{n \times n}$
- **<u>Method</u>: standard distributed memory algorithm**
- **<u>Performance</u>:** n^2/t
- **<u>Verification</u>: scaled residual between computed and reference matrix needs to be small**

  $\| A_0 - A \| / (\varepsilon \, n \, \| A_0 \|)$
- **<u>Restrictions</u>:**
  - **Use only 64-bit precision arithmetic**
  - **The same data distribution method as HPL**
- **<u>Temporal/spatial Locality</u>: similar to EP-STREAM, but includes global communication**

HPCC Tests - RandomAccess

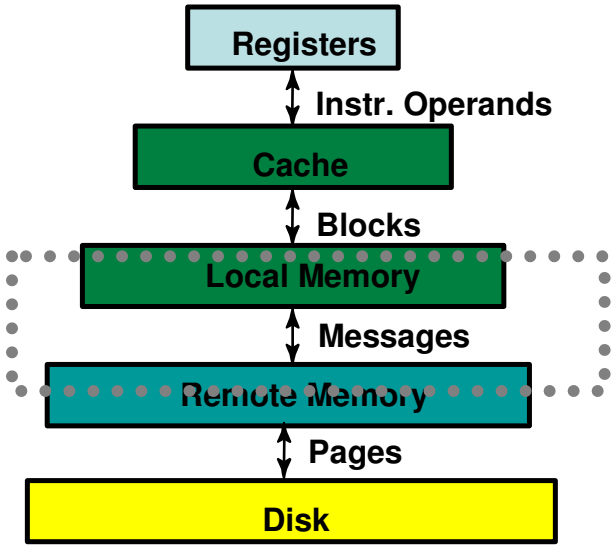
- **RandomAccess calculates a series of integer updates to random locations in memory**
- **<u>Objective</u>: perform computation on Table**

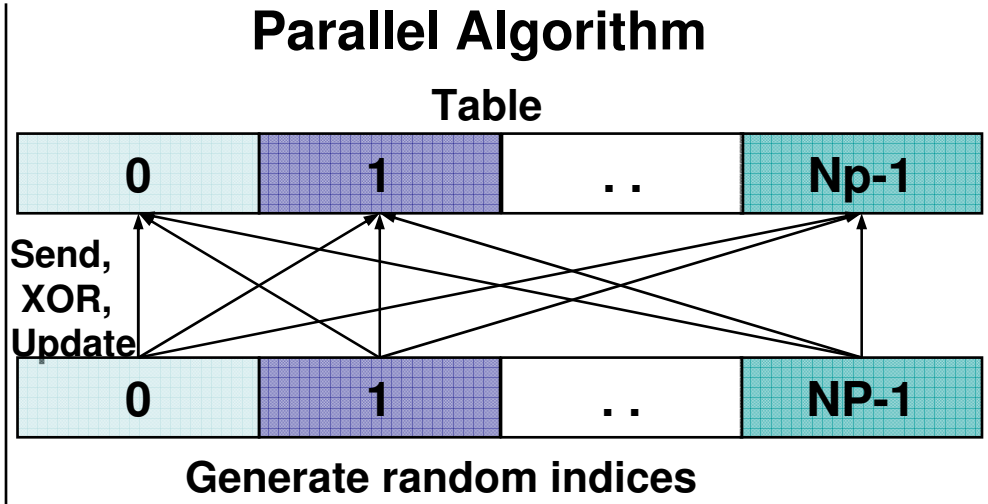
```
Ran = 1;
for (i=0; i<4*N; ++i) {
  Ran= (Ran<<1) ^ (((int64_t)Ran < 0) ? 7:0);
  Table[Ran & (N-1)] ^= Ran;
}
```

- **<u>Method</u>: loop iterations may be independent**
- **<u>Performance</u>:** $4N/t$
- **<u>Verification</u>: up to 1% of updates can be incorrect**
- **<u>Restrictions</u>:**
  - **Use at least 64-bit integers**
  - **About half of memory used for 'Table'**
  - **Parallel look-ahead limited to 1024 (limit locality)**

HPCC RandomAccess: Further Details

- Randomly updates N element table of unsigned integers
- Each processor generates indices, sends to all other processors, performs XOR
- Results are reported in Giga Updates Per Second (GUPS)





- **Randomly updates memory (requires all-to-all communication)**
- **Stresses interprocessor communication of *small* messages**
- **DARPA HPCS goal: 64,000 GUPS (2000x over current best)**

HPCC Tests - FFT

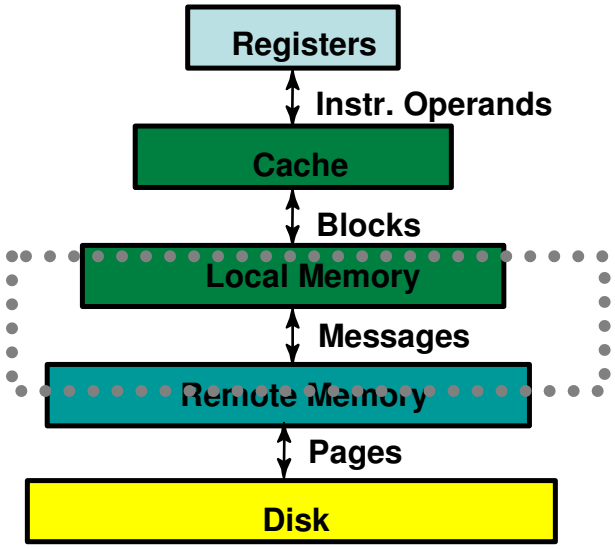
- **FFT = Fast Fourier Transform**
- **<u>Objective</u>: compute discrete Fourier Transform**

  $z_k = \Sigma x_j$ **exp**$(-2\pi\sqrt{-1}\, jk/n)$ $\qquad x,z \in \mathbf{C}^n$
- **<u>Method</u>: any standard framework (maybe optimized)**
- **<u>Performance</u>:** $5n\log_2 n/t$
- **<u>Verification</u>: scaled residual for inverse transform of computed vector needs to be small**

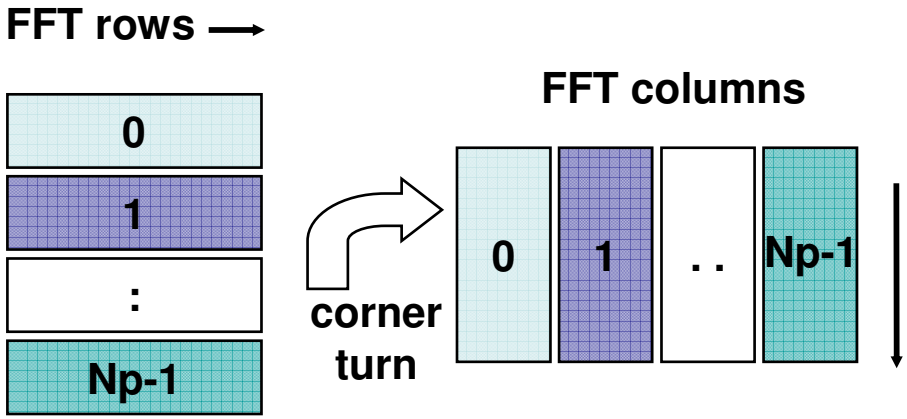
  $\| x - x^{(0)} \| / (\varepsilon \log_2 n)$
- **<u>Restrictions</u>:**
  - **Use only 64-bit precision arithmetic**
  - **Result needs to be in-order (not bit-reversed)**

HPCC FFT: Further Details

- 1D Fast Fourier Transforms an N element complex vector
- Typically done as a parallel 2D FFT
- Results are reported in floating point operations per second (flop/s)



Parallel Algorithm



- FFT a large complex vector (requires all-to-all communication)
- Stresses interprocessor communication of *large* messages
- DARPA HPCS goal: 0.5 Pflop/s (200x over current best)

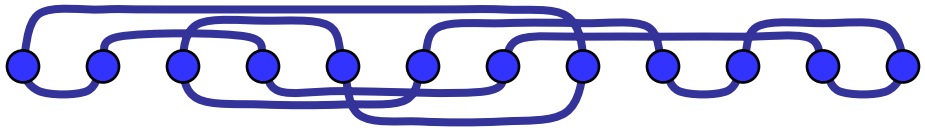
HPCC Tests – b_eff

- b_eff measures effective bandwidth and latency of the interconnect
- Objective: exchange 8 (for latency) and 2000000 (for bandwidth) messages in
  - ping-pong,
  - natural ring, and
  - random ring patterns
- Method: use standard MPI point-to-point routines
- Performance: n/t (for bandwidth)
- Verification: simple checksum on received bits
- Restrictions:
  - The messaging routines have to conform to the MPI standard

HPCC b_eff: Further Details

- **Parallel communication pattern on <u>all</u> MPI processes:**
  - **Natural ring**

  - **Random ring**



- **Bandwidth per process**
  - **Accumulated message size / wall-clock time / number of processes**
  - **On each connection messages in both directions**
  - **With 2xMPI_Sendrecv and MPI non-blocking → best result is used**
  - **Message size = 2,000,000 bytes**
- **Latency**
  - **Same patterns, message size = 8 bytes**
  - **Wall-clock time / (number of sendrecv per process)**

Naming Conventions

S	HPL	system
EP	STREAM	CPU
G	FFT	thread
	…	
	RandomAccess	

Examples:

1. G-HPL
2. S-STREAM-system

Measured on a single process

Per system, i.e., extrapolated to the total system

Base vs. Optimized Run

- 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

SC|05 HPCC Awards Class 2

Language	HPL	RandomAccess	STREAM	FFT
Python+MPI		√	√	
pMatlab	√	√	√	√
Cray MTA C		√		√
UPCx3	√	√	√	
Cilk	√	√	√	√
Parallel Matlab	√	√	√	√
MPT C	√			√
OpenMP, C++		√	√	
StarP	√		√	
HPF	√			√

Sample submission from committee members

Winners

Finalists

Augmenting TOP500's 27th Edition with HPCC

June 2006

	Computer	Rmax	HPL	PTRANS	STREAM	FFT	GUPS	Latency	B/W
1	BlueGene/L	280.6	259.2	4665.9	160	2311	35.47	5.92	0.159
2	BGW (**)	91	83.9	171.55	50	1235	21.61	4.70	0.159
3	ASC Purple	75.8	57.9	553	55	842	1.03	5.1	3.184
4	Columbia (**)	51.87	46.78	91.31	20	229	0.25	4.23	0.896
5	Tera-10	42.9							
6	Thunderbird	38.27							
7	Fire x4600	38.18							
8	BlueGene eServer	37.33							
9	Red Storm	36.19	32.99	1813.06	43.58	1118	1.02	7.97	1.149
10	Earth Simulator	35.86							

Balance: Random Ring B/W and CPU Speed