HPCG: ONE YEAR LATER

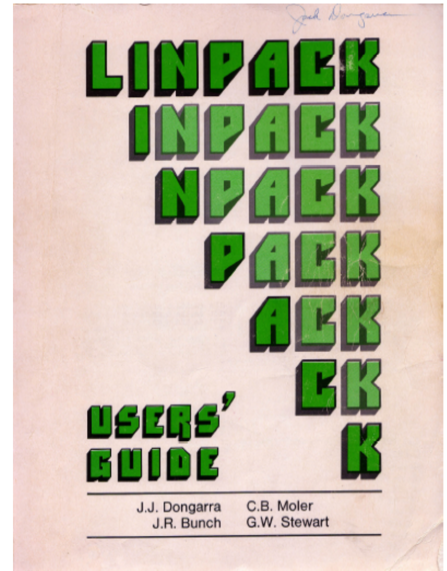
Jack Dongarra & Piotr Luszczek
University of Tennessee/ORNL

Michael Heroux
Sandia National Labs

http://tiny.cc/hpcg

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

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

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 put together 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	

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 37 years old
  - TOP500 (HPL) is 21.5 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

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

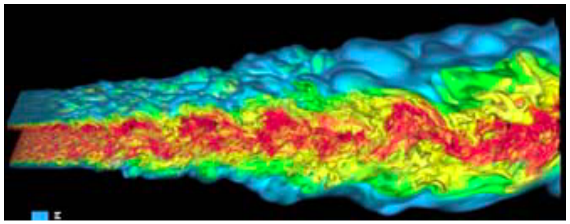
Many Other Benchmarks

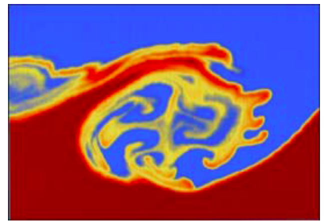
- TOP500
- Green 500
- Graph ~~500~~ 174
- Green/Graph
- Sustained Petascale Performance
- HPC Challenge
- Perfect
- ParkBench
- SPEC-hpc
- Livermore Loops
- EuroBen
- NAS Parallel Benchmarks
- Genesis
- RAPS
- SHOC
- LAMMPS
- Dhrystone
- Whetstone

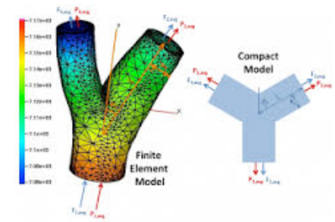
 http://tiny.cc/hpcg

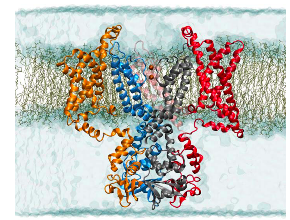
Goals for New Benchmark

- 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

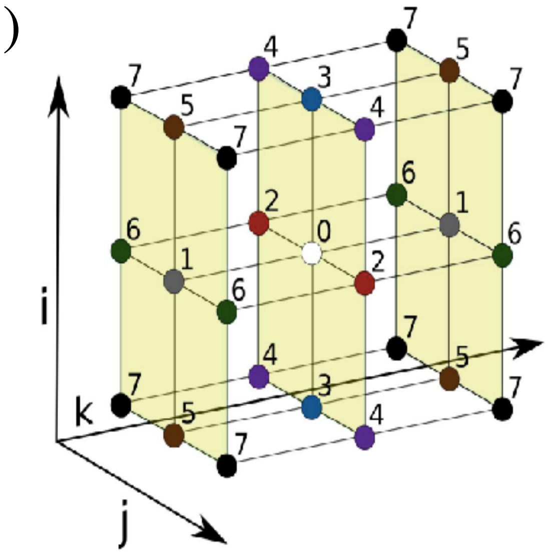
Proposal: HPCG

- 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.
  - Multi-scale execution of kernels via MG (truncated) V cycle.
  - Data-driven parallelism (unstructured sparse triangular solves).
- Strong verification and validation properties (via spectral properties of PCG).

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.

Example

- Build HPCG with default MPI and OpenMP modes enabled.

  export OMP_NUM_THREADS=1

  mpiexec –n 96 ./xhpcg 70 80 90
- Results in:

$$n_x = 70,\ n_y = 80,\ n_z = 90$$

$$np_x = 4,\ np_y = 4,\ np_z = 6$$

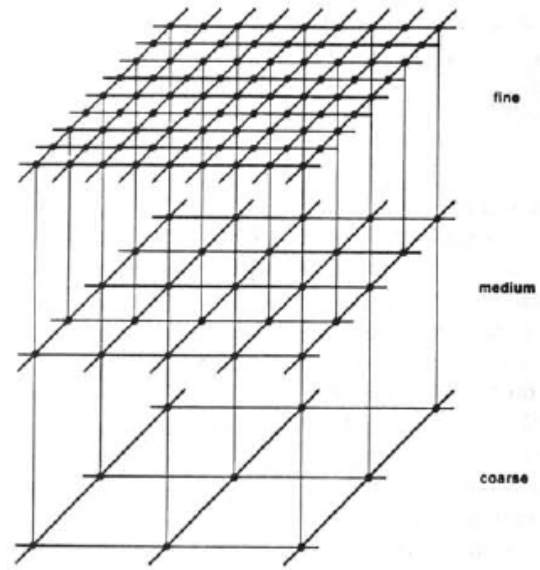
- Global domain dimensions: 280-by-320-by-540
- Number of equations per MPI process: 504,000
- Global number of equations: 48,384,000
- Global number of nonzeros: 1,298,936,872
- Note: Changing OMP_NUM_THREADS does not change any of these values.

PCG ALGORITHM

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

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().
    - If no coarse grids, has identical behavior as HPCG 1.X.



- 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;
```

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.

Example

- Reference PCG: 50 iterations, residual drop of 1e-6.
- Optimized PCG: Run one *set* of iterations
  - Multicolor ordering for Symmetric Gauss-Seidel:
    - Better vectorization, threading.
    - But: Takes 55 iterations to reach residual drop of 1e-6.
  - Overhead:
    - Extra 5 iterations.
    - Computing of multicolor ordering.
  - Compute number of sets we must run to fill entire execution time:
    - 5h/time-to-compute-1-set.
    - Results in thousands of CG set runs.
- Run and record residual for each set.
  - Report mean and variance (accounts for non-associativity of FP addition).

HPCG Parameters

- Iterations per set: 50.
- Total benchmark time for official result:
  - 3600 seconds.
  - Anything less is reported as a “tuning” result.
  - Default time 60 seconds.
- 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.
- Setup time: Amortized over 500 iterations.

Key Computation Data Patterns

- Domain decomposition:
  - SPMD (MPI): Across domains.
  - Thread/vector (OpenMP, compiler): Within domains.
- Vector ops:
  - AXPY: Simple streaming memory ops.
  - DOT/NRM2 : Blocking Collectives.
- Matrix ops:
  - SpMV: Classic sparse kernel (option to reformat).
  - Symmetric Gauss-Seidel: sparse triangular sweep.
    - Exposes real application tradeoffs:
      - threading & convergence vs. SPMD and scaling.
    - Enables leverage of new parallel patterns, e.g., futures.

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.

User tuning options

- MPI ranks vs. threads:
  - MPI-only: Strong algorithmic incentive to use.
  - MPI+X: Strong resource management incentive to use.

- Data structures:
  - Sparse and dense.
  - May not use knowledge of special sparse structure.
  - May not exploit regularity in data structures (x or y must be accessed indirectly when computing y = Ax).
  - Overhead of analysis/transformation is counted against time for ten 50 iteration sets (500 iterations).

User tuning options

- Permutations:
  - Can permute matrix for ComputeSpMV or ComputeMG or both.
  - Overhead is counted as with data structure transformations.
- Not permitted:
  - Algorithm changes to CG or MG that change behavior beyond permutations or FP arithmetic.
  - Change in FP precision.
  - Almost anything else not mentioned.

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.
  - Or maybe top 50 for now.

HPCG 3.X Features

- Truer C++ design:
  - Have gradually moved in that direction.
  - No one has complained.
- Request permutation vectors:
  - Permits explicit check again reference kernel results.
- Kernels will remain the same:
  - No disruption of vendor investments.

On Going Discussion and Feedback

- June 2013
  - Discussed at ISC
- November 2013
  - Discussed at SC13 in Denver during Top500 BoF
- January 2014
  - Discussed at DOE workshop
- March 2014
  - Discussed in DC at workshop
- June 2014
  - ISC talk at session

Signs of Uptake

- Discussions with and results from every vendor.
- Major, deep technical discussions with several.
- Same with most LCFs.
- SC’14 BOF on Optimizing HPCG.
- One ISC’14 and two SC’14 papers submitted.
  - Nvidia and Intel. 2/3 accepted.
- Optimized results for x86, MIC-based, Nvidia GPU-based systems.

HPL vs. HPCG: Bookends

- Some see HPL and HPCG as “bookends” of a spectrum.
  - Applications teams know where their codes lie on the spectrum.
  - Can gauge performance on a system using both HPL and HPCG numbers.
- Problem of HPL execution time still an issue:
  - Need a lower cost option. End-to-end HPL runs are too expensive.
  - Work in progress.

HPL
HPCG

Site	Computer	Cores	HPL Rmax (Pflops)	HPL Rank	HPCG (Pflops)
NSCC / Guangzhou	Tianhe-2 NUDT, Xeon 12C 2.2GHz + Intel Xeon Phi 57C + Custom	3,120,000	33.9	1	.580
RIKEN Advanced Inst for Comp Sci	K computer Fujitsu SPARC64 VIIIfx 8C + Custom	705,024	10.5	4	.427
DOE/OS Oak Ridge Nat Lab	Titan, Cray XK7 AMD 16C + Nvidia Kepler GPU 14C + Custom	560,640	17.6	2	.322
DOE/OS Argonne Nat Lab	Mira BlueGene/Q, Power BQC 16C 1.60GHz + Custom	786,432	8.59	5	.101#
Swiss CSCS	Piz Daint, Cray XC30, Xeon 8C + Nvidia Kepler 14C + Custom	115,984	6.27	6	.099
Leibniz Rechenzentrum	SuperMUC, Intel 8C + IB	147,456	2.90	12	.0833
CEA/TGCC-GENCI	Curie tine nodes Bullx B510 Intel Xeon 8C 2.7 GHz + IB	79,504	1.36	26	.0491
Exploration and Production Eni S.p.A.	HPC2, Intel Xeon 10C 2.8 GHz + Nvidia Kepler 14C + IB	62,640	3.00	11	.0489
DOE/OS L Berkeley Nat Lab	Edison Cray XC30, Intel Xeon 12C 2.4GHz + Custom	132,840	1.65	18	.0439 #
Texas Advanced Computing Center	Stampede, Dell Intel (8c) + Intel Xeon Phi (61c) + IB	78,848	.881*	7	.0161
Meteo France	Beaufix Bullx B710 Intel Xeon 12C 2.7 GHz + IB	24,192	.469 (.467*)	79	.0110
Meteo France	Prolix Bullx B710 Intel Xeon 2.7 GHz 12C + IB	23,760	.464 (.415*)	80	.00998
U of Toulouse	CALMIP Bullx DLC Intel Xeon 10C 2.8 GHz + IB	12,240	.255	184	.00725
Cambridge U	Wilkes, Intel Xeon 6C 2.6 GHz + Nvidia Kepler 14C + IB	3584	.240	201	.00385
TiTech	TUSBAME-KFC Intel Xeon 6C 2.1 GHz + IB	2720	.150	436	.00370

* scaled to reflect the same number of cores

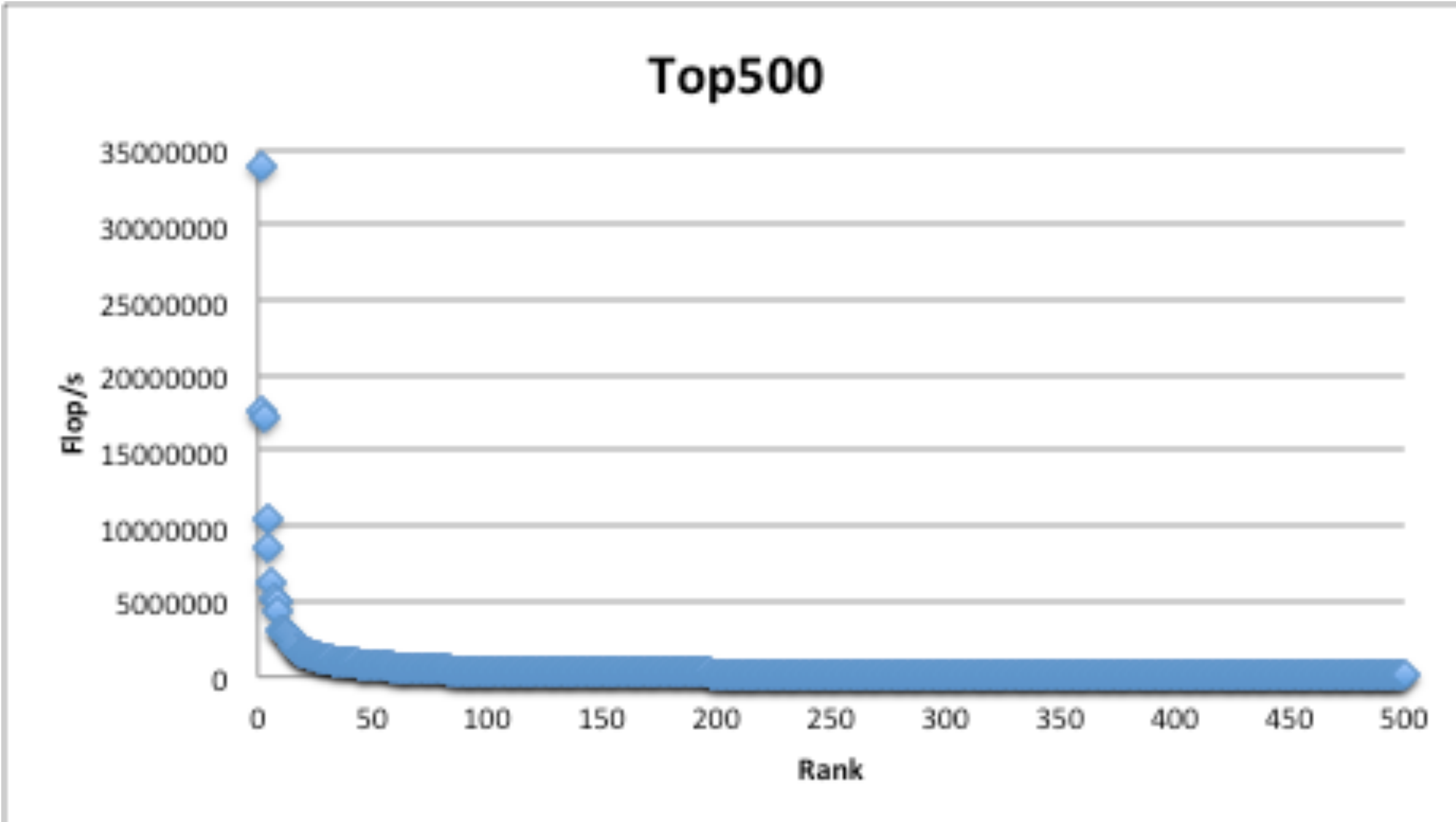
\# unoptimized implementation

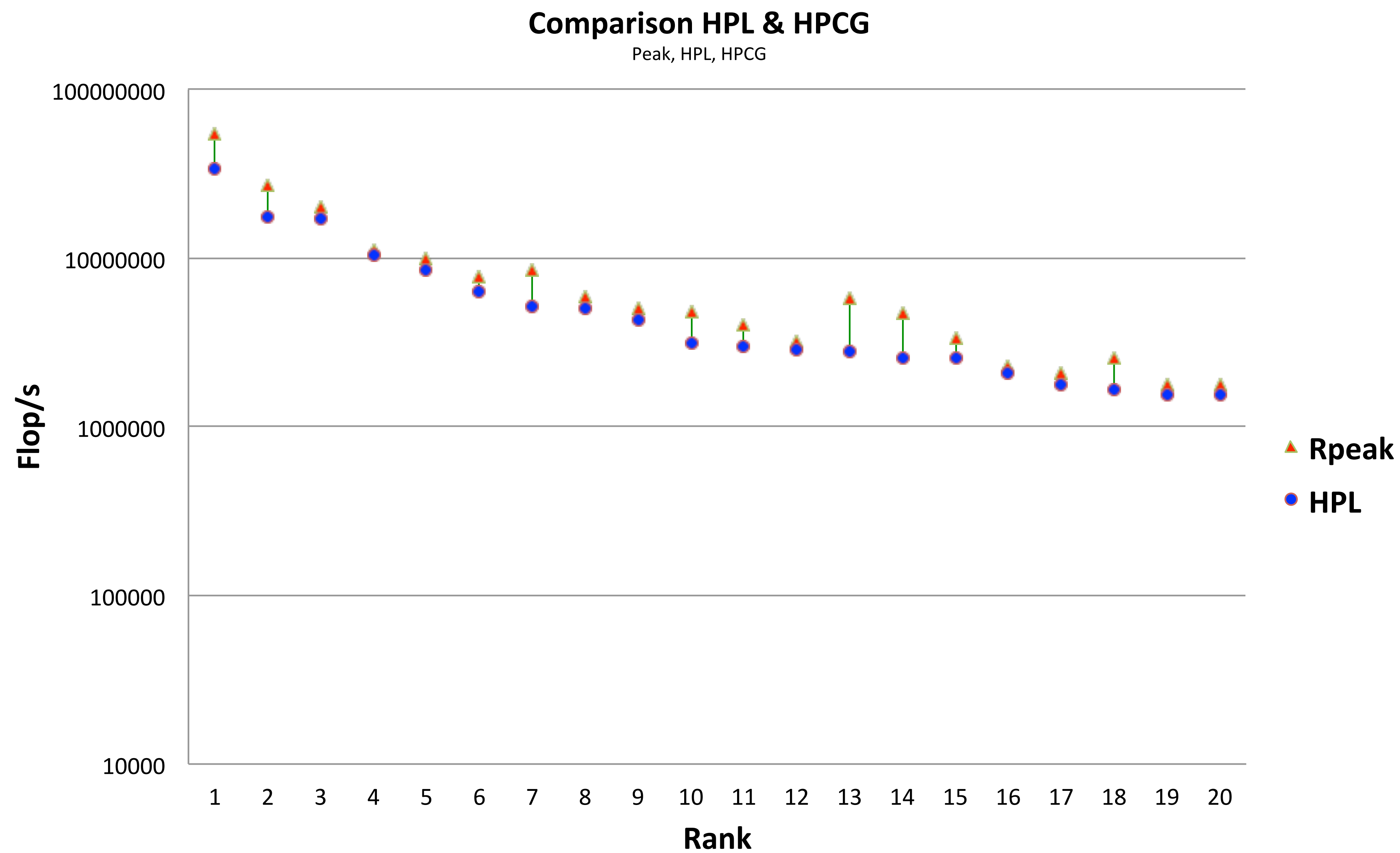
HPL
HPCG

Site	Computer	Cores	HPL Rmax (Pflops)	HPL Rank	HPCG (Pflops)	HPCG/ HPL
NSCC / Guangzhou	Tianhe-2 NUDT, Xeon 12C 2.2GHz + Intel Xeon Phi 57C + Custom	3,120,000	33.9	1	.580	1.7%
RIKEN Advanced Inst for Comp Sci	K computer Fujitsu SPARC64 VIIIfx 8C + Custom	705,024	10.5	4	.427	4.1%
DOE/OS Oak Ridge Nat Lab	Titan, Cray XK7 AMD 16C + Nvidia Kepler GPU 14C + Custom	560,640	17.6	2	.322	1.8%
DOE/OS Argonne Nat Lab	Mira BlueGene/Q, Power BQC 16C 1.60GHz + Custom	786,432	8.59	5	.101#	1.2%
Swiss CSCS	Piz Daint, Cray XC30, Xeon 8C + Nvidia Kepler 14C + Custom	115,984	6.27	6	.099	1.6%
Leibniz Rechenzentrum	SuperMUC, Intel 8C + IB	147,456	2.90	12	.0833	2.9%
CEA/TGCC-GENCI	Curie tine nodes Bullx B510 Intel Xeon 8C 2.7 GHz + IB	79,504	1.36	26	.0491	3.6%
Exploration and Production Eni S.p.A.	HPC2, Intel Xeon 10C 2.8 GHz + Nvidia Kepler 14C + IB	62,640	3.00	11	.0489	1.6%
DOE/OS L Berkeley Nat Lab	Edison Cray XC30, Intel Xeon 12C 2.4GHz + Custom	132,840	1.65	18	.0439 #	2.7%
Texas Advanced Computing Center	Stampede, Dell Intel (8c) + Intel Xeon Phi (61c) + IB	78,848	.881*	7	.0161	1.8%
Meteo France	Beaufix Bullx B710 Intel Xeon 12C 2.7 GHz + IB	24,192	.469 (.467*)	79	.0110	2.4%
Meteo France	Prolix Bullx B710 Intel Xeon 2.7 GHz 12C + IB	23,760	.464 (.415*)	80	.00998	2.4%
U of Toulouse	CALMIP Bullx DLC Intel Xeon 10C 2.8 GHz + IB	12,240	.255	184	.00725	2.8%
Cambridge U	Wilkes, Intel Xeon 6C 2.6 GHz + Nvidia Kepler 14C + IB	3584	.240	201	.00385	1.6%
TiTech	TUSBAME-KFC Intel Xeon 6C 2.1 GHz + IB	2720	.150	436	.00370	2.5%

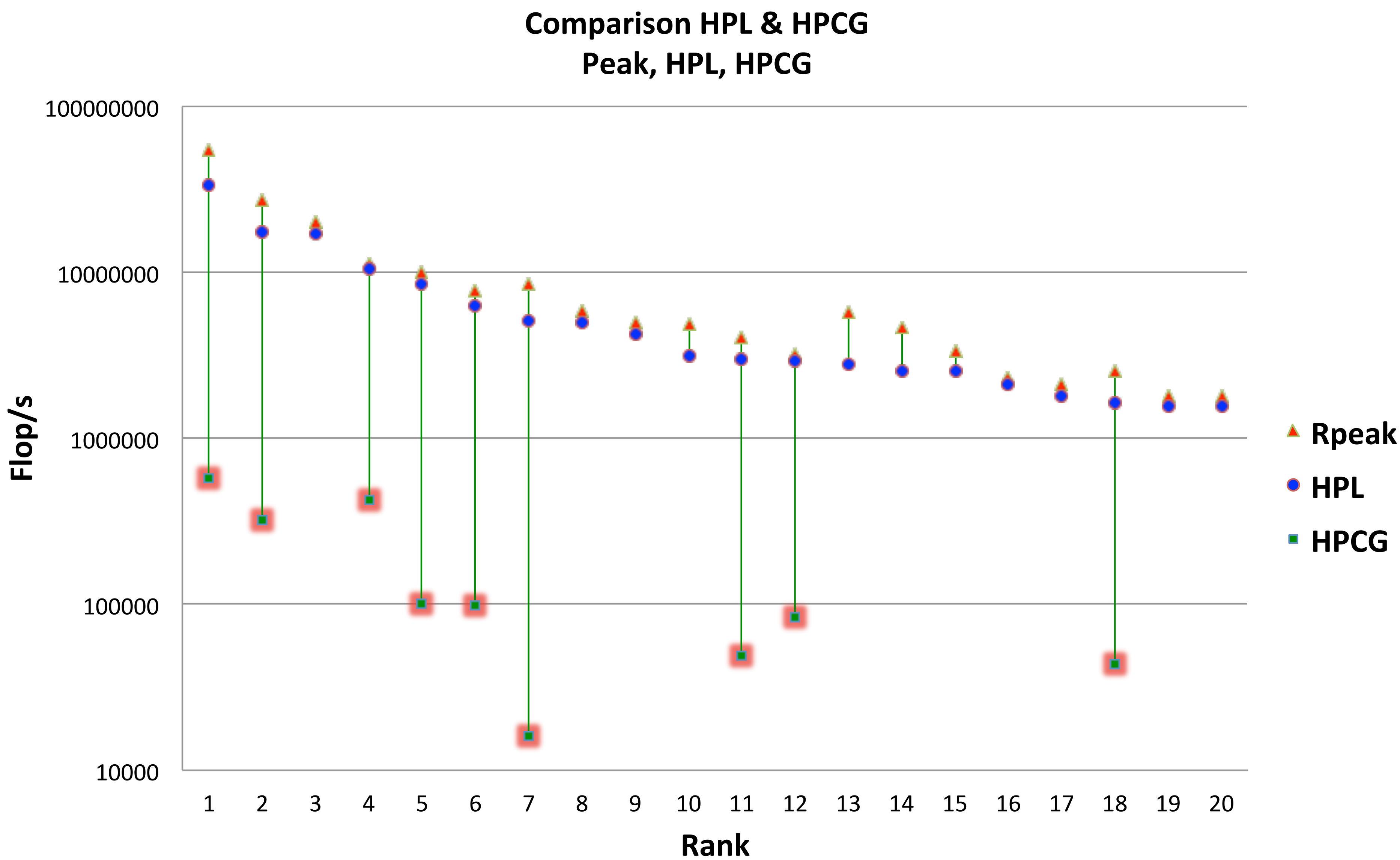
* scaled to reflect the same number of cores

unoptimized implementation


Top500
35000000
30000000
25000000
20000000
15000000
10000000
5000000
0
Flop/s
0
50
100
150
200
250
300
350
400
450
500
Rank



Comparison HPL & HPCG
Peak, HPL, HPCG
100000000
10000000
1000000
100000
10000
Flop/s
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Rank
Rpeak
HPL

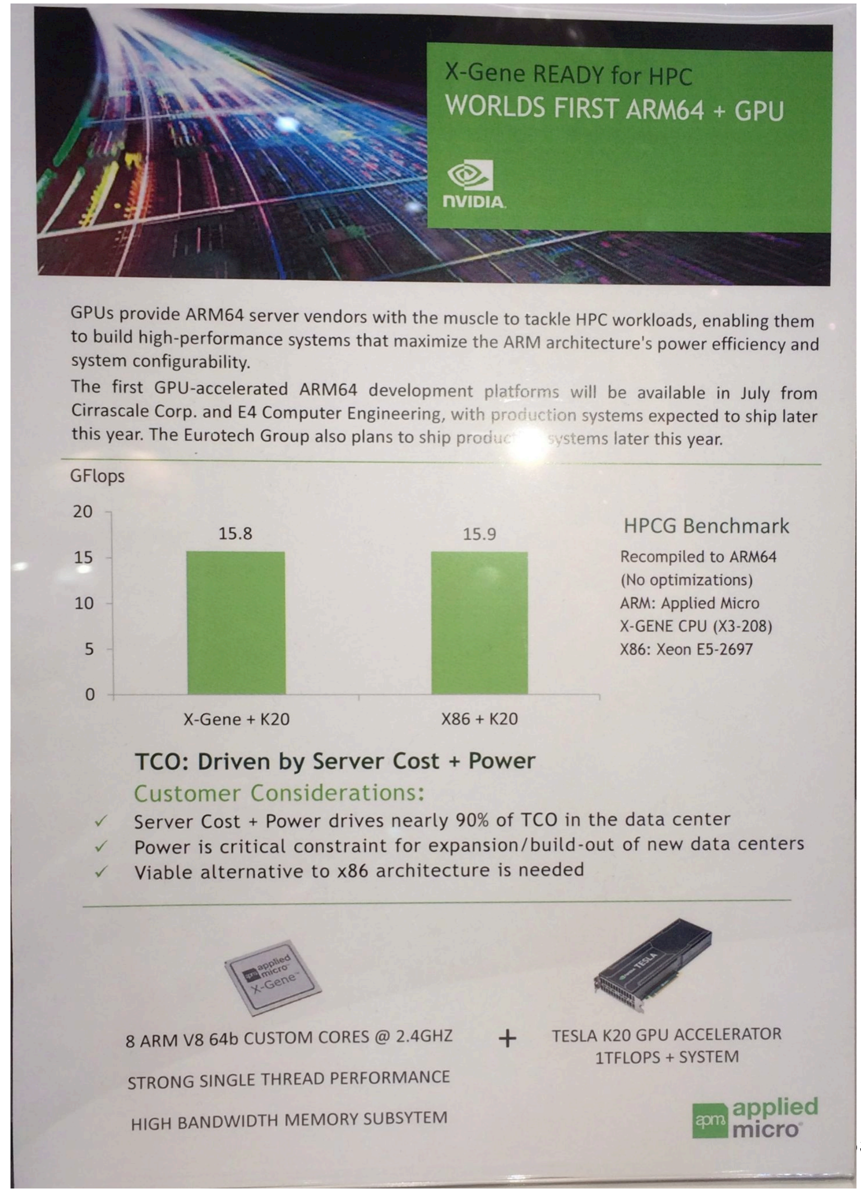

Comparison HPL & HPCG
Peak, HPL, HPCG
100000000
10000000
1000000
100000
10000
Flop/s
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
Rank
Rpeak
HPL
HPCG

Optimized Versions of HPCG

- Intel
  - MKL has packaged CPU version of HPCG
    - See: http://bit.ly/hpcg-intel
  - In the process of packaging Xeon Phi version to be released soon.
- Nvidia
  - Massimiliano Fatica and Evertt Phillips
  - Binary available
    - Contact Massimiliano mfatica@nvidia.com
- Bull
  - Developed by CEA requesting the release

07

Nvidia has it on their ARM64+K20



HPCG Tech Reports

Toward a New Metric for Ranking High Performance Computing Systems

- Jack Dongarra and Michael Heroux

HPCG Technical Specification

- Jack Dongarra, Michael Heroux, 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