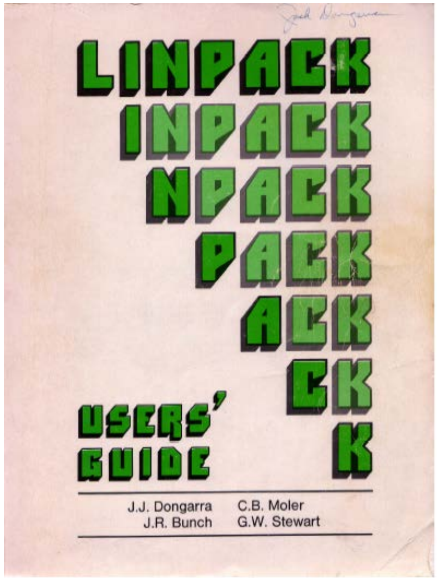
TOWARD A NEW (ANOTHER) METRIC FOR RANKING HIGH PERFORMANCE COMPUTING SYSTEMS

Jack Dongarra & Piotr Luszczek
University of Tennessee/ORNL

Michael Heroux
Sandia National Labs

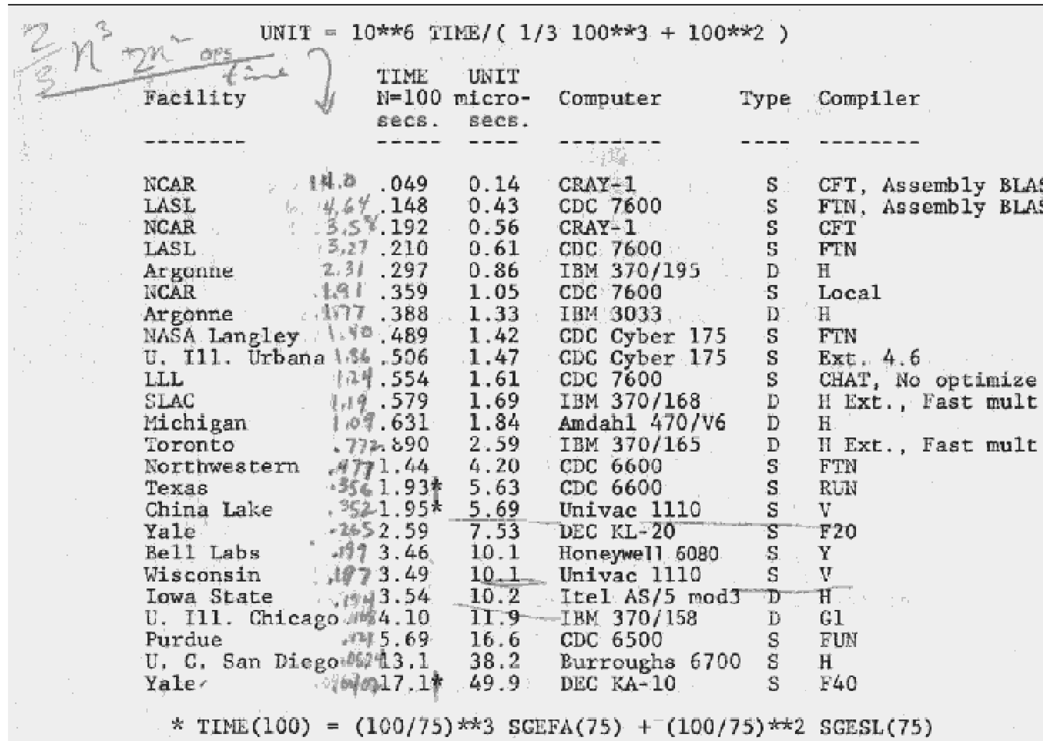
Confessions of an Accidental Benchmarker

- Appendix B of the LINPACK Users' Guide
  - Designed to help users extrapolate execution LINPACK software package
- First benchmark report from 1977;
  - Cray 1 to DEC PDP-10



Started 36 Years Ago

- In the late 70's the fastest computer ran LINPACK at 14 Mflop/s
- In the late 70's floating point operations were expensive compared to other operations and data movement
- Matrix size, n = 100
  - That's what would fit in memory



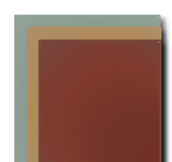
$\frac{2}{3}n^3 + 2n^2$ ops / time

UNIT = 10**6 TIME/(1/3 100**3 + 100**2)

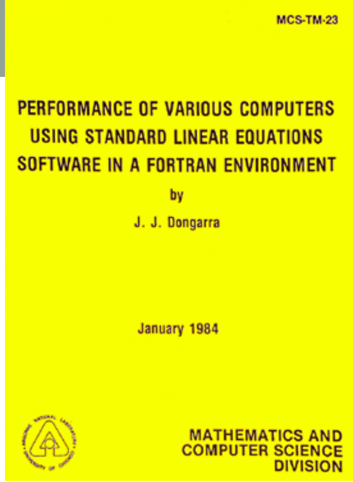
Facility		TIME N=100 secs.	UNIT micro-secs.	Computer	Type	Compiler
NCAR	14.3	.049	0.14	CRAY-1	S	CFT, Assembly BLAS
LASL	4.64	.148	0.43	CDC 7600	S	FTN, Assembly BLAS
NCAR	3.55	.192	0.56	CRAY-1	S	CFT
LASL	3.27	.210	0.61	CDC 7600	S	FTN
Argonne	2.31	.297	0.86	IBM 370/195	D	H
NCAR	1.91	.359	1.05	CDC 7600	S	Local
Argonne	1.77	.388	1.33	IBM 3033	D	H
NASA Langley	1.40	.489	1.42	CDC Cyber 175	S	FTN
U. Ill. Urbana	1.36	.506	1.47	CDC Cyber 175	S	Ext. 4.6
LLL	1.24	.554	1.61	CDC 7600	S	CHAT, No optimize
SLAC	1.19	.579	1.69	IBM 370/168	D	H Ext., Fast mult.
Michigan	1.09	.631	1.84	Amdahl 470/V6	D	H
Toronto	.772	.890	2.59	IBM 370/165	D	H Ext., Fast mult.
Northwestern	.477	1.44	4.20	CDC 6600	S	FTN
Texas	.356	1.93*	5.63	CDC 6600	S	RUN
China Lake	.352	1.95*	5.69	Univac 1110	S	V
Yale	.265	2.59	7.53	DEC KL-20	S	F20
Bell Labs	.197	3.46	10.1	Honeywell 6080	S	Y
Wisconsin	.197	3.49	10.1	Univac 1110	S	V
Iowa State	.194	3.54	10.2	Itel AS/5 mod3	D	H
U. Ill. Chicago	.166	4.10	11.9	IBM 370/158	D	G1
Purdue	.124	5.69	16.6	CDC 6500	S	FUN
U. C. San Diego	.0524	13.1	38.2	Burroughs 6700	S	H
Yale	[illegible]	17.1*	49.9	DEC KA-10	S	F40

* TIME(100) = (100/75)**3 SGEFA(75) + (100/75)**2 SGESL(75)

- LINPACK code is based on "right-looking" algorithm:
  - $O(n^3)$ Flop/s and $O(n^3)$ data movement



Benchmarks Evolve: From LINPACK to HPL to TOP500



- LINPACK Benchmark report, ANL TM-23, 1984
  - **Performance of Various Computers Using Standard Linear Equations Software**, listed about 70 systems.
- Over time the LINPACK Benchmark when through a number of changes.
  - Began with Fortran code, run the code as is, no changes, N = 100 (Table 1)
  - Later N = 1000 introduced, hand coding to allow for optimization and parallelism (Table 2)
  - Timing harness provided to generate matrix, check the solution
  - The basic algorithm, GE/PP, remained the same.
- 1989 started putting together Table 3 (Toward Peak Performance) of the LINPACK benchmark report.
  - N allowed to be any size
  - Timing harness provided to generate matrix, check the solution
  - List R_{max}, N_{max}, R_{peak}
- In 2000 we put together an optimized implementation of the benchmark, called High Performance LINPACK or HPL.
  - Just needs optimized version of BLAS and MPI.

TOP500

- In 1986 Hans Meuer started a list of supercomputer around the world, they were ranked by peak performance.
- Hans approached me in 1992 to merge our lists into the "TOP500".
- The first TOP500 list was in June 1993.

Rank	Site	System	Cores	Rmax (GFlop/s)	Rpeak (GFlop/s)	Power (kW)
1	Los Alamos National Laboratory United States	CM-5/1024 Thinking Machines Corporation	1,024	59.7	131.0	
2	Minnesota Supercomputer Center United States	CM-5/544 Thinking Machines Corporation	544	30.4	69.6	
3	National Security Agency United States	CM-5/512 Thinking Machines Corporation	512	30.4	65.5	
4	NCSA United States	CM-5/512 Thinking Machines Corporation	512	30.4	65.5	
5	NEC Japan	SX-3/44R NEC	4	23.2	25.6	
6	Atmospheric Environment Service (AES)	SX-3/44	4	20.0	22.0	

Rules For HPL and TOP500

- Have to compute the solution to a prescribed accuracy.
- Excludes the use of a fast matrix multiply algorithm like "Strassen's Method"
- Algorithms which compute a solution in a precision lower than full precision (64 bit floating point arithmetic) and refine the solution using an iterative approach.
- The authors of the TOP500 reserve the right to independently verify submitted LINPACK results, and exclude computer from the list which are not valid or not general purpose in nature.
- By general purpose computer we mean that the computer system must be able to be used to solve a range of scientific problems.
- Any computer designed specifically to solve the LINPACK benchmark problem or have as its major purpose the goal of a high TOP500 ranking will be disqualified.

High Performance LINPACK (HPL)

- Is a **widely recognized** and discussed metric for ranking high performance computing systems
- When HPL gained prominence as a performance metric in the early 1990s there **was a strong correlation between its predictions of system rankings and the ranking that full-scale applications would realize**.
- **Computer vendors pursued designs that would increase their HPL performance**, which would in turn improve overall application performance.
- Today HPL remains **valuable as a measure of historical trends**, and as a stress test, especially for leadership class systems that are pushing the boundaries of current technology.

HPL has a Number of Problems

- HPL performance of computer systems are **no longer so strongly correlated to real application performance**, especially for the broad set of HPC applications governed by partial differential equations.
- **Designing a system for good HPL performance can actually lead to design choices that are wrong** for the real application mix, or add unnecessary components or complexity to the system.

Concerns

- The **gap between HPL predictions and real application performance will increase** in the future.
- A computer system with the potential to run **HPL at an Exaflop is a design that may be very unattractive for real applications.**
- Future **architectures targeted toward good HPL performance will not be a good match for most applications.**
- This leads us to a think about a different metric

HPL - Good Things

- Easy to run
- Easy to understand
- Easy to check results
- Stresses certain parts of the system
- Historical database of performance information
- Good community outreach tool
- “Understandable” to the outside world

- “If your computer doesn’t perform well on the LINPACK Benchmark, you will probably be disappointed with the performance of your application on the computer.”

HPL - Bad Things

- LINPACK Benchmark is 36 years old
  - TOP500 (HPL) is 20.5 years old
- Floating point-intensive performs $O(n^3)$ floating point operations and moves $O(n^2)$ data.
- No longer so strongly correlated to real apps.
- Reports Peak Flops (although hybrid systems see only 1/2 to 2/3 of Peak)
- Encourages poor choices in architectural features
- Overall usability of a system is not measured
- Used as a marketing tool
- Decisions on acquisition made on one number
- Benchmarking for days wastes a valuable resource

Running HPL

- In the beginning to run HPL on the number 1 system was under an hour.
- On Livermore's Sequoia IBM BG/Q the HPL run took about a day to run.
  - They ran a size of n=12.7 x 10^6 (1.28 PB)
  - 16.3 PFlop/s requires about 23 hours to run!!
- The longest run was 60.5 hours
  - JAXA machine
    - Fujitsu FX1, Quadcore SPARC64 VII 2.52 GHz
  - A matrix of size n = 3.3 x 10^6
  - .11 Pflop/s #160 today

#1 System on the TOP500 Over the Past 20 Years (16 machines in that club)

9 6 2

TOP500 List	Computer	r_max (Tflop/s)	n_max	Hours	MW
6/93 (1)	TMC CM-5/1024	.060	52224	0.4	
11/93 (1)	Fujitsu Numerical Wind Tunnel	.124	31920	0.1	1.
6/94 (1)	Intel XP/S140	.143	55700	0.2	
11/94 - 11/95 (3)	Fujitsu Numerical Wind Tunnel	.170	42000	0.1	1.
6/96 (1)	Hitachi SR2201/1024	.220	138,240	2.2	
11/96 (1)	Hitachi CP-PACS/2048	.368	103,680	0.6	
6/97 - 6/00 (7)	Intel ASCI Red	2.38	362,880	3.7	.85
11/00 - 11/01 (3)	IBM ASCI White, SP Power3 375 MHz	7.23	518,096	3.6	
6/02 - 6/04 (5)	NEC Earth-Simulator	35.9	1,000,000	5.2	6.4
11/04 - 11/07 (7)	IBM BlueGene/L	478.	1,000,000	0.4	1.4
6/08 - 6/09 (3)	IBM Roadrunner – PowerXCell 8i 3.2 Ghz	1,105.	2,329,599	2.1	2.3
11/09 - 6/10 (2)	Cray Jaguar - XT5-HE 2.6 GHz	1,759.	5,474,272	17.3	6.9
11/10 (1)	NUDT Tianhe-1A, X5670 2.93Ghz NVIDIA	2,566.	3,600,000	3.4	4.0
6/11 - 11/11 (2)	Fujitsu K computer, SPARC64 VIIIfx	10,510.	11,870,208	29.5	9.9
6/12 (1)	IBM Sequoia BlueGene/Q	16,324.	12,681,215	23.1	7.9
11/12 (1)	Cray XK7 Titan AMD + NVIDIA Kepler	17,590.	4,423,680	0.9	8.2
6/13 - 11/13 (2)	NUDT Tianhe-2 Intel IvyBridge & Xeon Phi	33,862.	9,960,000	5.4	17.8

Ugly Things about HPL

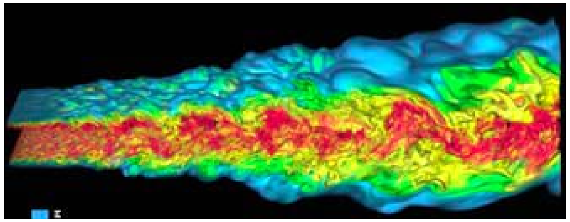
- Doesn’t probe the architecture; only one data point
- Constrains the technology and architecture options for HPC system designers.
  - Skews system design.
- Floating point benchmarks are not quite as valuable to some as data-intensive system measurements

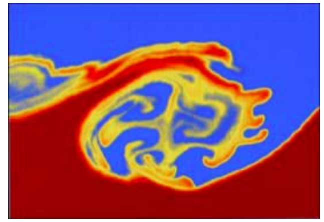
Many Other Benchmarks

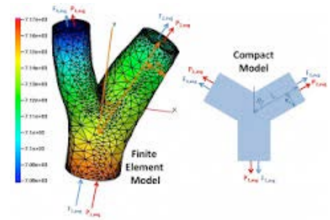
- TOP500
- Green 500
- Graph ~~500~~ 160
- Sustained Petascale Performance
- HPC Challenge
- Perfect
- ParkBench
- SPEC-hpc
- Big Data Top100
- Livermore Loops
- EuroBen
- NAS Parallel Benchmarks
- Genesis
- RAPS
- SHOC
- LAMMPS
- Dhrystone
- Whetstone
- I/O Benchmarks

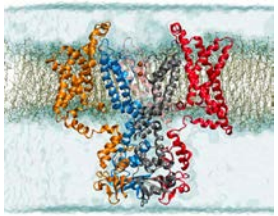
Goals for New Benchmark

- Augment the TOP500 listing with a benchmark that correlates with important scientific and technical apps not well represented by HPL









- Encourage vendors to focus on architecture features needed for high performance on those important scientific and technical apps.
  - Stress a balance of floating point and communication bandwidth and latency
  - Reward investment in high performance collective ops
  - Reward investment in high performance point-to-point messages of various sizes
  - Reward investment in local memory system performance
  - Reward investment in parallel runtimes that facilitate intra-node parallelism
- Provide an outreach/communication tool
  - Easy to understand
  - Easy to optimize
  - Easy to implement, run, and check results
- Provide a historical database of performance information
  - The new benchmark should have longevity

Proposal: HPCG

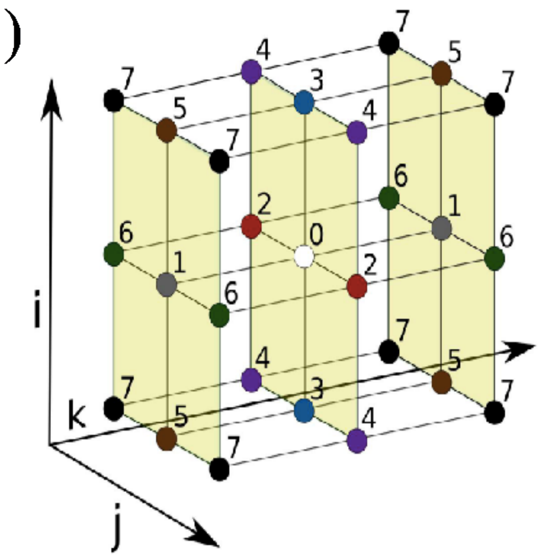
- High Performance Conjugate Gradient (HPCG).
- Solves $Ax=b$, A large, sparse, b known, x computed.
- An optimized implementation of PCG contains essential computational and communication patterns that are prevalent in a variety of methods for discretization and numerical solution of PDEs

- Patterns:
  - Dense and sparse computations.
  - Dense and sparse collective.
  - Data-driven parallelism (unstructured sparse triangular solves).

- Strong verification and validation properties

Model Problem Description

- Synthetic discretized 3D PDE (FEM, FVM, FDM).
- Single DOF heat diffusion model.
- Zero Dirichlet BCs, Synthetic RHS s.t. solution = 1.
- Local domain: $(n_x \times n_y \times n_z)$
- Process layout: $(np_x \times np_y \times np_z)$
- Global domain: $(n_x * np_x) \times (n_y * np_y) \times (n_z * np_z)$
- Sparse matrix:
  - 27 nonzeros/row interior.
  - 7 – 18 on boundary.
  - Symmetric positive definite.



27-point stencil operator

HPCG Design Philosophy

- Relevance to broad collection of important apps.
- Simple, single number.
- Few user-tunable parameters and algorithms:
  - The system, not benchmarker skill, should be primary factor in result.
  - Algorithmic tricks don’t give us relevant information.
- Algorithm (PCG) is vehicle for organizing:
  - Known set of kernels.
  - Core compute and data patterns.
  - Tunable over time (as was HPL).
- Easy-to-modify:
  - _ref kernels called by benchmark kernels.
  - User can easily replace with custom versions.
  - Clear policy: Only kernels with _ref versions can be modified.

PCG ALGORITHM

- $p_0 := x_0$, $r_0 := b\text{-}Ap_0$
- Loop i = 1, 2, …
  - o $z_i := M^{-1} r_{i\text{-}1}$
  - o if i = 1
    - ▪ $p_i := z_i$
    - ▪ $\alpha_i :=$ dot_product($r_{i\text{-}1}$, z)
  - o else
    - ▪ $\alpha_i :=$ dot_product($r_{i\text{-}1}$, z)
    - ▪ $\beta_i := \alpha_i / \alpha_{i\text{-}1}$
    - ▪ $p_i := \beta_i * p_{i\text{-}1} + z_i$
  - o end if
  - o $\alpha_i :=$ dot_product($r_{i\text{-}1}$, z_i) /dot_product(p_i, $A*p_i$)
  - o $x_{i+1} := x_i + \alpha_i * p_i$
  - o $r_i := r_{i\text{-}1} - \alpha_i * A * p_i$
  - o if $\|r_i\|_2$ < tolerance then Stop
- end Loop

Execution: 7 Phases

Problem Setup

- Construct Geometry.
- Generate Problem.
- Setup Halo Exchange.
- Initialize Sparse Meta-data.
- Call user-defined OptimizeProblem function. This function permits the user to change data structures and perform permutation that can improve execution.

Validation Testing

- Perform spectral properties PCG Tests:
  - Convergence for 10 distinct eigenvalues:
    - No preconditioning.
    - With Preconditioning
- Symmetry tests:
  - Sparse MV kernel.
  - MG kernel.

Reference Sparse MV and Gauss-Seidel kernel timing.

- Time calls to the reference versions of sparse MV and MG for inclusion in output report.

Reference CG timing and residual reduction.

- Time the execution of 50 iterations of the reference PCG implementation.
- Record reduction of residual using the reference implementation. The optimized code must attain the same residual reduction, even if more iterations are required.

Optimized CG Setup.

- Run one set of Optimized PCG solver to determine number of iterations required to reach residual reduction of reference PCG.
- Record iteration count as **numberOfOptCgIters.**
- Detect failure to converge.
- Compute how many sets of Optimized PCG Solver are required to fill benchmark timespan. Record as **numberOfCgSets**

Optimized CG timing and analysis.

- Run **numberOfCgSets** calls to optimized PCG solver with **numberOfOptCgIters** iterations.
- For each set, record residual norm.
- Record total time.
- Compute mean and variance of residual values.

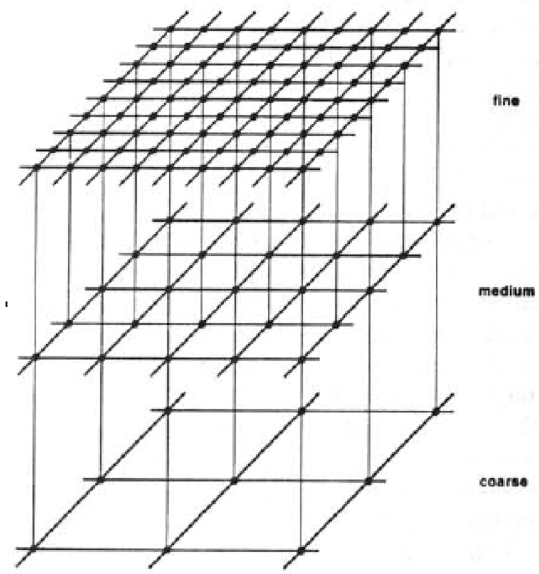
Report results

- Write a log file for diagnostics and debugging.
- Write a benchmark results file for reporting official information.

Preconditioner

- Hybrid geometric/algebraic multigrid:
  - Grid operators generated synthetically:
    - Coarsen by 2 in each x, y, z dimension (total of 8 reduction each level).
    - Use same GenerateProblem() function for all levels.
  - Grid transfer operators:
    - Simple injection. Crude but…
    - Requires no new functions, no repeat use of other functions.
    - Cheap.
  - Smoother:
    - Symmetric Gauss-Seidel [ComputeSymGS()].
    - Except, perform halo exchange prior to sweeps.
    - Number of pre/post sweeps is tuning parameter.
  - Bottom solve:
    - Right now just a single call to ComputeSymGS().

(In 2D, something like this)



- Symmetric Gauss-Seidel preconditioner
  - In Matlab that might look like:

```
LA = tril(A); UA = triu(A); DA = diag(diag(A));

x = LA\y;
x1 = y - LA*x + DA*x; % Subtract off extra
       diagonal contribution
x = UA\x1;
```

HPCG Parameters

- Iterations per set: 50.
- Total benchmark time for official result:
  - Repeated until 3600 seconds (1 hour run).
  - Anything less is reported as a “tuning” result.
- Coarsening: 2x – 2x – 2x (8x total).
- Number of levels:
  - 4 (including finest level).
  - Requires nx, ny, nz divisible by 8.
- Pre/post smoother sweeps: 1 each.

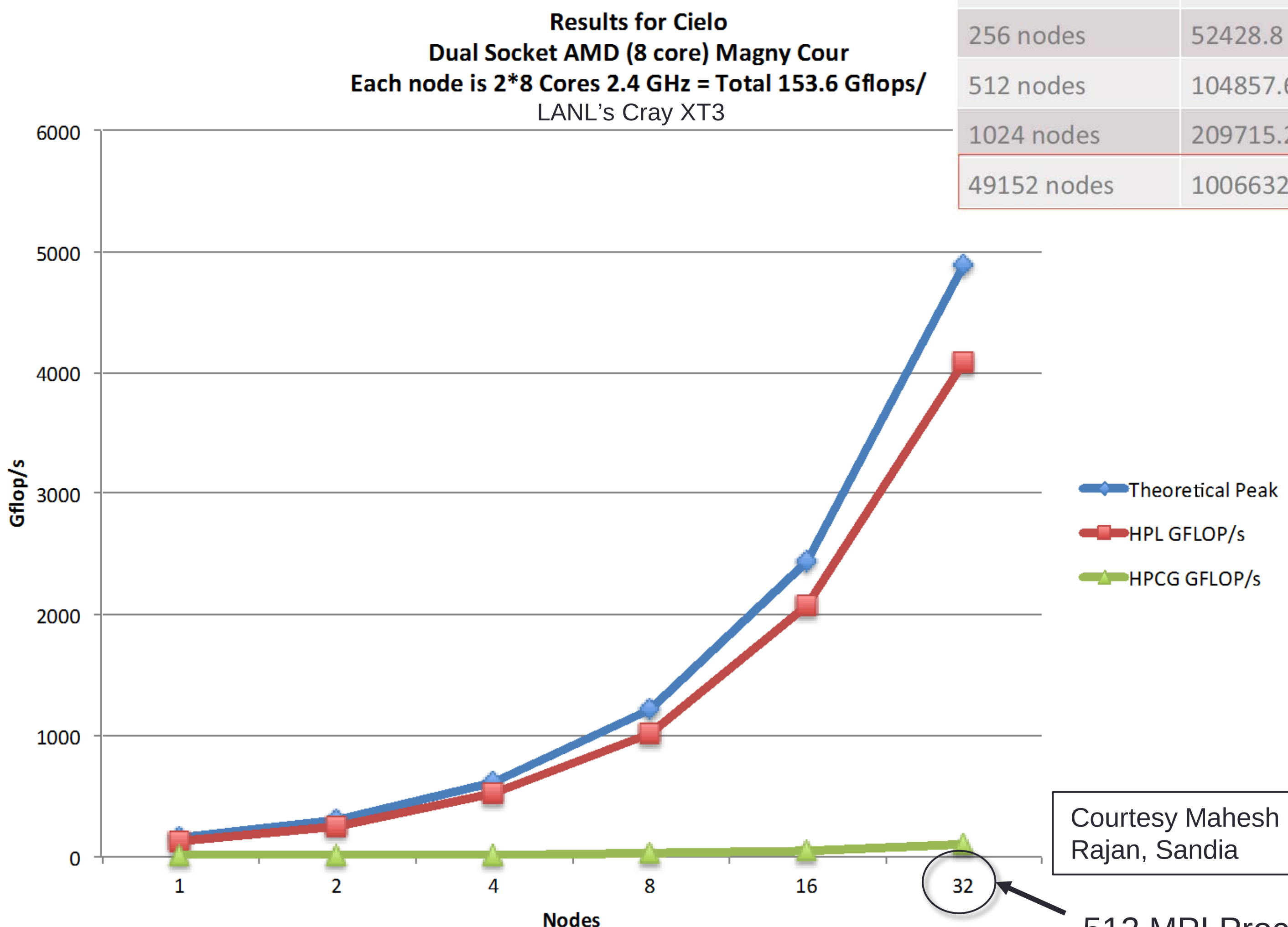
Merits of HPCG

- Includes major communication/computational patterns.
  - Represents a minimal collection of the major patterns.
- Rewards investment in:
  - High-performance collective ops.
  - Local memory system performance.
  - Low latency cooperative threading.
- Detects/measures variances from bitwise reproducibility.
- Executes kernels at several (tunable) granularities:
  - nx = ny = nz = 104 gives
  - nlocal = 1,124,864; 140,608; 17,576; 2,197
  - ComputeSymGS with multicoloring adds one more level:
    - 8 colors.
    - Average size of color = 275.
    - Size ratio (largest:smallest): 4096
  - Provide a “natural” incentive to run a big problem.

Performance “Shock”

Mira Partition Size ANL’s IBM BG/Q	Peak Gflops	Sustained Gflops	% of peak
64 nodes	13107.2	73.4	0.56%
128 nodes	26214.4	147.43	0.56%
256 nodes	52428.8	293.8	0.56%
512 nodes	104857.6	587.97	0.56%
1024 nodes	209715.2	1176.69	0.56%
49152 nodes	10066329.6	55177.6	0.55%

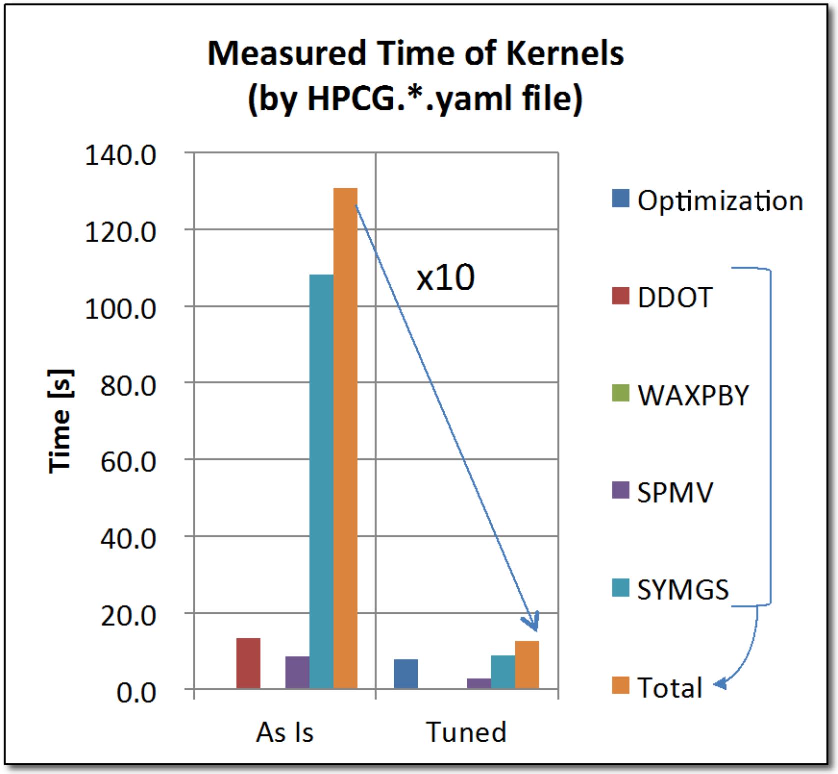
Courtesy Kalyan Kumaran, Argonne



Courtesy Mahesh Rajan, Sandia

512 MPI Processes

Tuning result on the K computer



Summary of "as is" code on the K

- Parallel scalability shouldn't be obstacle for large scale problem
- We are focusing on single CPU performance improvement

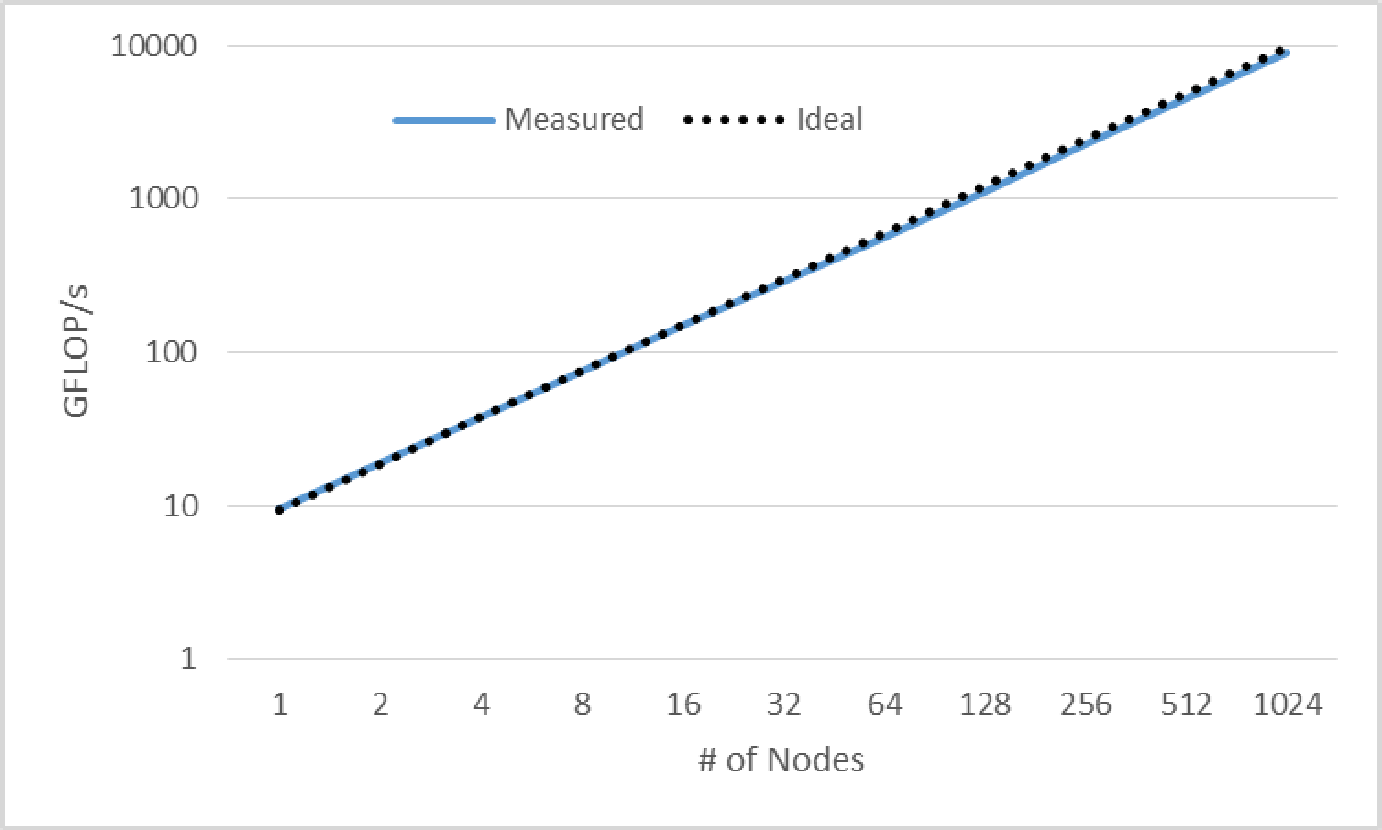
Improvement

- Total x10 speed up now
  - Continuous memory for matrix
  - Multi-coloring for SYMGS multi-threading
- Under Studying
  - Node re-ordering for SPMV
  - Advanced matrix storage way
  - And so on

8 Processes, 8 Threads/Process (Peak 128x8 GFLOPS)

Slide courtesy Naoya Maruyama, RIKEN AICS, and Fujitsu

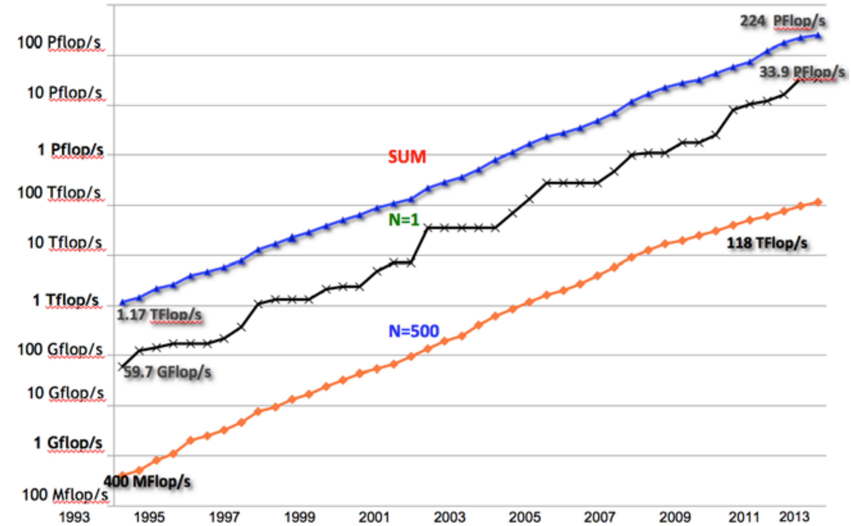
Multi-node Scaling



Stampede cluster, dual socket of 8-core SNB, 2.7 GHz
2 MPI processes per node (1 MPI process per skt. for NUMA)
160^3 input per MPI process
93% parallelization efficiency with 1024 nodes

HPCG and HPL

- We are NOT proposing to eliminate HPL as a metric.
- The historical importance and community outreach value is too important to abandon.
- HPCG will serve as an alternate ranking of the Top500.
  - Similar perhaps to the Green500 listing.



Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)	HPCG
1	National Super Computer Center in Guangzhou China	Tianhe-2 (MilkyWay-2) - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3,120,000	33,862.7	54,902.4	17,808	
2	DOE/SC/Oak Ridge National Laboratory United States	Titan - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209	
3	DOE/NNSA/LLNL United States	Sequoia - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1,572,864	17,173.2	20,132.7	7,890	
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705,024	10,510.0	11,280.4	12,660	
5	DOE/SC/Argonne National Laboratory United States	Mira - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786,432	8,586.6	10,066.3	3,945	
6	Swiss National Supercomputing Centre (CSCS) Switzerland	Piz Daint - Cray XC30, Xeon E5-2670 8C 2.600GHz, Aries interconnect , NVIDIA K20x Cray Inc.	115,984	6,271.0	7,788.9	2,325	
7	Texas Advanced Computing Center/Univ. of Texas United States	Stampede - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462,462	5,168.1	8,520.1	4,510	
8	Forschungszentrum Juelich (FZJ) Germany	JUQUEEN - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458,752	5,008.9	5,872.0	2,301	

Signs of Interest and Uptake

- Input from a various people at DOE Labs
- Discussions with and results from every HPC vendor.
  - Major, deep technical discussions with several.
- Same with most LCFs.
- Intel-sponsored SC’14 Workshop on Optimizing HPCG.

HPCG Tech Reports

Toward a New Metric for Ranking High Performance Computing Systems

- Jack Dongarra and Michael Heroux

HPCG Technical Specification

- Michael Heroux, Jack Dongarra, Piotr Luszczek

- http://tiny.cc/hpcg


SANDIA REPORT
SAND2013-8752
Unlimited Release
Printed October 2013

HPCG Technical Specification

Michael A. Heroux, Sandia National Laboratories[1]
Jack Dongarra and Piotr Luszczek, University of Tennessee

Prepared by

SANDIA REPORT
SAND2013-4744
Unlimited Release
Printed June 2013

Toward a New Metric for Ranking High Performance Computing Systems

Jack Dongarra, University of Tennessee
Michael A. Heroux, Sandia National Laboratories[1]

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Approved for public release; further dissemination unlimited.

Sandia National Laboratories

[1] Corresponding Author, maherou@sandia.gov