Supercomputer Benchmarks A comparison of HPL, HPCG, and HPGMG and their Utility for the TOP500

Erich Strohmaier
HPCG: Jack Dongarra, Mike Heroux, Piotr Luszczek
HPGMG: Samuel Williams, Mark Adams

BERKELEY LAB
Lawrence Berkeley National Laboratory

WSSP Workshop, December 6, 2016

TOP500 Project

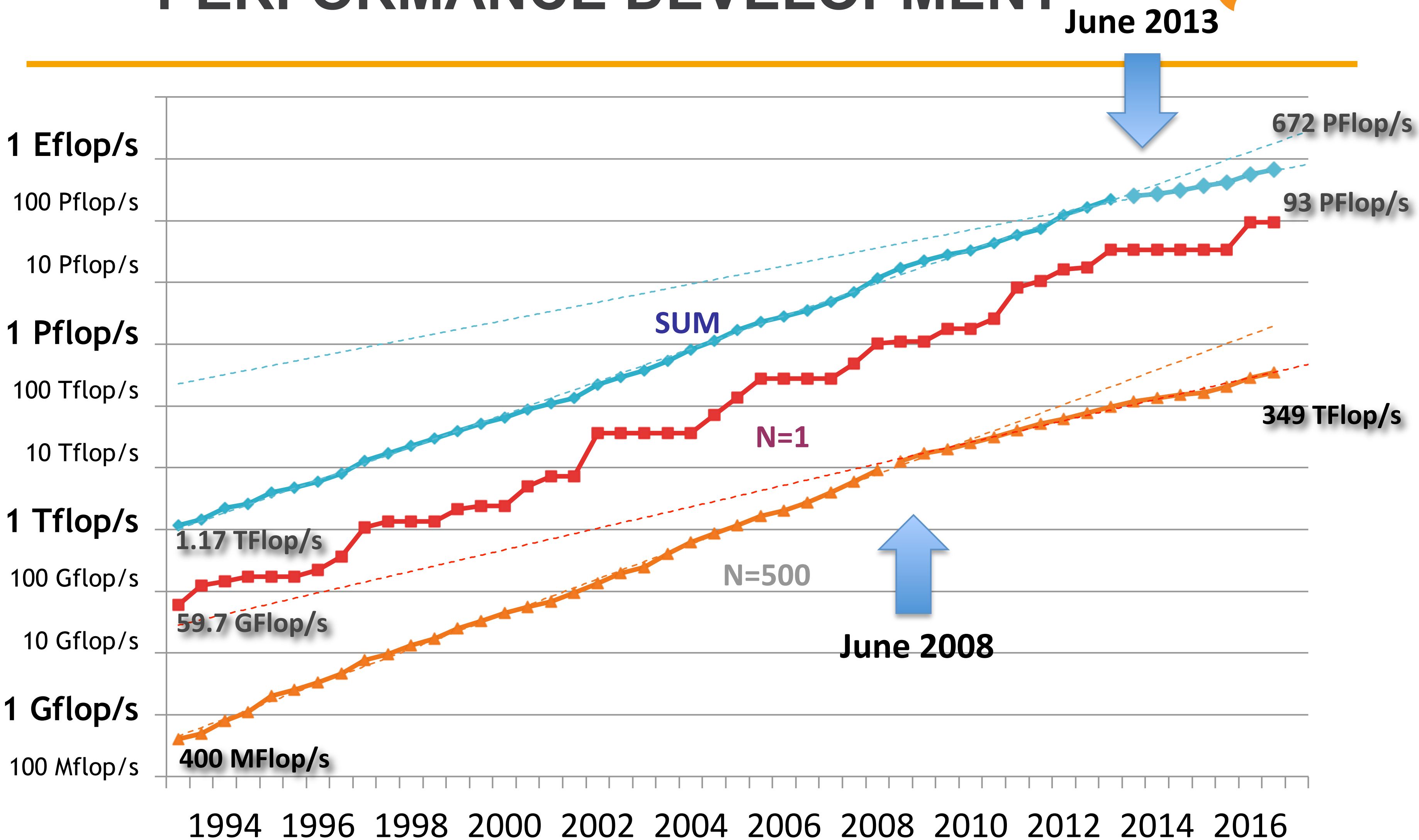
- Listing of the 500 most powerful computers in the world
- Yardstick: Rmax of Linpack
  - Solve Ax=b, dense problem, matrix is random
- Update twice a year since 1993:
  - ISC'xy in June in Germany • SCxy in November in the U.S.
- All information available from the TOP500 web site at: www.top500.org

TOP500 - Principles

- Adaptive definition of ‘Supercomputer’ for collecting market statistics
- Simple metric and procedure (few rules)
- Based on measured performance (system has to function)
- Floating point benchmark (‘scientific computing’ in early 90s)
- High performing (optimizable) to encourage adoption
- Broad system coverage
- HPL (High Performance Linpack) had widest coverage by a factor 2-3 x at least
  - In 1993 and still !
- But in benchmarking no benchmark serves all purposes and you need to know what you what to pick an appropriate benchmark!

#	Site	Manufact.	Computer	Country	Cores	Rmax [Pflops]	Power [MW]
1	National Supercomputing Center in Wuxi	NRCPC	Sunway TaihuLight NRCPC Sunway SW26010, 260C 1.45GHz	China	10,649,600	93.0	15.4
2	National University of Defense Technology	NUDT	Tianhe-2 NUDT TH-IVB-FEP, Xeon 12C 2.2GHz, IntelXeon Phi	China	3,120,000	33.9	17.8
3	Oak Ridge National Laboratory	Cray	Titan Cray XK7, Opteron 16C 2.2GHz, Gemini, NVIDIA K20x	USA	560,640	17.6	8.21
4	Lawrence Livermore National Laboratory	IBM	Sequoia BlueGene/Q, Power BQC 16C 1.6GHz, Custom	USA	1,572,864	17.2	7.89
5	Lawrence Berkeley National Laboratory	Cray	Cori Cray XC40, Intel Xeons Phi 7250 68C 1.4 GHz, Aries	USA	622,336	14.0	3.94
6	JCAHPC Joint Center for Advanced HPC	Fujitsu	Oakforest-PACS PRIMERGY CX1640 M1, Intel Xeons Phi 7250 68C 1.4 GHz, OmniPath	Japan	556,104	13.6	2.72
7	RIKEN Advanced Institute for Computational Science	Fujitsu	K Computer SPARC64 VIIIfx 2.0GHz, Tofu Interconnect	Japan	795,024	10.5	12.7
8	Swiss National Supercomputing Centre (CSCS)	Cray	Piz Daint Cray XC50, Xeon E5 12C 2.6GHz, Aries, NVIDIA Tesla P100	Switzer-land	206,720	9.78	1.31
9	Argonne National Laboratory	IBM	Mira BlueGene/Q, Power BQC 16C 1.6GHz, Custom	USA	786,432	8.59	3.95
10	Los Alamos NL / Sandia NL	Cray	Trinity Cray XC40, Xeon E5 16C 2.3GHz, Aries	USA	301,0564	8.10	4.23

PERFORMANCE DEVELOPMENT



TOP500 Main Criticism

- HPL is too floating point-intensive
  - It performs $O(n^3)$ floating point operations and moves $O(n^2)$ data (locally and globally) – and n grows historically !
    - Memory size ~ Moore's Law(Time) ~ a^Time
    - n ~ Sqrt(Memory) ~ Sqrt(a^Time)
    - **Byte/Flop ~ 12/n = O(1/n)**
    - **1979: 100^2; 1986: 1000^2 (1/10); TOP500: 1993: 5e4 (1/50); 2016 1e7 (1/200) : Total: 1/10^5 !**
- HPL does not representative our workloads and applications (any more) {but recently Deep Learning !?!)
- HPL sometimes produces rankings contrary to our intuition
- Too easy to build stunt machines:
  - Achieve high Linpack
  - Are not good for much else!

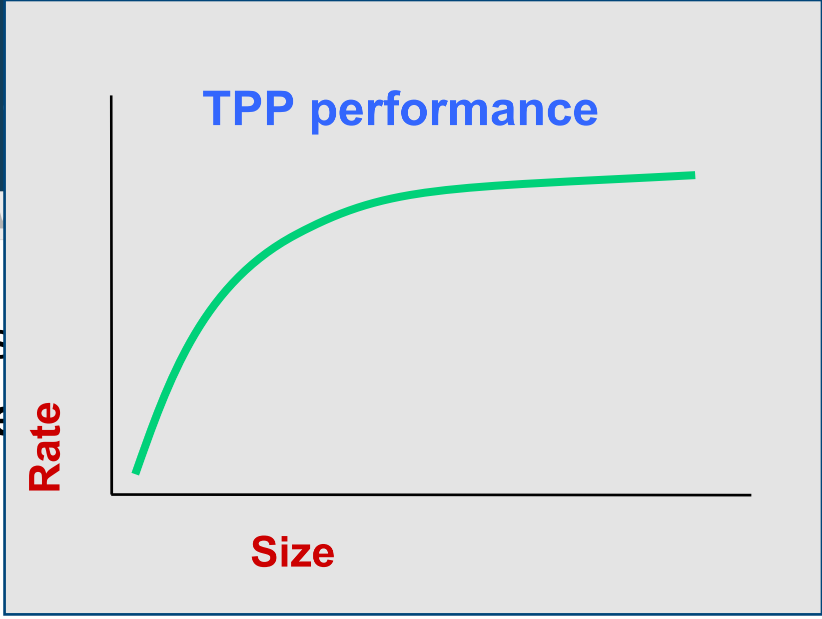
Why did Linpack Work so Well?

Many reasons, here are 3 essentials for the ***business as a list***:

1) Easy and continuous scalable problem size

- Otherwise you never keep up with Moore's Law
- You would loose comparability across discrete sizes
- It provides ONE simple performance number
- **Simplicity!**

Why did Linpack W



Many other reasons, here are 3 essentials

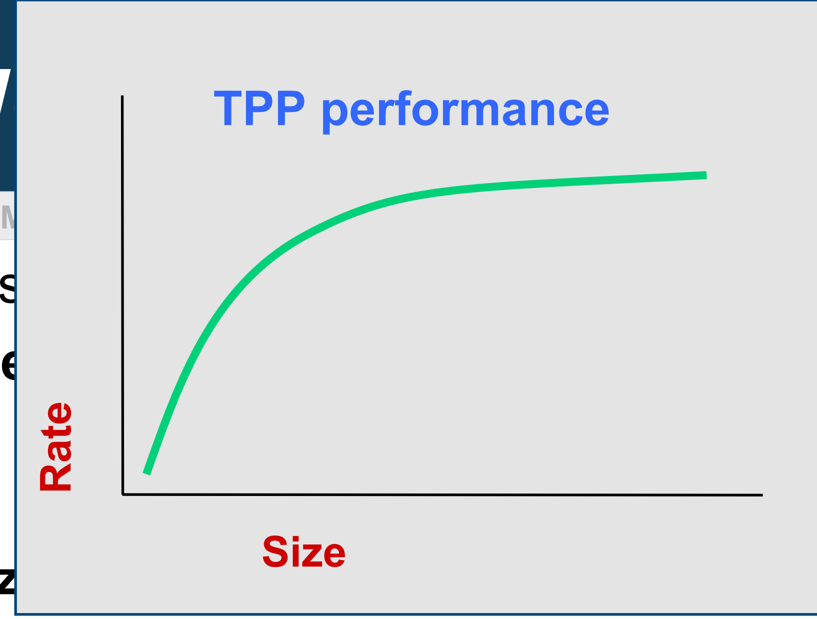
1) Easy and continuous scalable proble

- Simplicity

2) Asymptotically best performance

- For both **system size** and **problem size**
- This gets people to measure **full systems** and **fill up the memory** (no in-cache measurements)
- Preempts a boatload of bad tricks and games
- This also means your benchmark must scale aka cannot be too hard!!!
- **Brings out correct long term trends!**

Why did Linpack W



Many other reasons, here are 3 essentials

1) Easy and continuous scalable proble

- Simplicity

2) Asymptotically best performance

- For both **system size** and **problem siz**
- Brings out correct long term trends

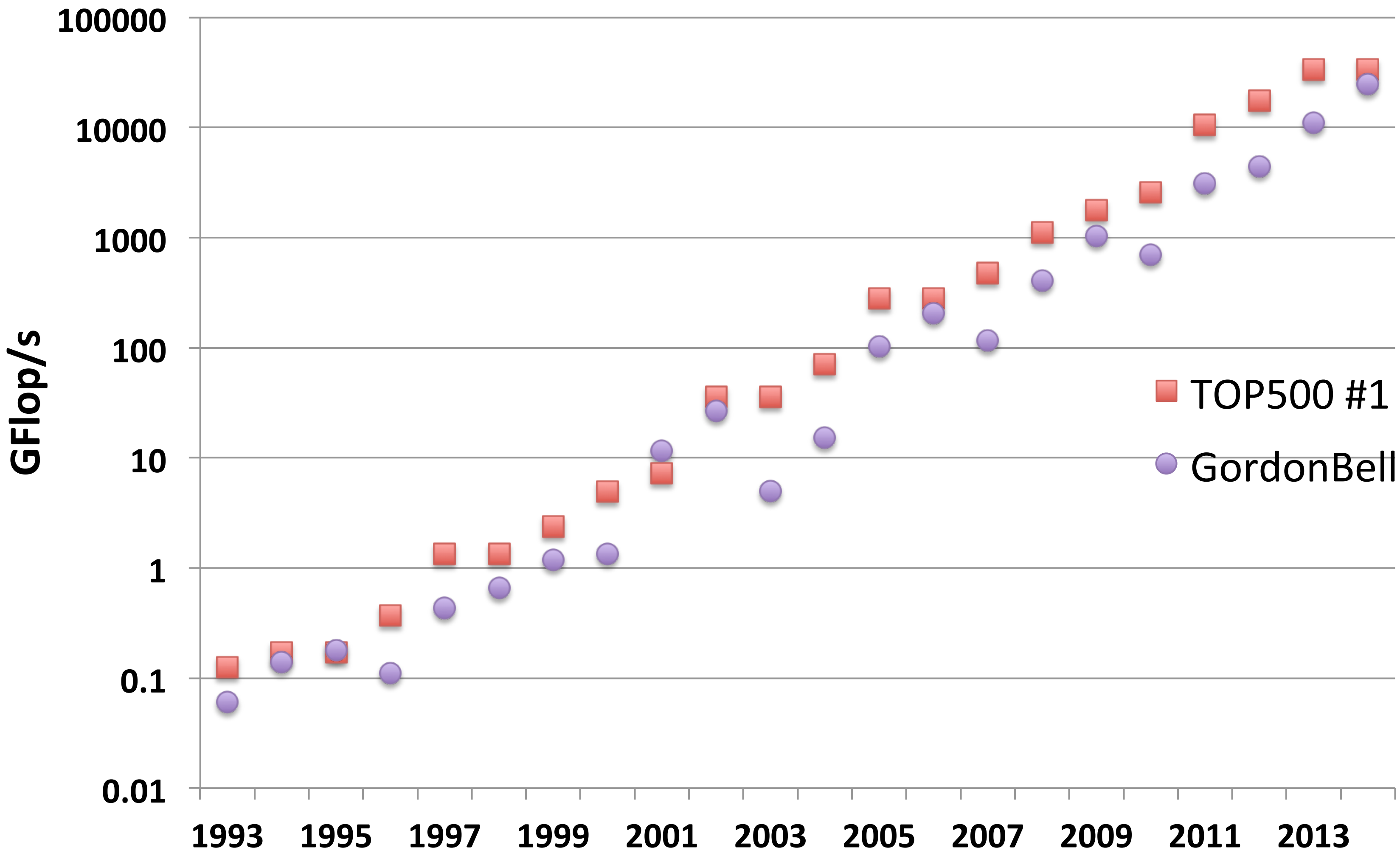
3) Convex performance curves over system size and problem size

- This allows a *safe* interpolation to smaller systems
- Important for coverage of large variety of installed system sizes
- This is probably a corollary to 2)
  (It looks like a more restricting requirement)

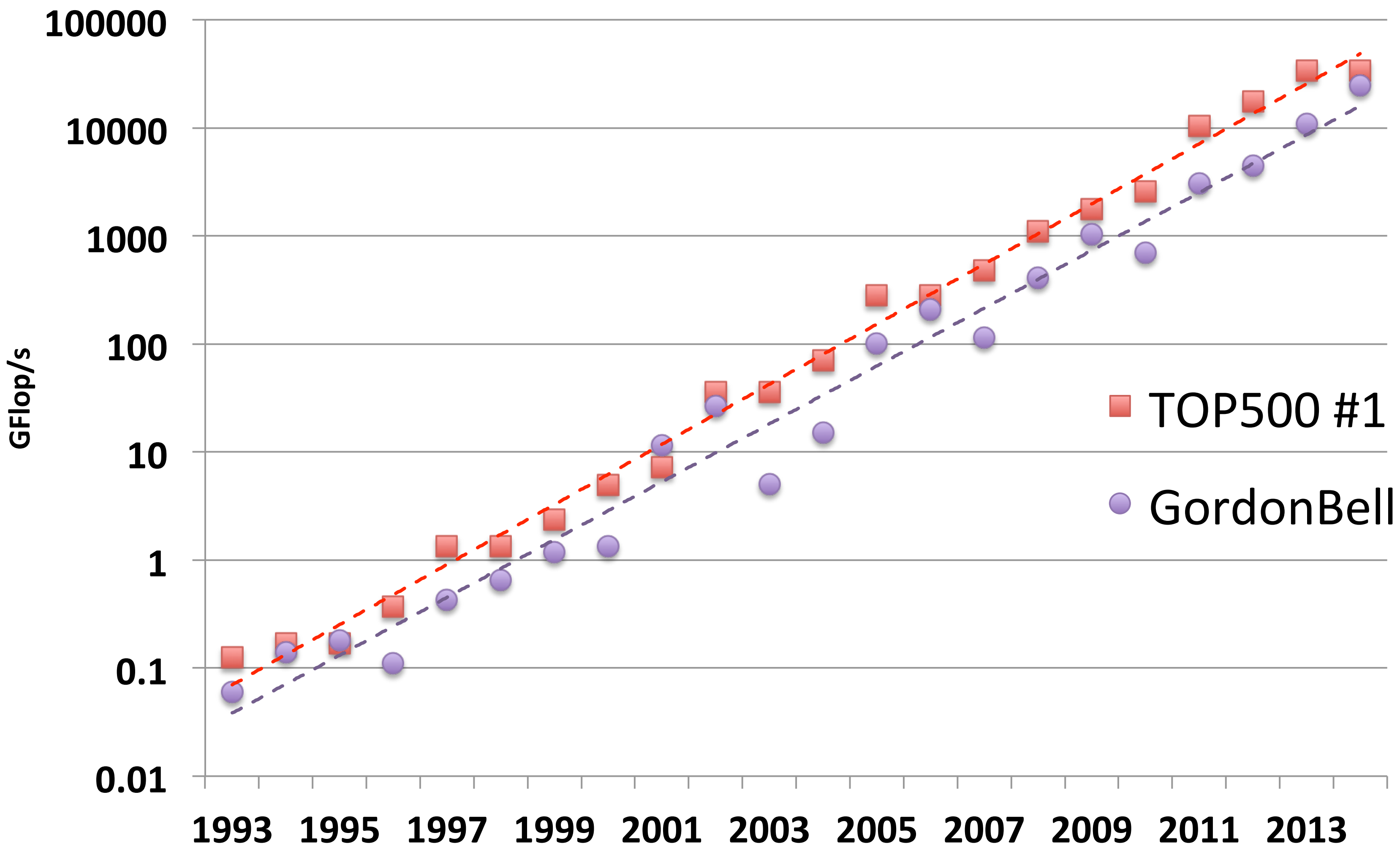
CORRELATION TO APPLICATION PERFORMANCE

- Example for ‘Application Performance’
  - Gordon Bell Awards
    - Handed out at SC since 1987
    - “Best” Application Performance Category
- Correlation with TOP500
  - Different Applications
  - Potentially different systems for GB and #1 TOP500

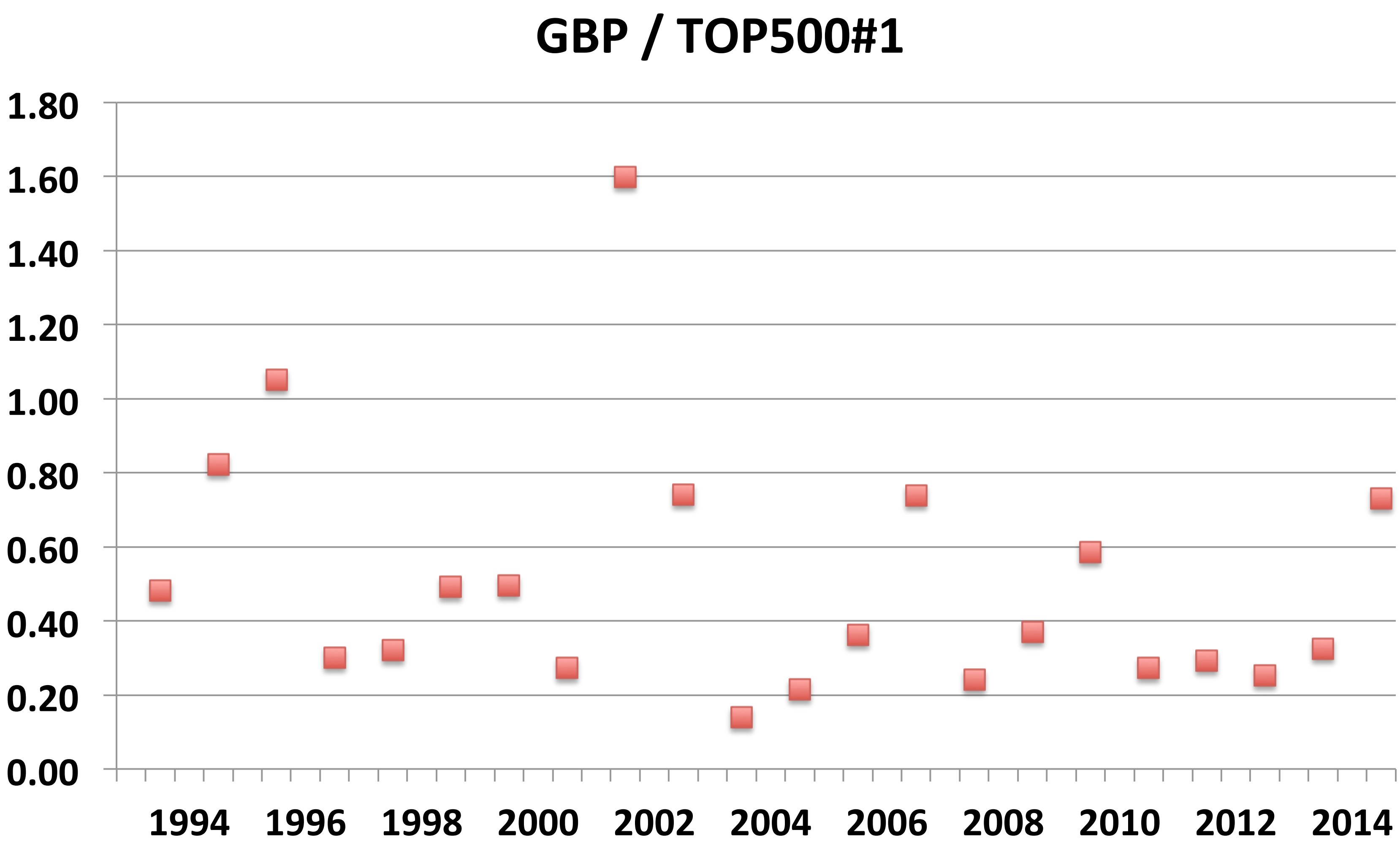
TOP500 VS GORDON BELL



TOP500 VS GORDON BELL



TOP500 VS GORDON BELL



Criteria for Additional Benchmarks

- **Represents a real computational problem**:
  - Allows simple problem scaling
  - Performs asymptotically optimal for system and problem size
    - Learned from HPL how important this is in practice
  - Main features all scale with O(n)
  - **Byte/Flop ~ O(1)**
- **Changes relative rankings compared to TOP500**:
  - New ordering should fit common sense (?)
  - **Should reorder by a sufficient magnitude !**
  - Otherwise the new benchmark is redundant
- **Makes it hard(er) to build stunt-machines**

Alternative Benchmarks

HPL is poorly correlated with the numerical methods used in applications today…

- $O(N^3)$ computational complexity
- O(N) arithmetic intensity (flop:byte)
- flop-limited (**measures peak flops**)
- **run times can exceed 24hrs**

Today's applications often use superior numerical methods which …

- have O(N) computational complexity
- have O(1) arithmetic intensity
- often DRAM or network limited
- have run times of O(10s)

- HPC Applications are increasingly built on scalable/hierarchical/ recursive algorithms…
- Use an algorithm that is understandable by CS grad students
- Runs on any scale machine (single core to exascale)
- Specify the algorithm (math), but leave the implementation free
- Reward systems that are tightly integrated

(Past) Alternative Benchmarks

- **NAS Parallel Benchmarks / NPB (1991)**
  - 8 benchmarks including CG (stored matrix), MG (ccPoisson), FFT, …
  - strong scaled with a few *classes* of problem sizes
- **HPCC (2005)**
  - multi-component, weak scaled benchmarks
  - STREAM: overly simple DRAM bandwidth kernel
  - GUPS: Random access kernel; atypical of most HPC applications
  - HPL: LINPACK; peak flop/s; atypical of most HPC applications
  - FFT: common method for small-scale HPC and simple problems (e.g. constant coefficient Poisson with periodic BC's)
- **Graph500**
  - BFS on graphs
  - little/no FP (targets a different domain)
  - specified problem sizes (scale problems)

High Performance Conjugate Gradient HPCG (V3.0)

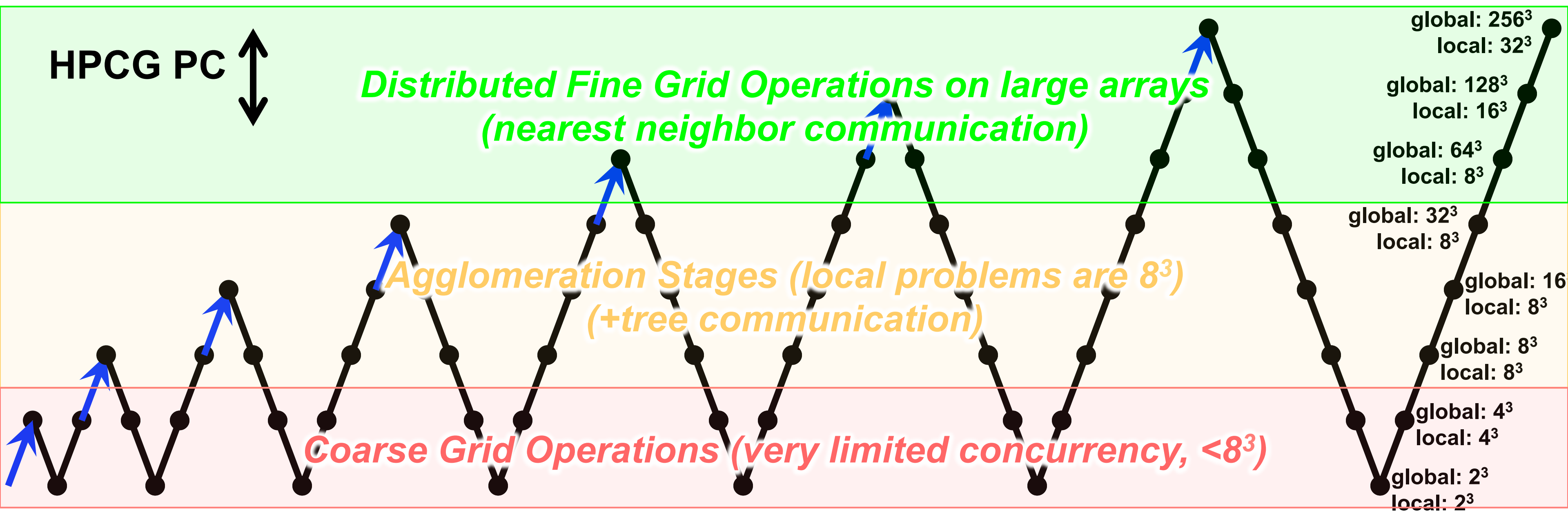
- Solves *Ax=b, A* large, **sparse**, *b* known, *x* computed.
- Originally local and symmetric Gauss-Seidel preconditioner, then 2 level MG PC, now 4 level
  - Try to avoid Stream like behavior
  - **Byte/Flop >~ 4**; O(1) computation; O(n^2/3) global communication
- A multigrid preconditioned CG (PCG) exercises a variety of computational and communication patterns on a nested set of coarse grids
  - Sparse matrix-vector multiplication
  - Sparse triangle solve
  - Vector updates
  - Global dot products
  - Local symmetric Gauss-Seidel smoother
- Has gained some traction (61 systems in TOP500 measured)

HPGMG Specification

- Geometric Multigrid
  - Multigrid on a structured Cartesian grid **= understandable**
  - GMG is found at the heart of many DOE applications including AMR/MG frameworks like CHOMBO and BoxLib **= relevant**
- HPGMG (High Performance Geometric Multigrid)
  - Solves variable coefficient Poisson $b\nabla \cdot \beta \nabla u = f$ on the $[0,1]^3$
  - Cubical Cartesian grid with Dirichlet BC's
  - Uses asymptotically exact Full Multigrid (**FMG**) which is a one-pass direct solver built from a hierarchy of MG V-Cycles.
  - Fully specified stencils and smoothers
- Three variants of HPGMG have been evaluated:
  - HPGMG-FV (Finite Volume, 2nd order, memory intensive)
  - **HPGMG-FV (Finite Volume, 4th order, memory/cache intensive)**
    - **Byte/Flop ~ 1;** O(1) computation; O(n^2/3) global communication
  - HPGMG-FE (Finite Element, 3rd order, cache/floating-point intensive)
- Reference Implementations on https://bitbucket.org/hpgmg/hpgmg

HPGMG has Multiple Communication Patterns

- Work is redistributed onto fewer cores (agglomeration)
- Coarse grid solves can occur on a single core of a single node
- Coarse grid solution is propagated to every thread in the system



HPCG – November 2016 Top10

Rnk	Machine	Cores	HPL Res	HPL Rnk	HPCG PF/s	% Peak
1	**K computer**	705,024	10.510	7	0.6027	5.3%
2	**Tianhe-2**	3,120,000	33.863	2	0.5800	1.1%
3	**Oakforest**	557,056	13.555	6	0.3855	1.5%
4	**TaihuLight**	10,649,600	93.015	1	0.3712	0.3%
5	**Cori**	632,400	13.832	5	0.3554	1.3%
6	**Sequoia**	1,572,864	17.173	4	0.3304	1.6%
7	**Titan**	560,640	17.590	3	0.3223	1.2%
8	**Trinity**	301,056	8.101	10	0.1826	1.6%
9	**Pleiades**	243,008	5.952	13 *	0.1752	2.5%
10	**Mira**	786,432	8.587	9	0.1670	1.7%

* #11 and #12 have no HPCG numbers

http://hpcg-benchmark.org

Complete list on hpcg-benchmark.org:
http://www.hpcg-benchmark.org/custom/index.html?lid=155&slid=289

HPGMG - November 2016 Ranking

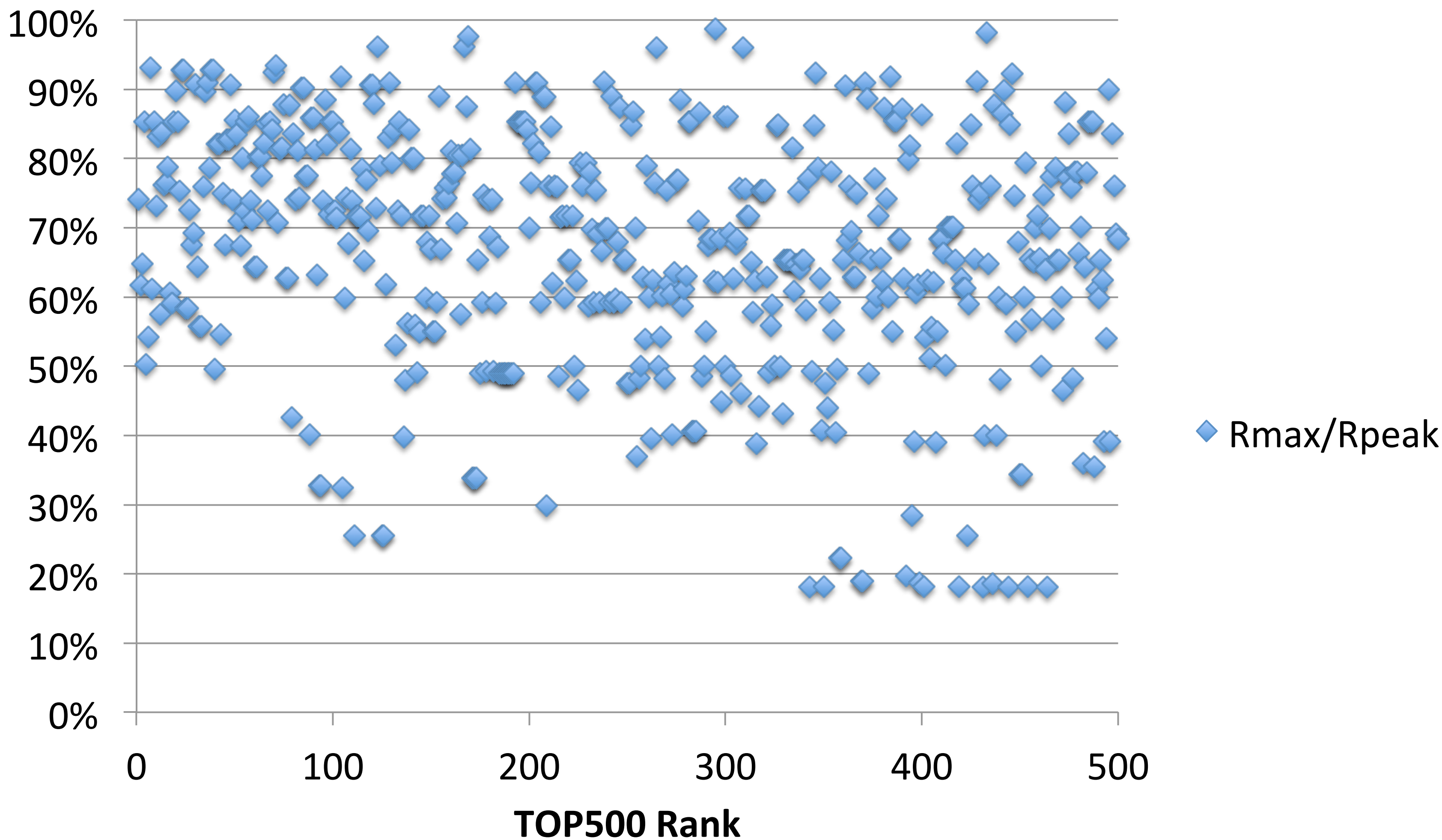
HPGMG Rank	System Site	System Name	10^9 DOF/s	MPI	OMP	Acc	DOF per Process	Top500 Rank	Notes
1	ALCF	Mira	500	49152	64	0	36M	9	BGQ
2	HLRS	Hazel Hen	495	15408	12	0	192M	14	
3	OLCF	Titan	440	16384	4	1	32M	3	K20x GPU
4	KAUST	Shaheen II	326	12288	16	0	144M	15	
5	NERSC	Edison	296	10648	12	0	128M	60	
6	CSCS	Piz Daint	153	4096	8	1	32M	8*	K20x GPU
7	Tohoku University	SX-ACE	73.8	4096	1	0	128M	-	vector
8	LRZ	SuperMUC	72.5	4096	8	0	54M	36	
9	NREL	Peregrine	10.0	1024	12	0	16M	-	
10	NREL	Peregrine	5.29	512	12	0	16M	-	
11	HLRS	SX-ACE	3.24	256	1	0	32M	-	vector
12	NERSC	Babbage	0.762	256	45	0	8M	-	KNC

DOF/s * 1200 ~ Flop/s

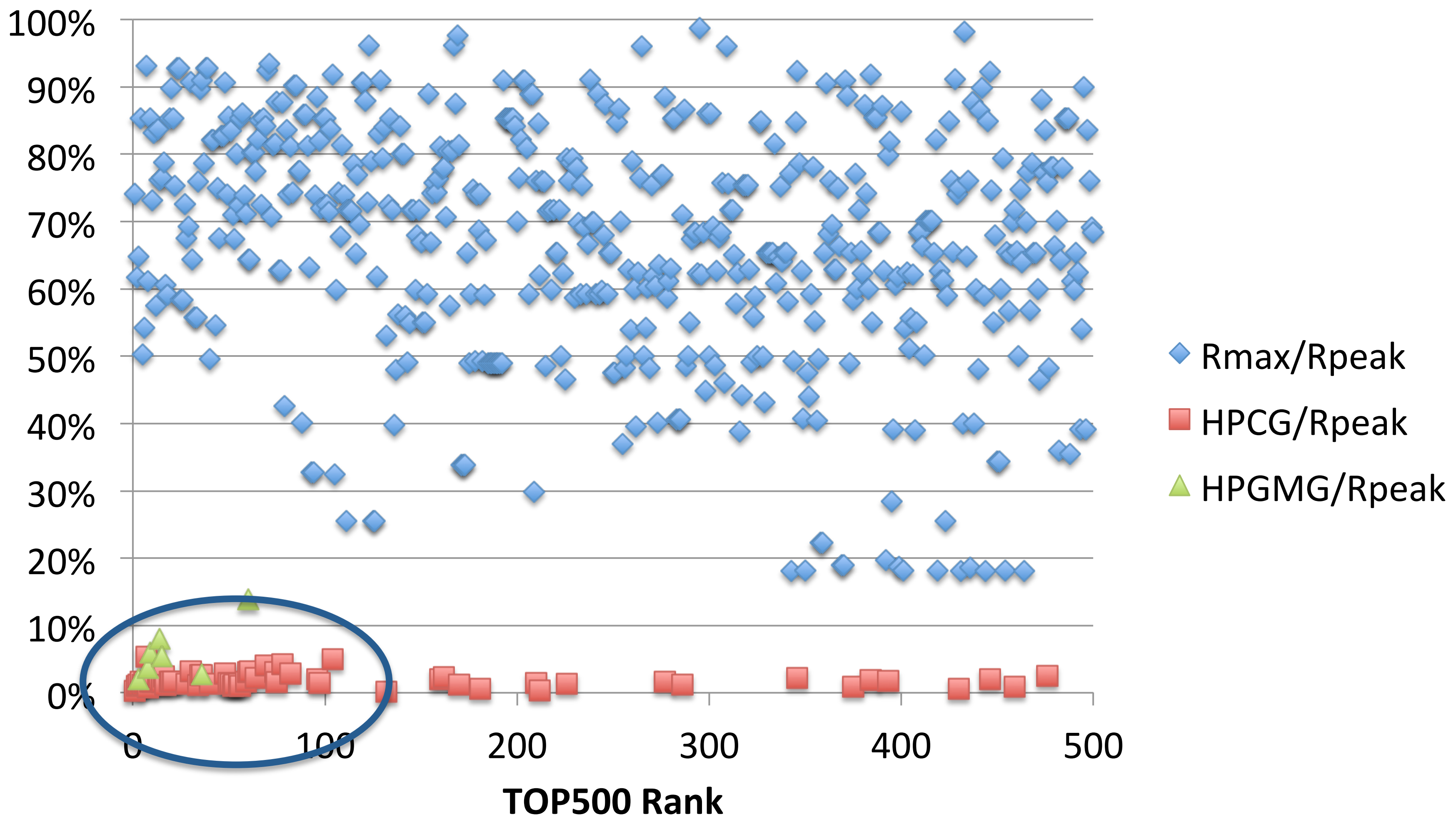
* Measured prior to upgrade?
Would Be #12 now

BERKELEY LAB

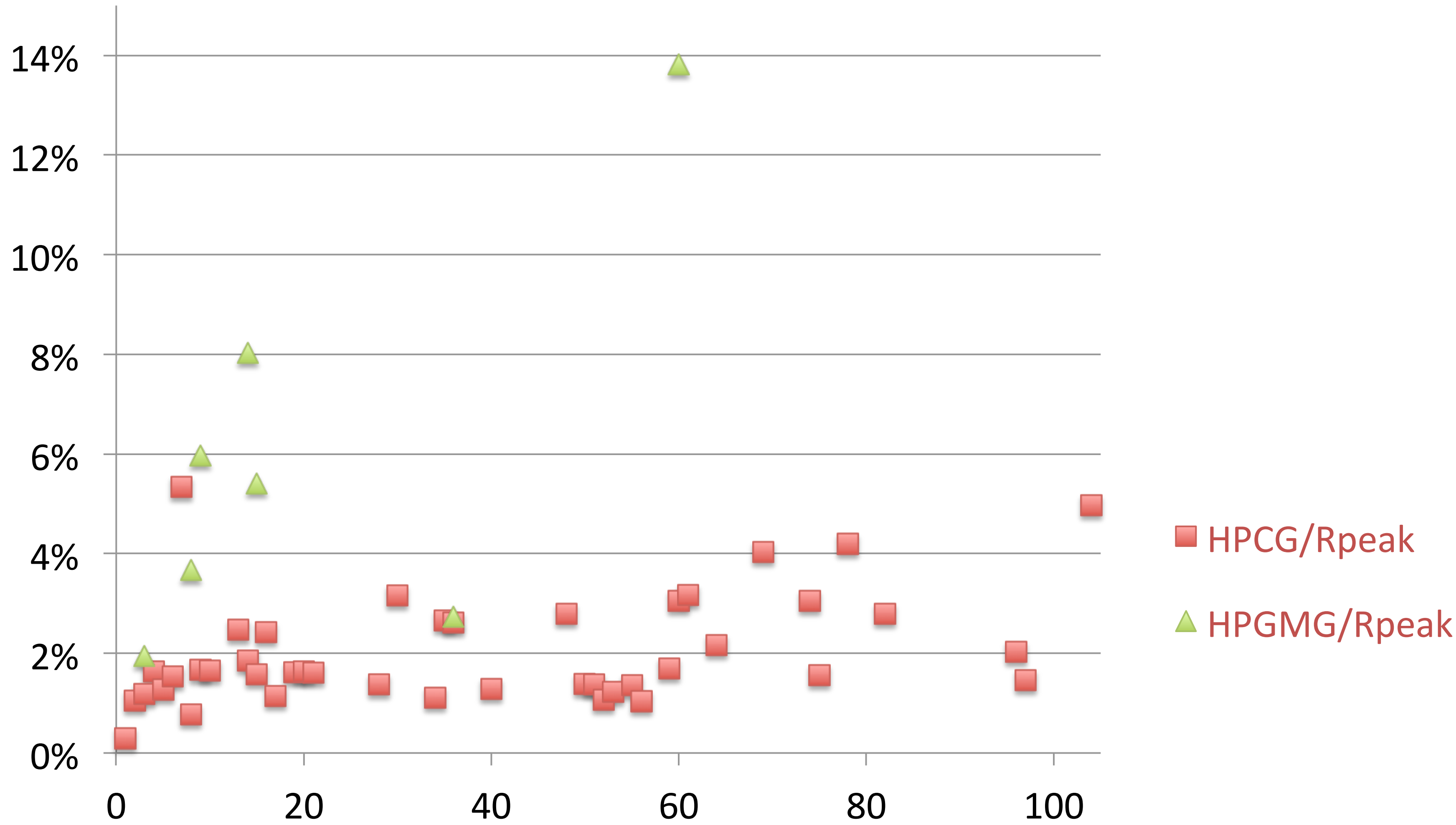
HPL EFFICIENCY



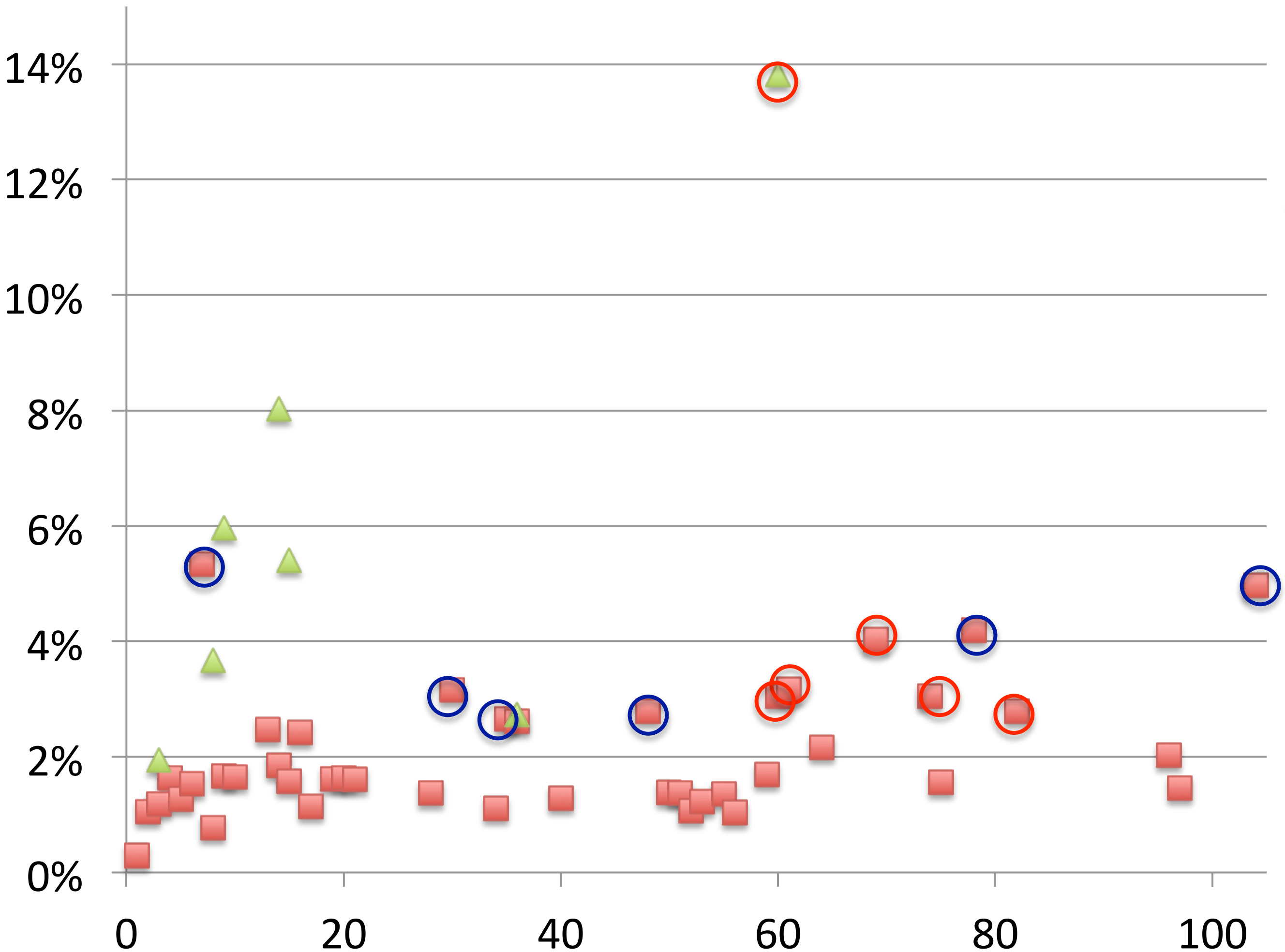
HPCG AND HPGMG EFFICIENCY



HPCG AND HPGMG EFFICIENCY



HPCG AND HPGMG EFFICIENCY



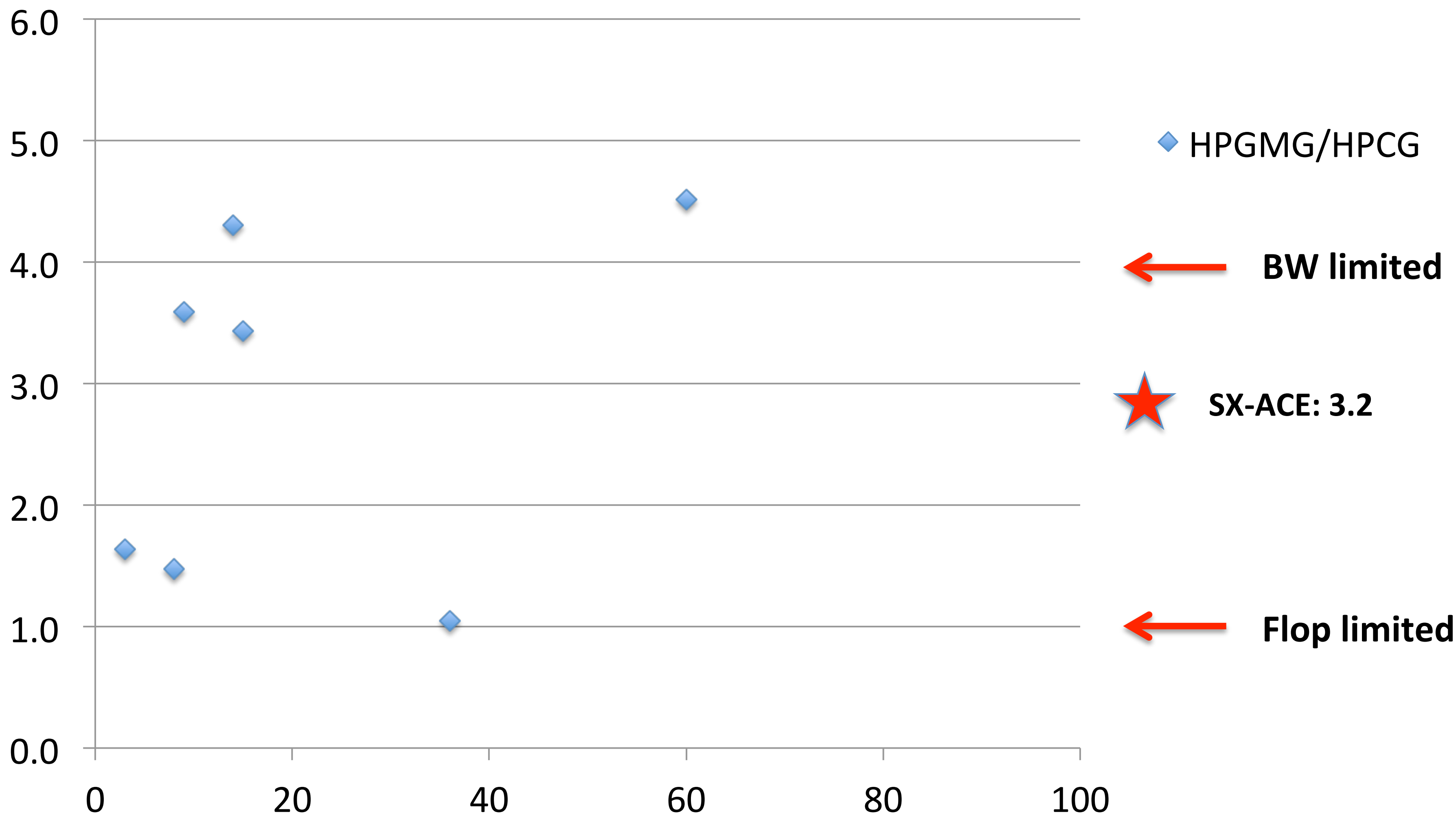
Tohoku NEC SX-ACE:

- **Not in TOP500**
- **123.1 Tflop/s**
- **HPL: 93.9%**
- **HPCG: 10.7%**
- **HPGMG: 33.8% (off this chart)**
- **1 Byte/Flop**

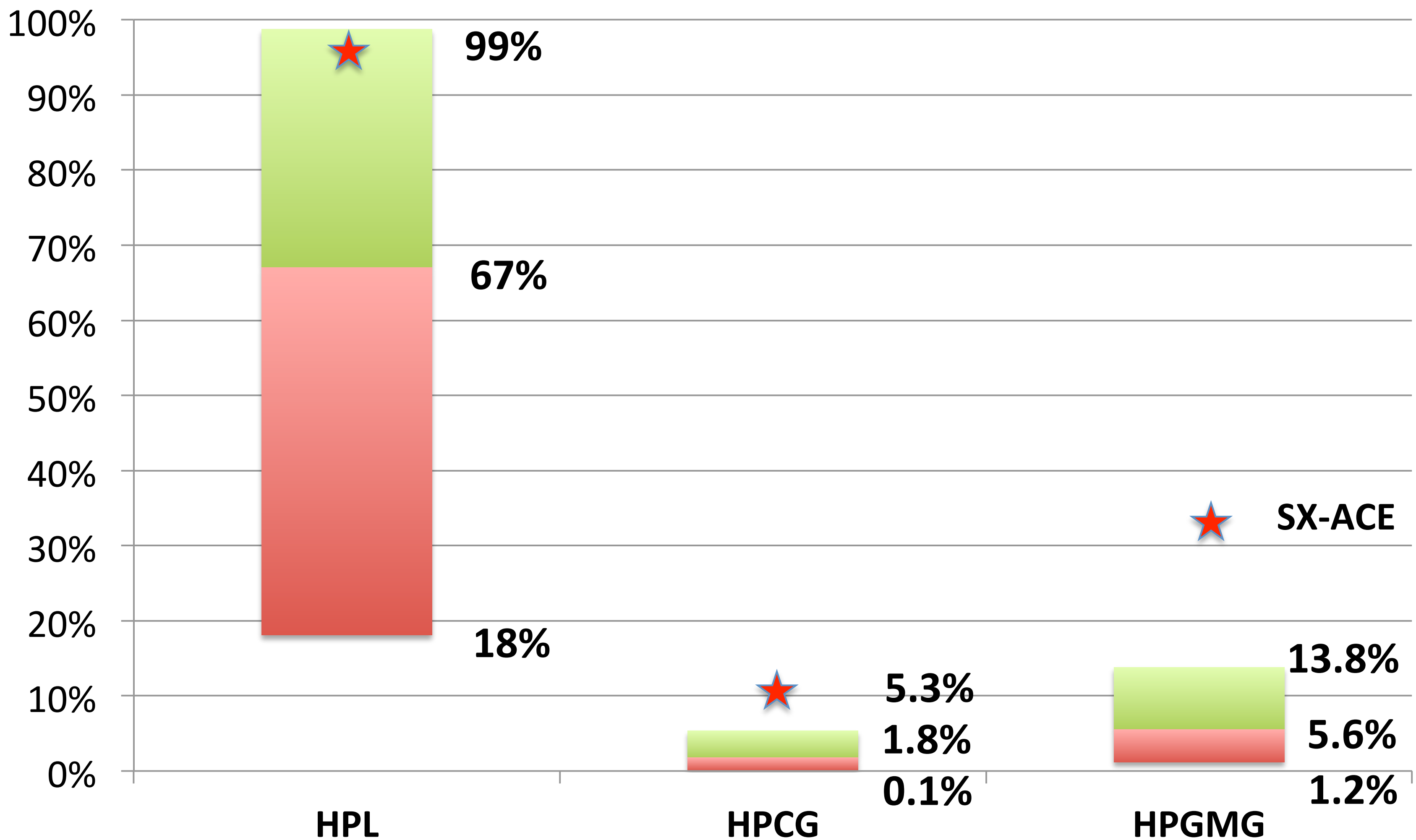
½ Byte/Flop

< 1/4 Byte/Flop

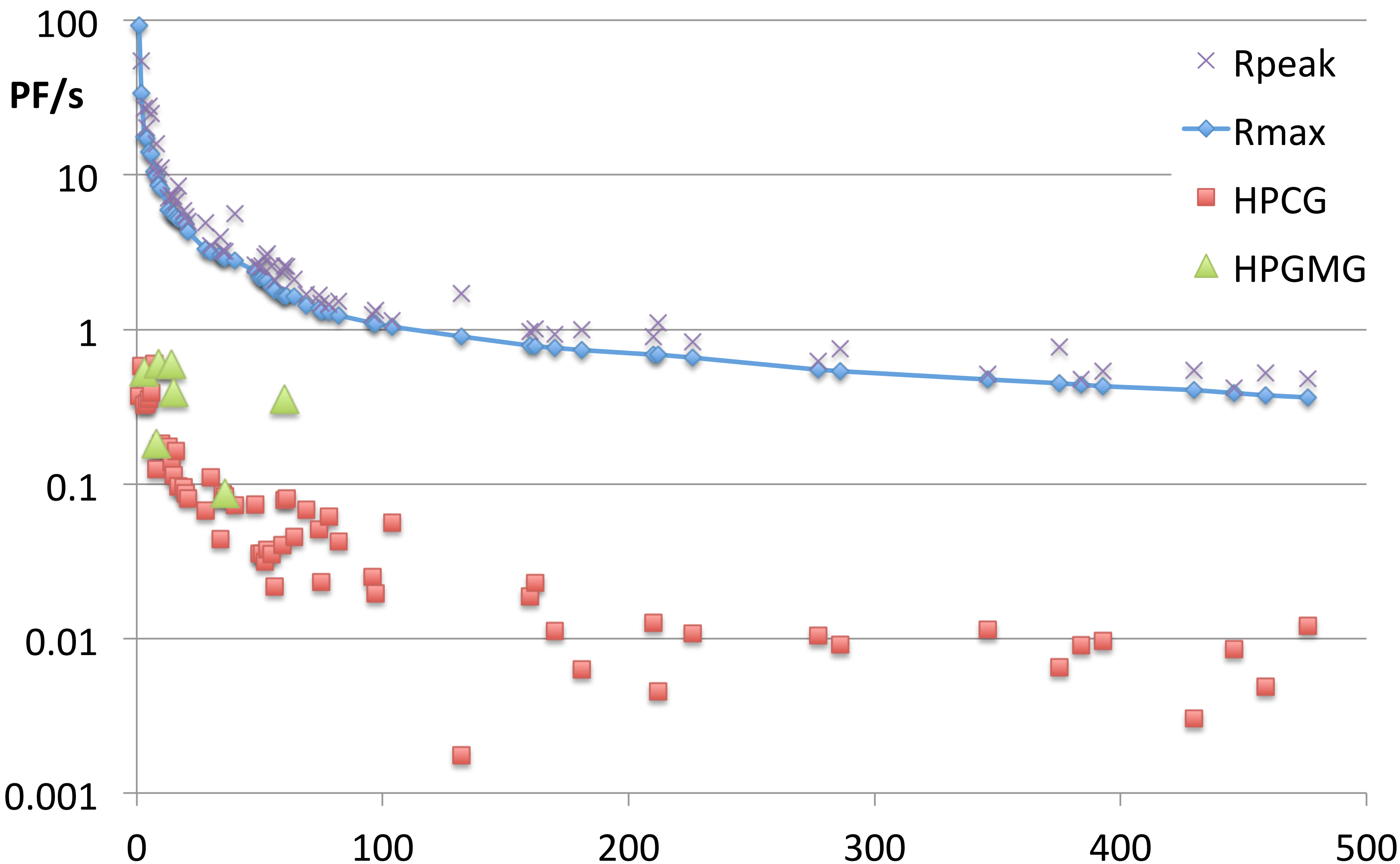
HPCG AND HPGMG EFFICIENCY



RANGE OF EFFICIENCIES



RE-ORDERING ON THE TOP500



RE-ORDERING ON THE TOP500