HPGMG

High Performance Geometric Multigrid: a New Computer Architecture Benchmark

Mark Adams (LBNL), Jed Brown (ANL,CU)
John Shalf (LBNL), Erich Strohmaier (LBNL)
Brian Van Straalen (LBNL), Sam Williams (LBNL)

IHPCF'15, 20-22 May 2015, Tianjin City China

Benchmarking

1. Drive system designers, provide tool/target for designers
2. Ranking – Top500
3. Define machine capability for procurement to allocate resources
   – How do you measure benefit of an exa-machine? Exa-science?
   – HPC facility production capability hard to measure

- HPGMG: a small & (conceptually) simple benchmark
  – Not a replacement for extensive procurement suites (eg, CORAL)
  – Sensitive many metrics: MPI BW & rates; DRAM BW; FPU, cache …
  – Balanced exercise of machines **emphasis on scalability** – tightly integrated
- Community needs build tools aid in allocating resources cost effectively
  – *Develop benchmark codes* (eg, **HPGMG**, CORAL benchmark suite, …)
  – Correlate benchmarks & applications – **data on efficacy of benchmarks**

Motivation: HPL is a distorting force in HPC

- We all dislike HPL's influence, depends perspective, we don't do chemistry
  - HPL good benchmark but abused – good HPL machine not good for most apps now
- HPL got a lot right:
  - Solve a global problem - algebraic equation solve - fully coupled
  - Best practices solver algorithm, well defined & understood – Gauss LU
  - Good implementations - block all levels memory, etc.

1. Applications changed: now exploit structure get O(1) work / word
2. Architectures changed: trend in relative memory bandwidth decay
   - My experience: stored matrix, 3D elasticity, algebraic multigrid, PETSc (IBMs)
   - 1994 (92) IBM RS6K (my first AMG): 25% R_{peak}
   - 2004 (00) IBM SP3 Gordon Bell (special category): 5% R_{peak}
   - 2014 (12) IBM BG/Q (PETSc ex56): 1% R_{peak}

- Applications now use algorithms that exploit structure (e.g., multigrid)
- HPL sensitive FPU; FPU over provisioned for most apps (cheap & HPL influence)
- HPL and HPCG each stress one component (FPU & DRAM bandwidth resp.)
  - HPGMG is sensitive to: MPI BW & message rates, DRAM BW, FPU, OMP runtime…

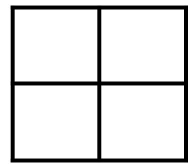
Alternate Benchmarks

- **NAS Parallel Benchmarks / NPB (1991)**
  - 8 benchmarks including CG (stored matrix), MG (ccPoisson), FFT, …
  - specified problem sizes, strong scaled
- **HPCC (2005)**
  - weak scaled
  - STREAM: 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
- **HPCG (2013)**
  - solves a FEM problem on a structured grid using a stored matrix PCG algorithm
  - weak scaled
  - after discussions with us, a (2-level) MG preconditioner was added.
- **Graph500**
  - BFS on graphs
  - little/no FP (targets a different domain)
  - specified problem sizes

HPGMG design & philosophy

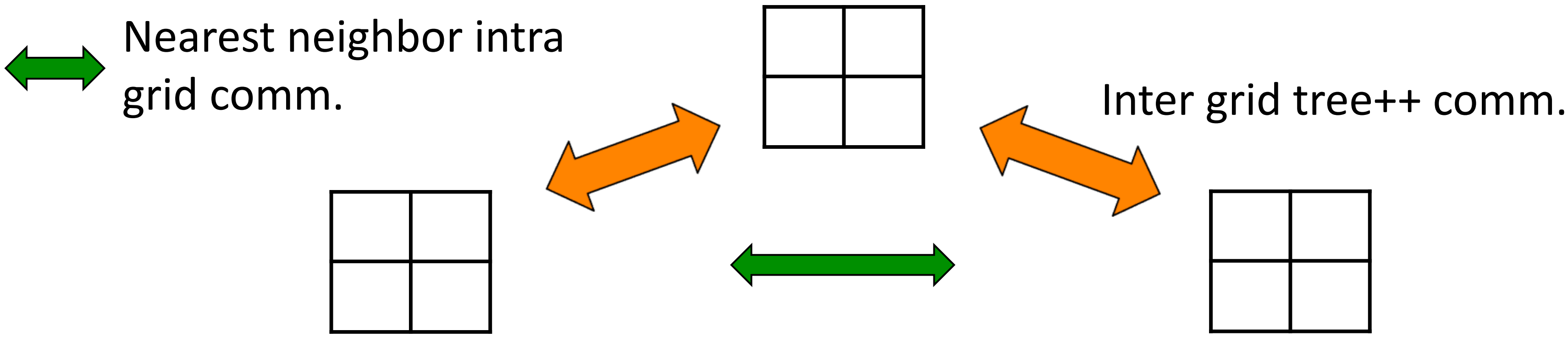
- Design: Full geometric multigrid (FMG) Laplacian solve Cartesian grids
  - Non-iterative, asymptotically exact solver with O(N) work complexity
  - Built-in correctness verification – oblivious to floating point errors
  - **Metric: equations solved / sec**
- Scale Free specification: Solution independent parallelization strategy
  - Do not mathematically punish/reward fine/coarse grain parallelism
  - No problem size specified
- Stresses interconnect – global tree w/ non-trivial software kernels
  - **Reward systems that are tightly integrated**
  - Hard problem implicitly demands good end-to-end engineering (HPL)
- Benchmark remain relevant indefinitely
  - Increasingly effective proxy over time more apps use efficient algorithms
- HPGMG is good direct proxy for matrix free stencil & FE apps, & multigrid, but intended to mirror sensitivities of apps with no superficial similarities:
  - Correlate apps and HPGMG with metrics – **efficacy demonstrated w/ data**
  - Metrics that apps are sensitive to: MPI BW & message rate, DRAM BW, …

Multigrid Tree++ & Nearest Neighbor Communication Patterns



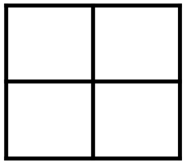
FMG starts with accurate solve on coarsest grid

Multigrid Tree++ & Nearest Neighbor Communication Patterns



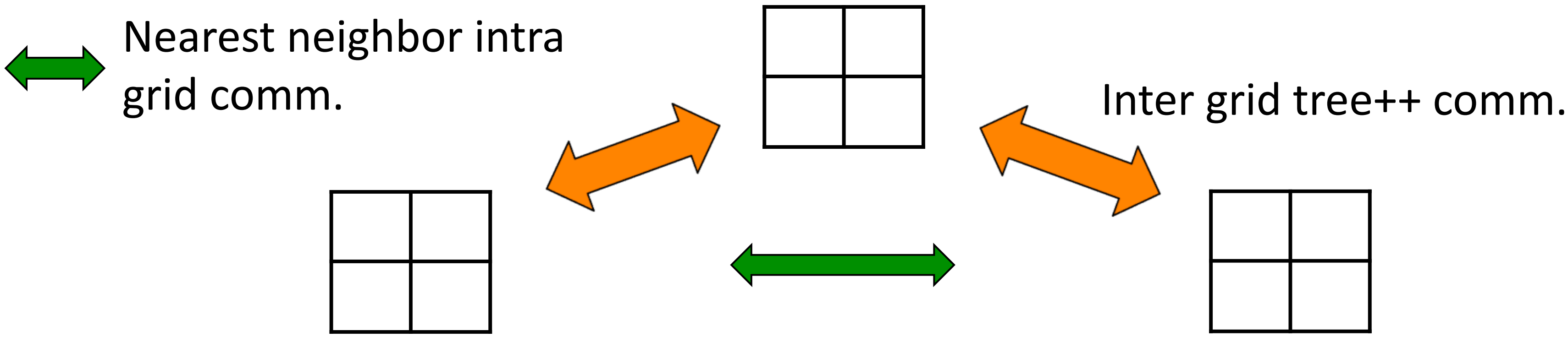
Refine grid split processes, building tree

Multigrid Tree++ & Nearest Neighbor Communication Patterns



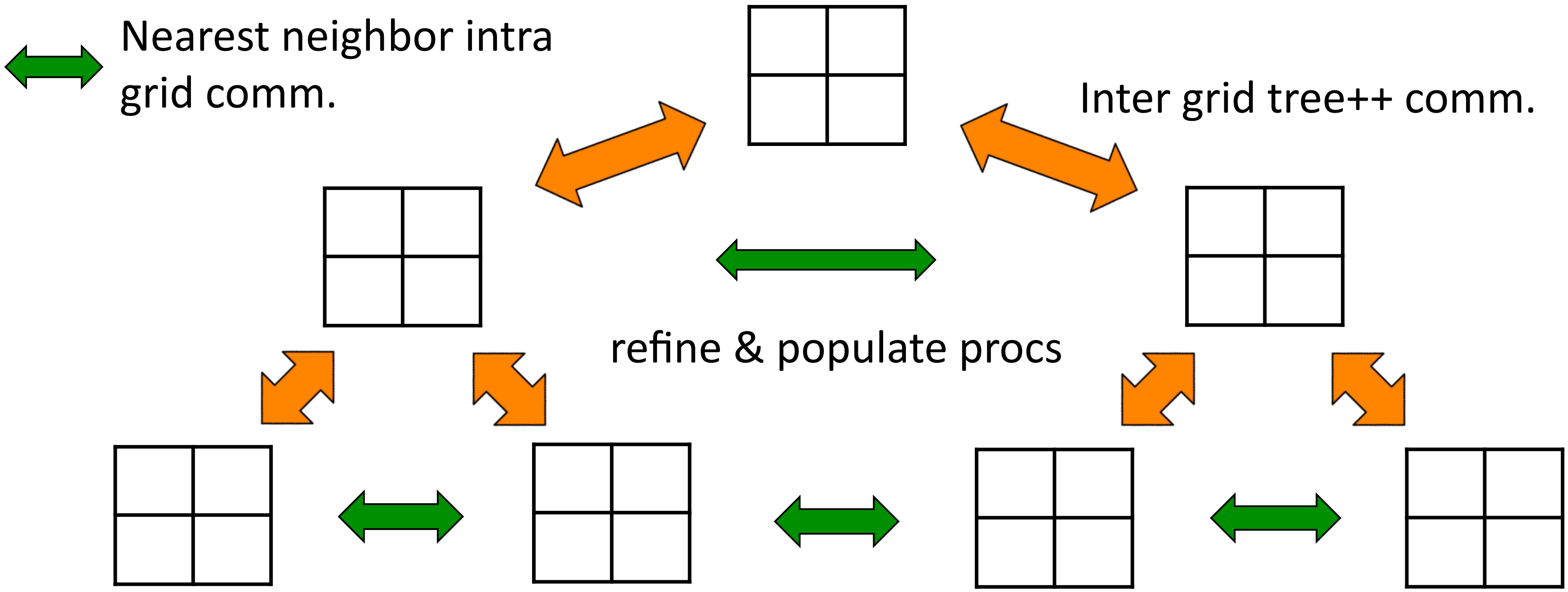
FMG goes back to coarse grid after each new level

Multigrid Tree++ & Nearest Neighbor Communication Patterns

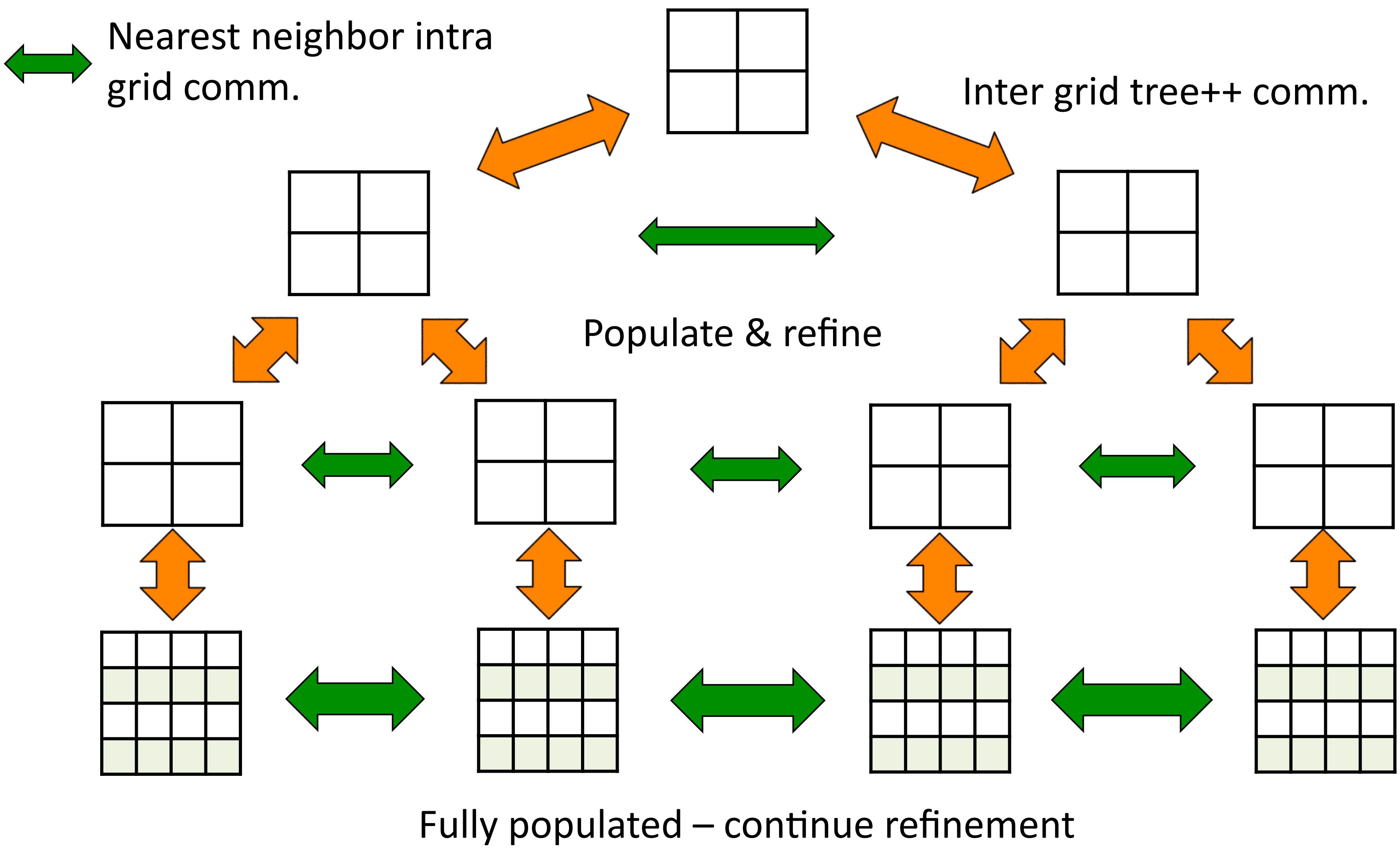


Back down

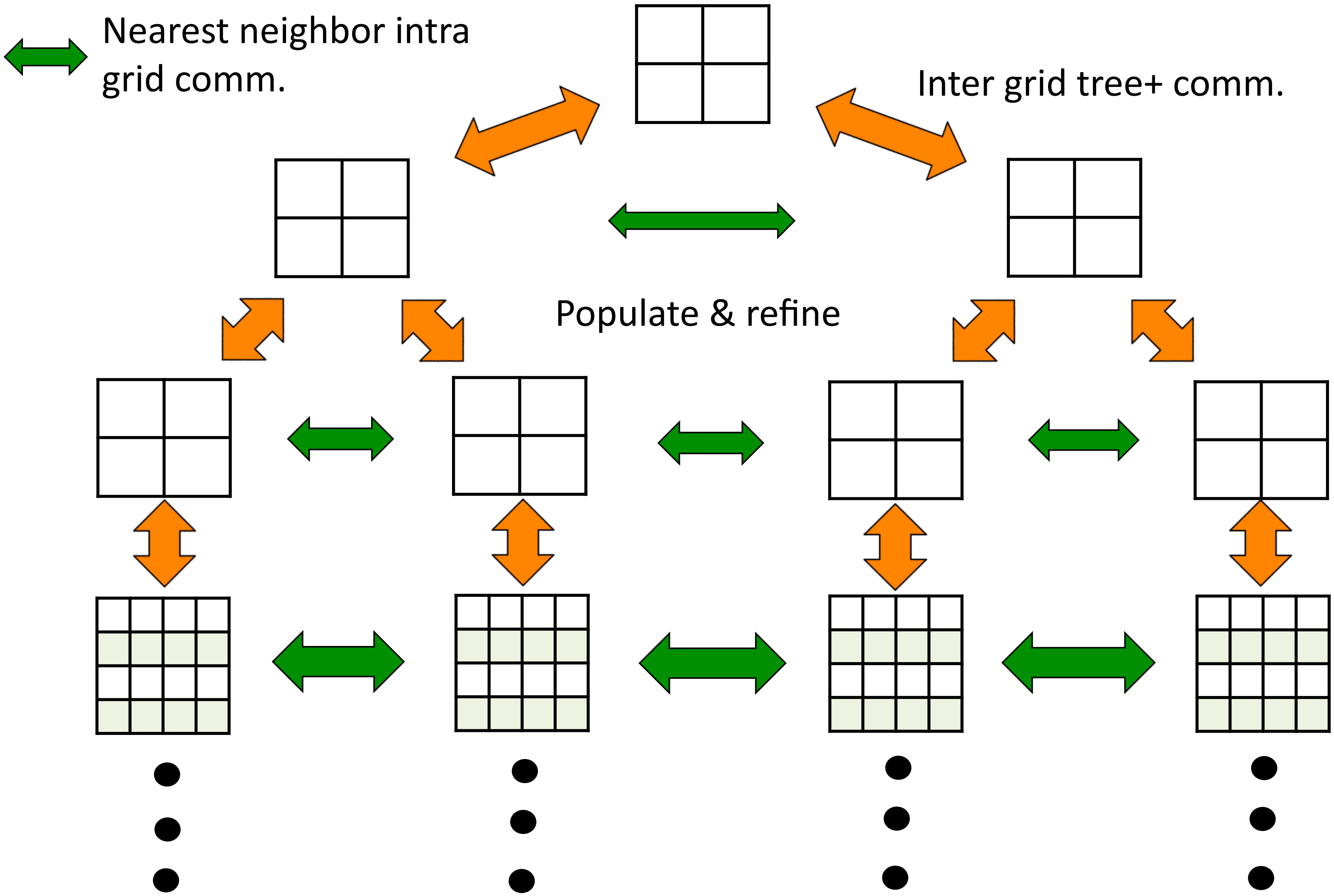
Multigrid Tree++ & Nearest Neighbor Communication Patterns



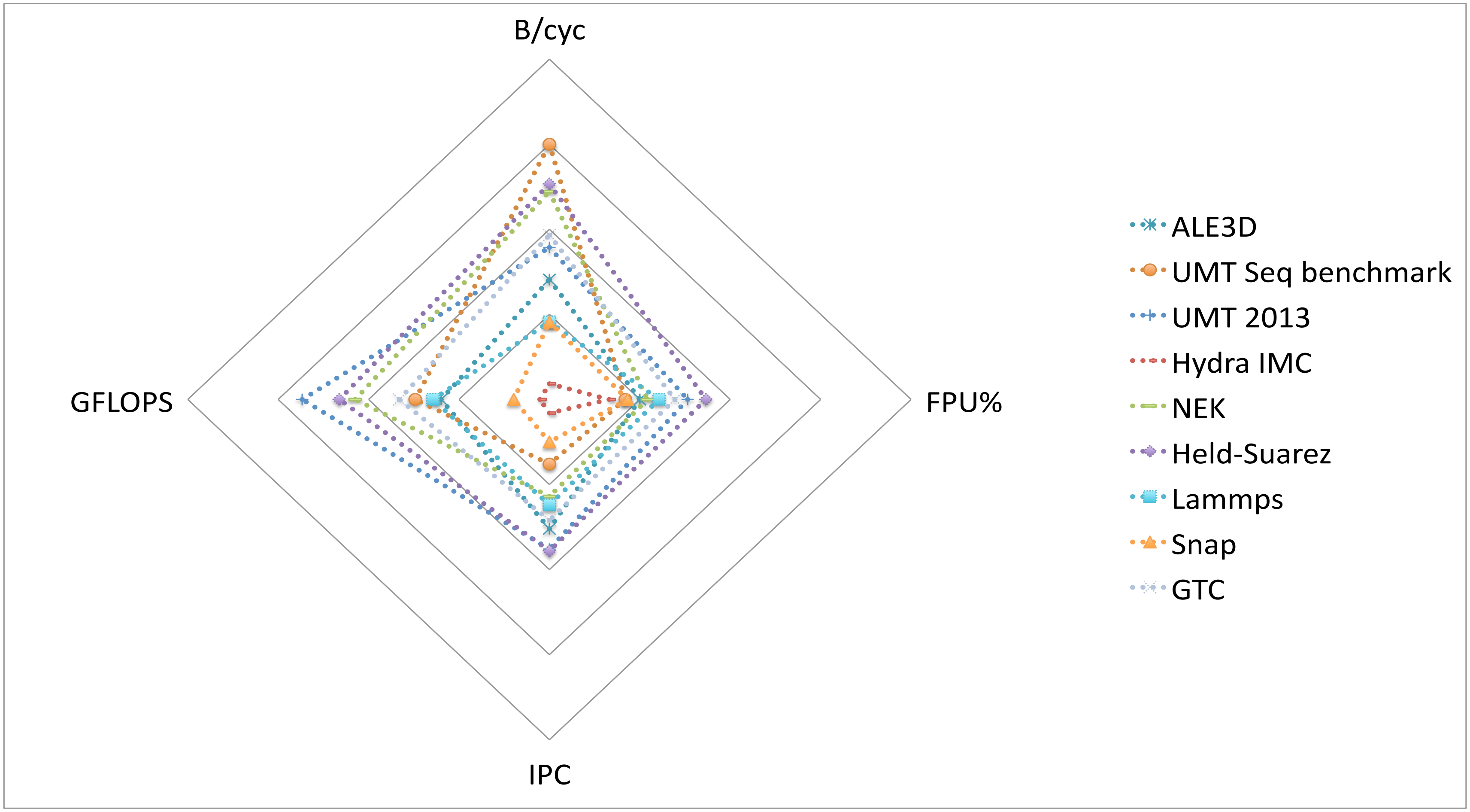
Multigrid Tree++ & Nearest Neighbor Communication Patterns



Multigrid Tree++ & Nearest Neighbor Communication Patterns

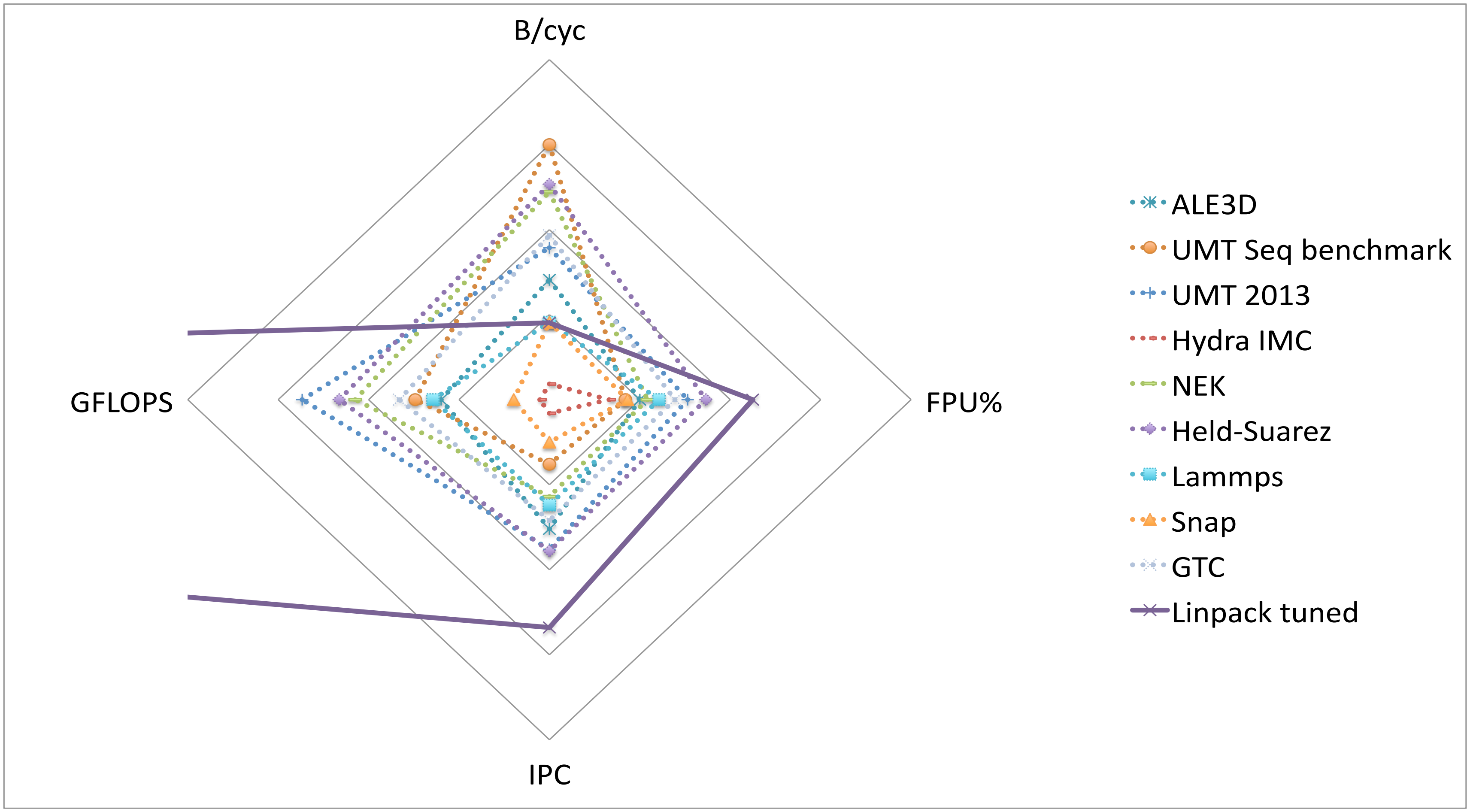


Metrics: Applications & Benchmarks HPM on IBM BG/Q data



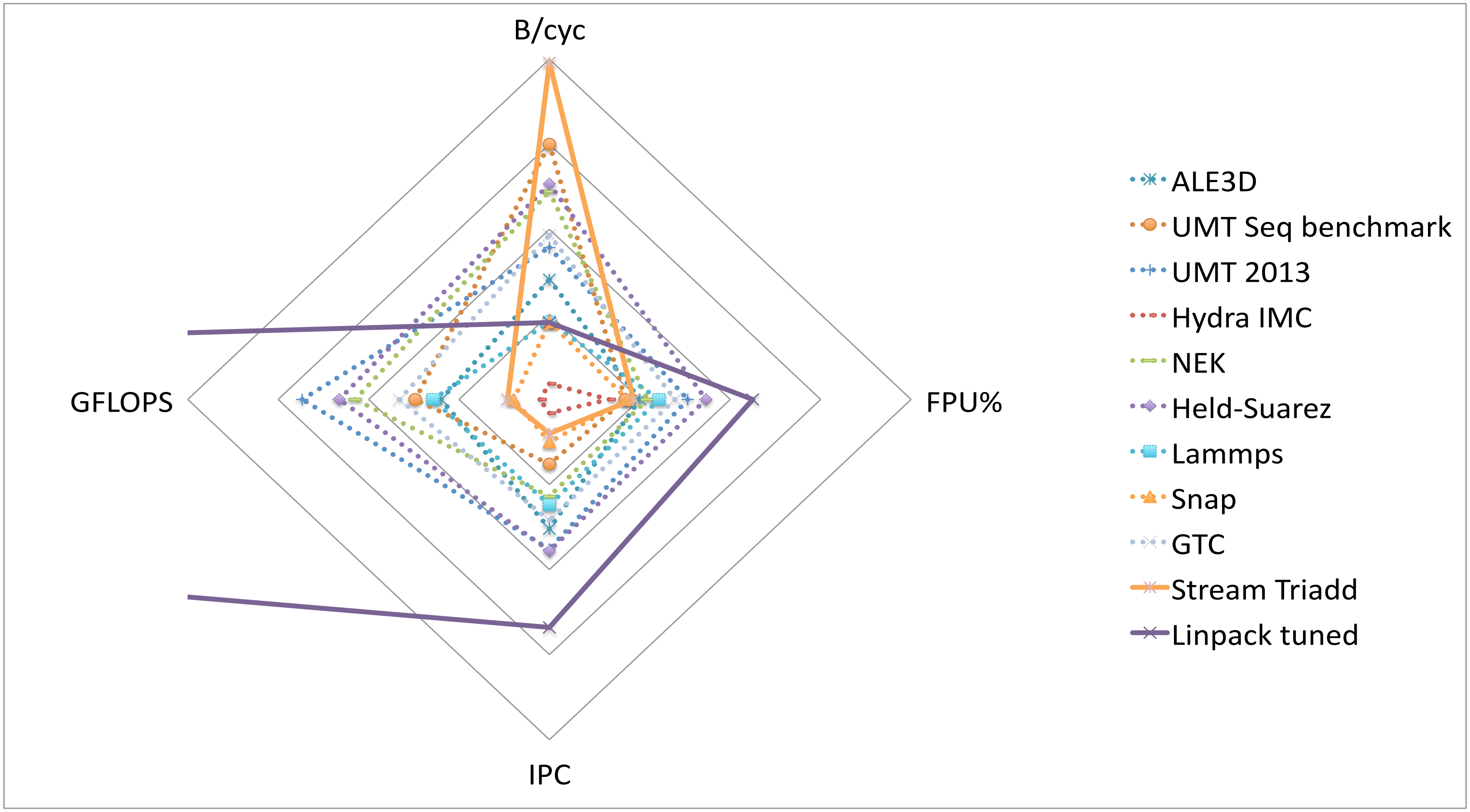
Courtesy Bert Still and Ian Karlin, LLNL

Metrics: Applications & Benchmarks
HPM on IBM BG/Q data



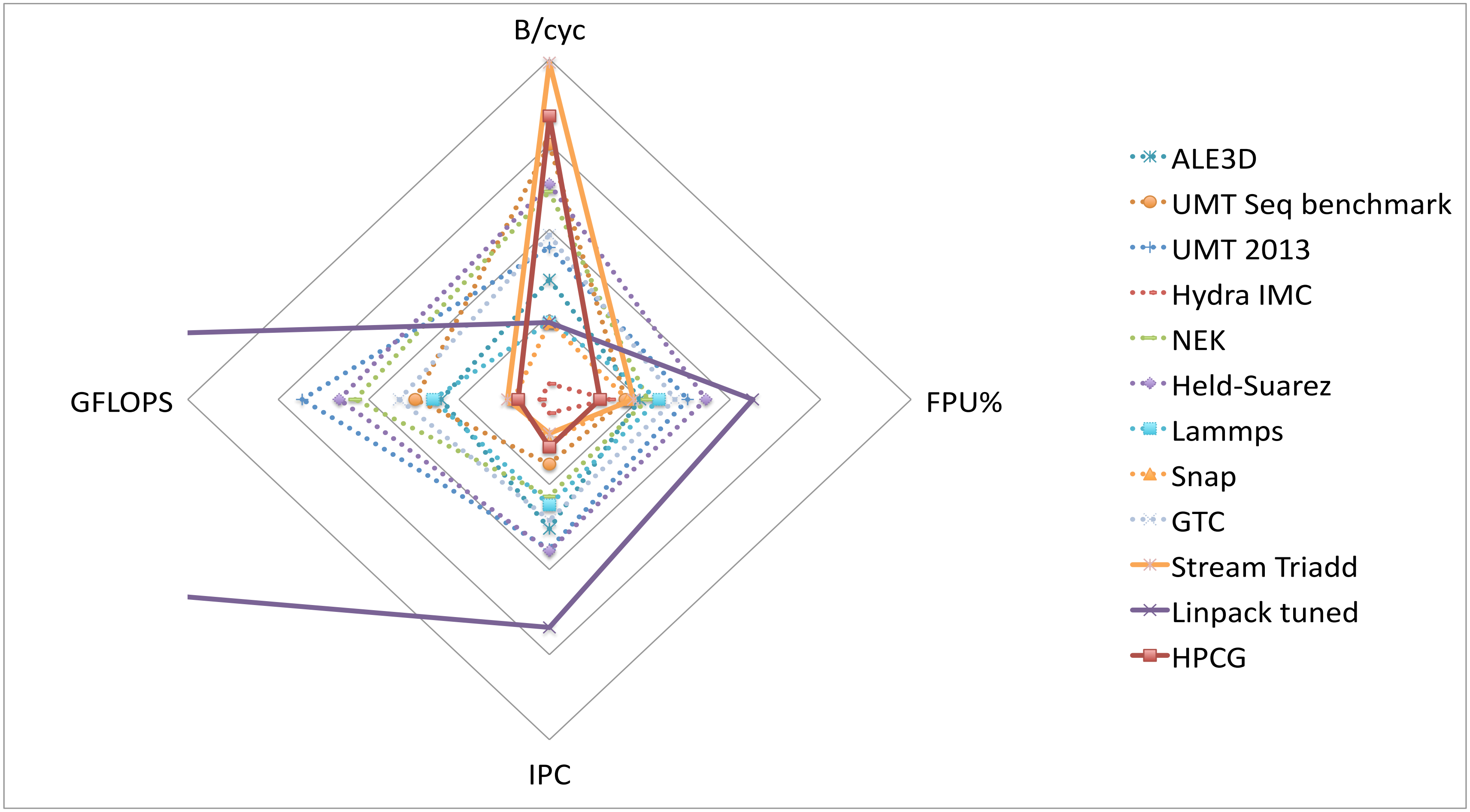
Courtesy Bert Still and Ian Karlin, LLNL
Qbox & HACC removed (by LLNL)

Metrics: Applications & Benchmarks HPM on IBM BG/Q data



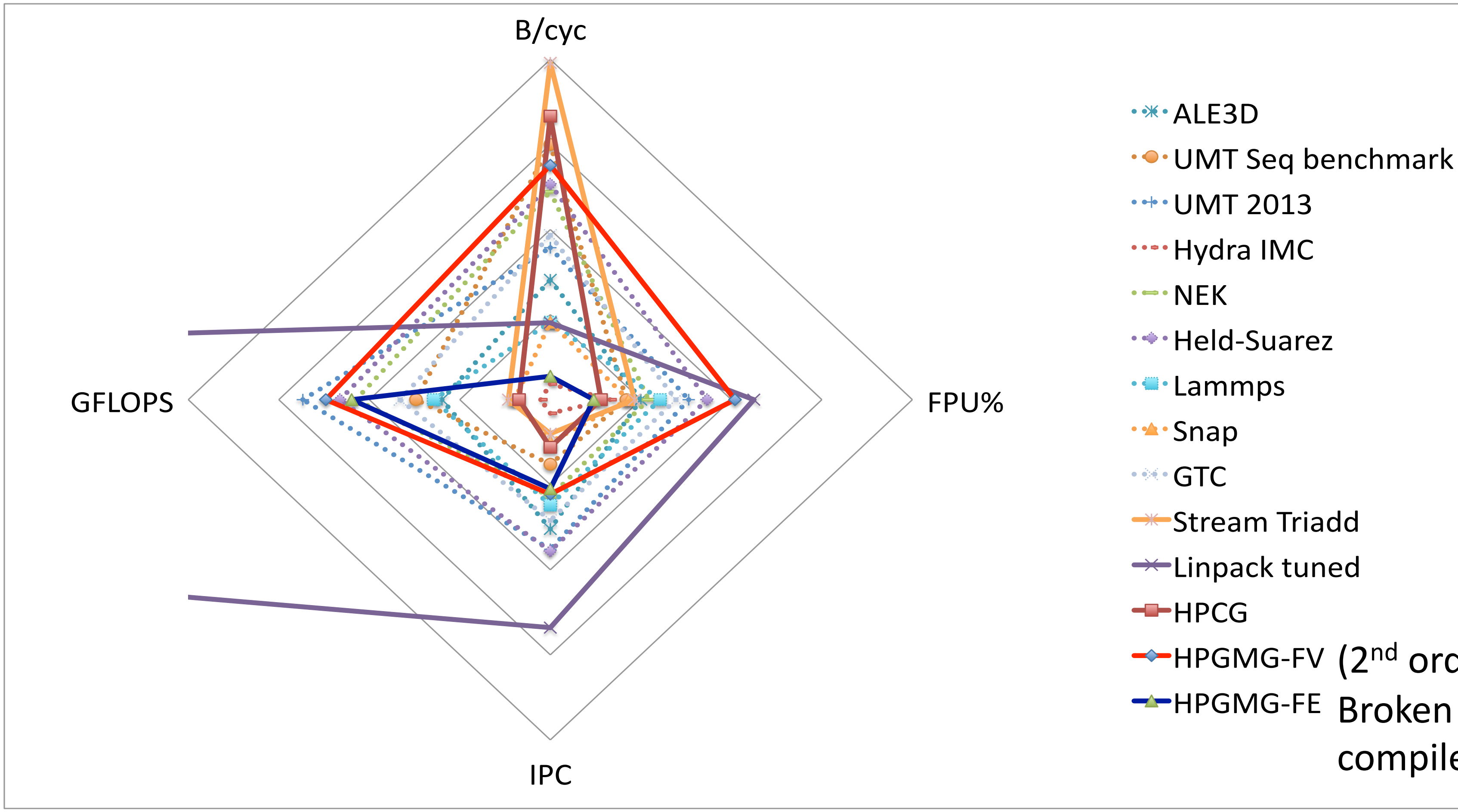
Courtesy Bert Still and Ian Karlin, LLNL

Metrics: Applications & Benchmarks HPM on IBM BG/Q data



Courtesy Bert Still and Ian Karlin, LLNL

Metrics: Applications & Benchmarks HPM on IBM BG/Q data



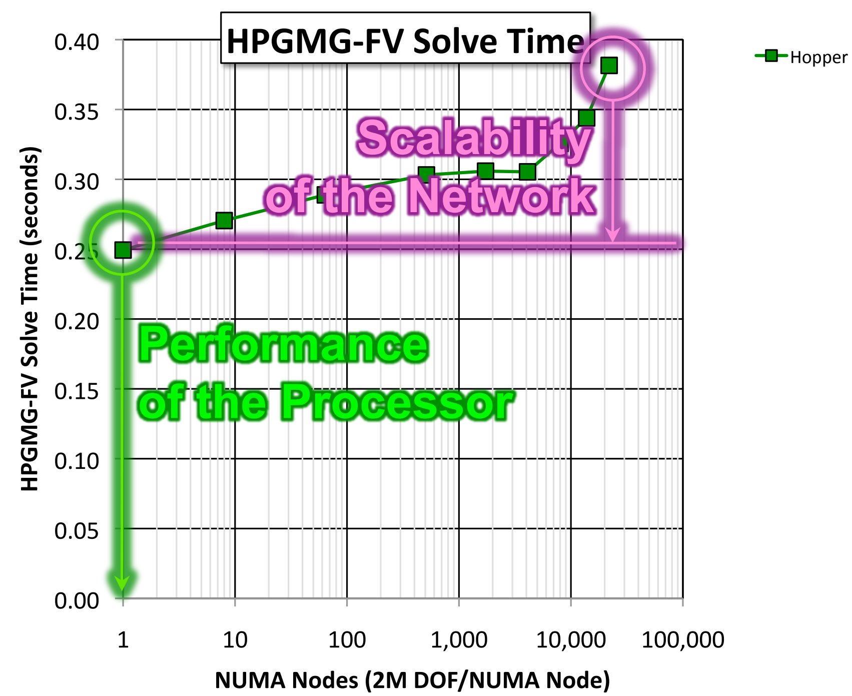
(2nd order)

Broken compiler

Courtesy Bert Still and Ian Karlin, LLNL

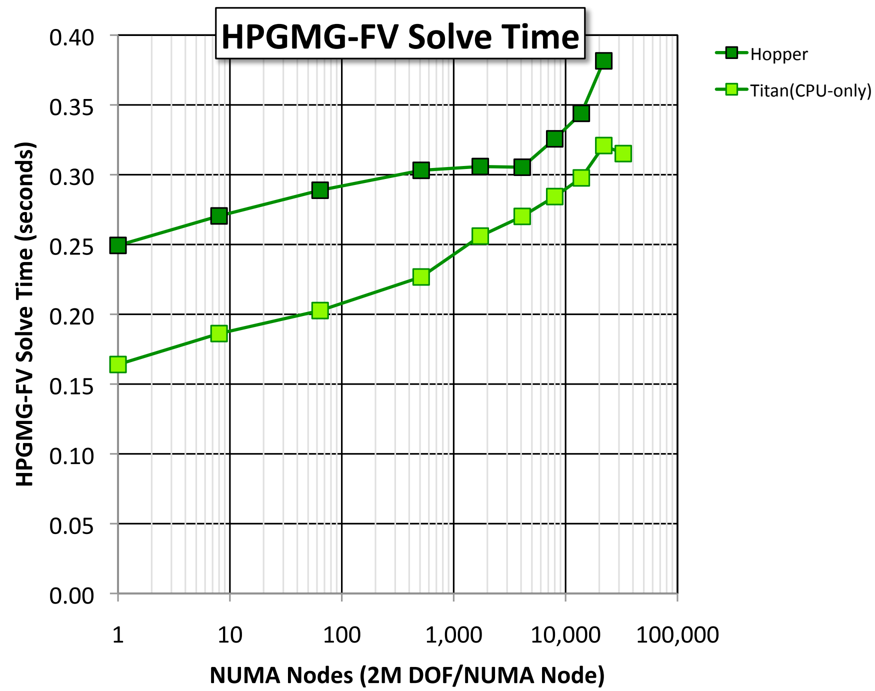
HPGMG-FV

- 3D Torus (Gemini)
- Hopper (Cray XE6)
  - single process multigrid solve time is fast (250ms)
  - performance degrades at scale
  - larger problem sizes mitigate this lack of scalability



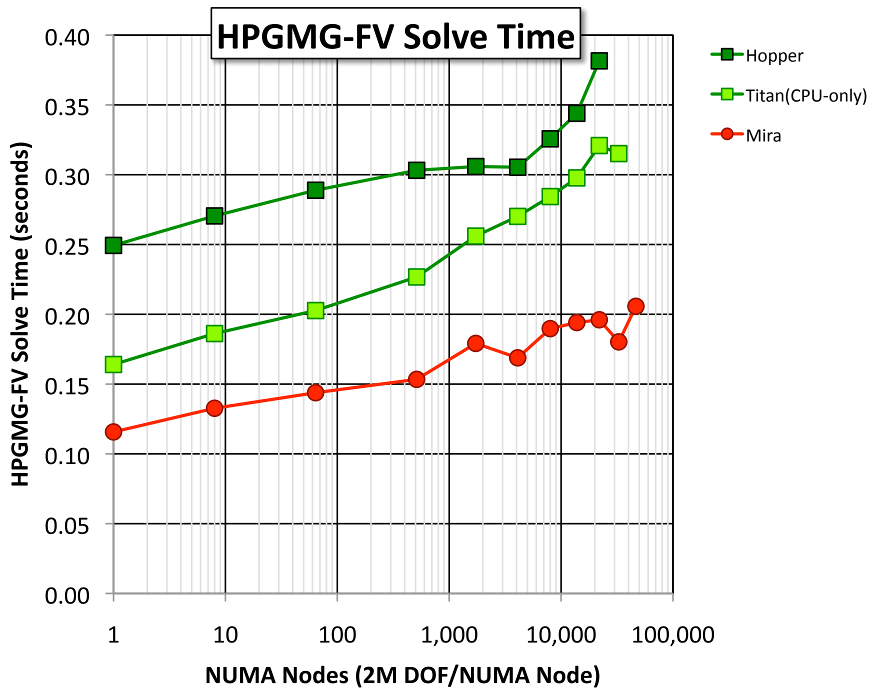
HPGMG-FV

- 3D Torus (Gemini)
- Hopper (Cray XE6)
  - single process multigrid solve time is fast (250ms)
  - performance degrades at scale
  - larger problem sizes mitigate this lack of scalability
- Titan (Cray XK7)
  - use only CPUs (same MPI+OpenMP)
  - delivered 50% better performance per socket and ~2x better overall performance
  - However, as Hopper and Titan both use Gemini, the network impeded performance at scale



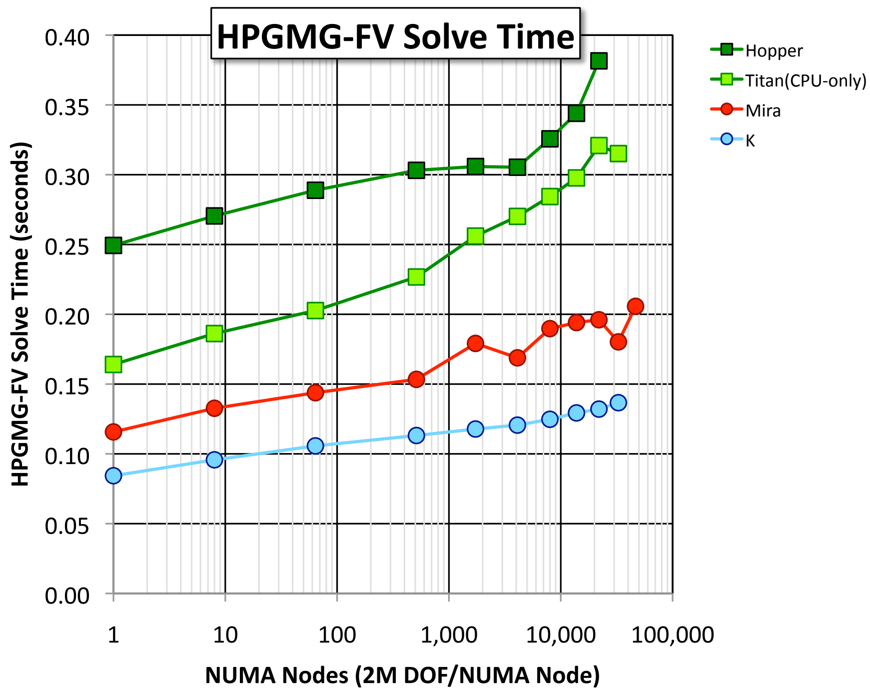
HPGMG-FV

- Blue Gene/Q
- Mira (Blue Gene/Q)
  - custom processor enabled better performance per socket
  - custom network (5D torus) enabled better scalability



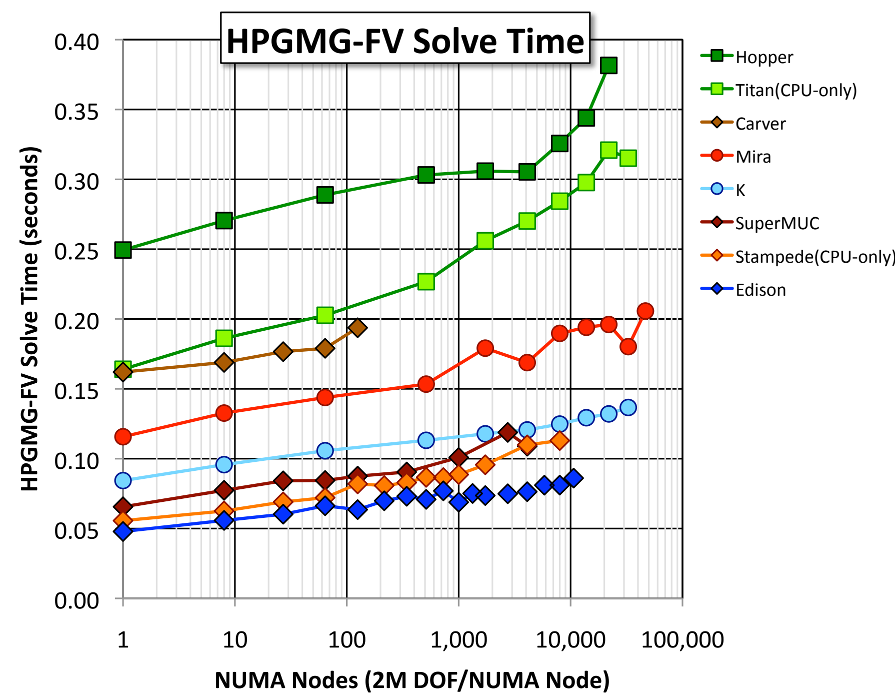
K (6D Torus/Mesh)

- K (Sparc VIIIfx) at RIKEN
  - less flops per proc than BGQ
  - more bandwidth per proc than BGQ
    = deliver better HPGMG-FV performance per node.
  - TOFU (6D) delivered similar (but smoother) scalability to BGQ



Fat Trees and Dragonfly

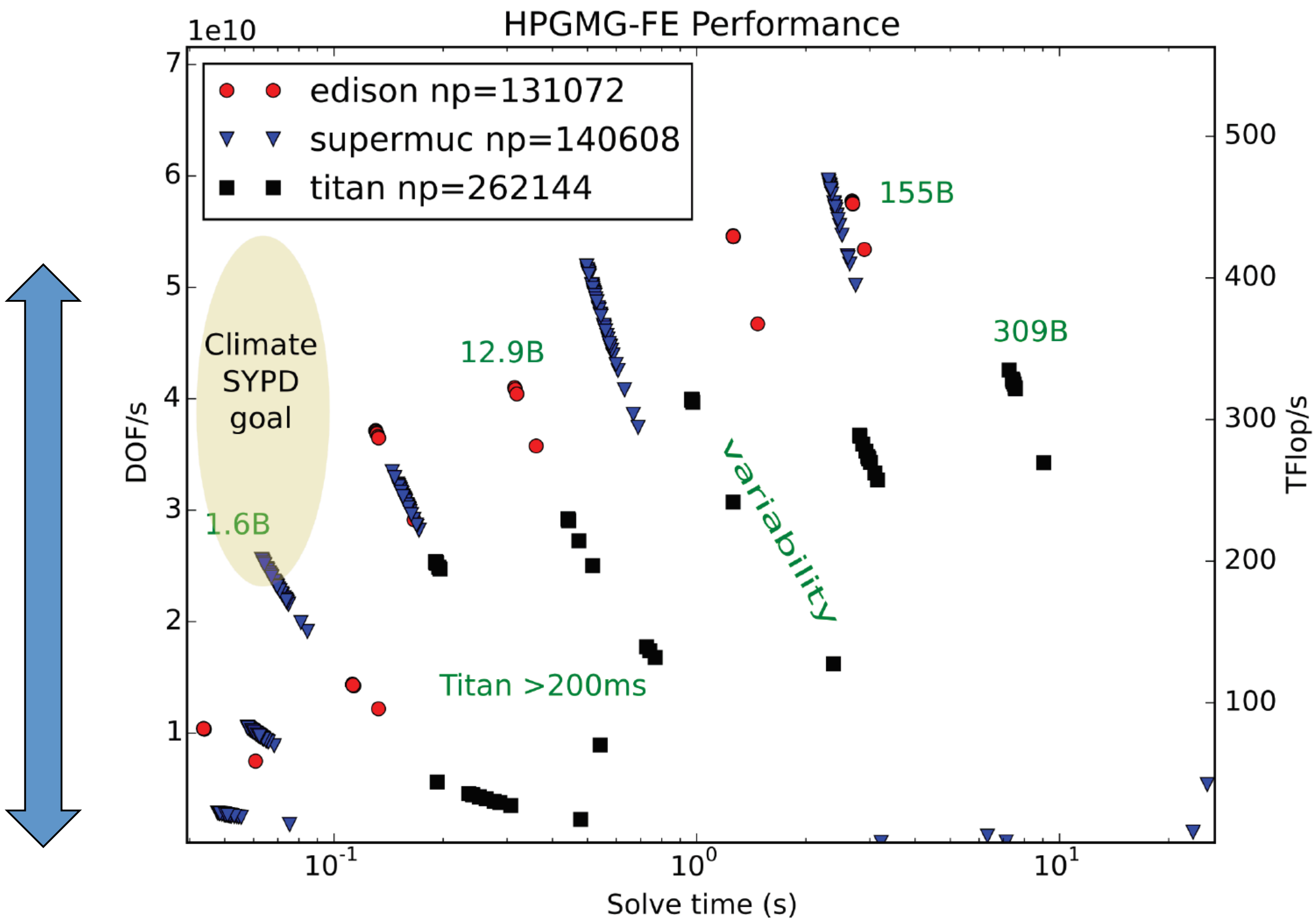
- Xeon processors (IVB, SNB, NHM) are common on the Top500 today
- Xeon performance defined by #cores, processor frequency, and memory frequency
- **Fat Trees saw degraded scaling beyond 1K sockets**
- XC30/Aries (Dragonfly), K (6D), and BGQ (5D) **continued to scale well from 1K-48K sockets**



Machine spectra

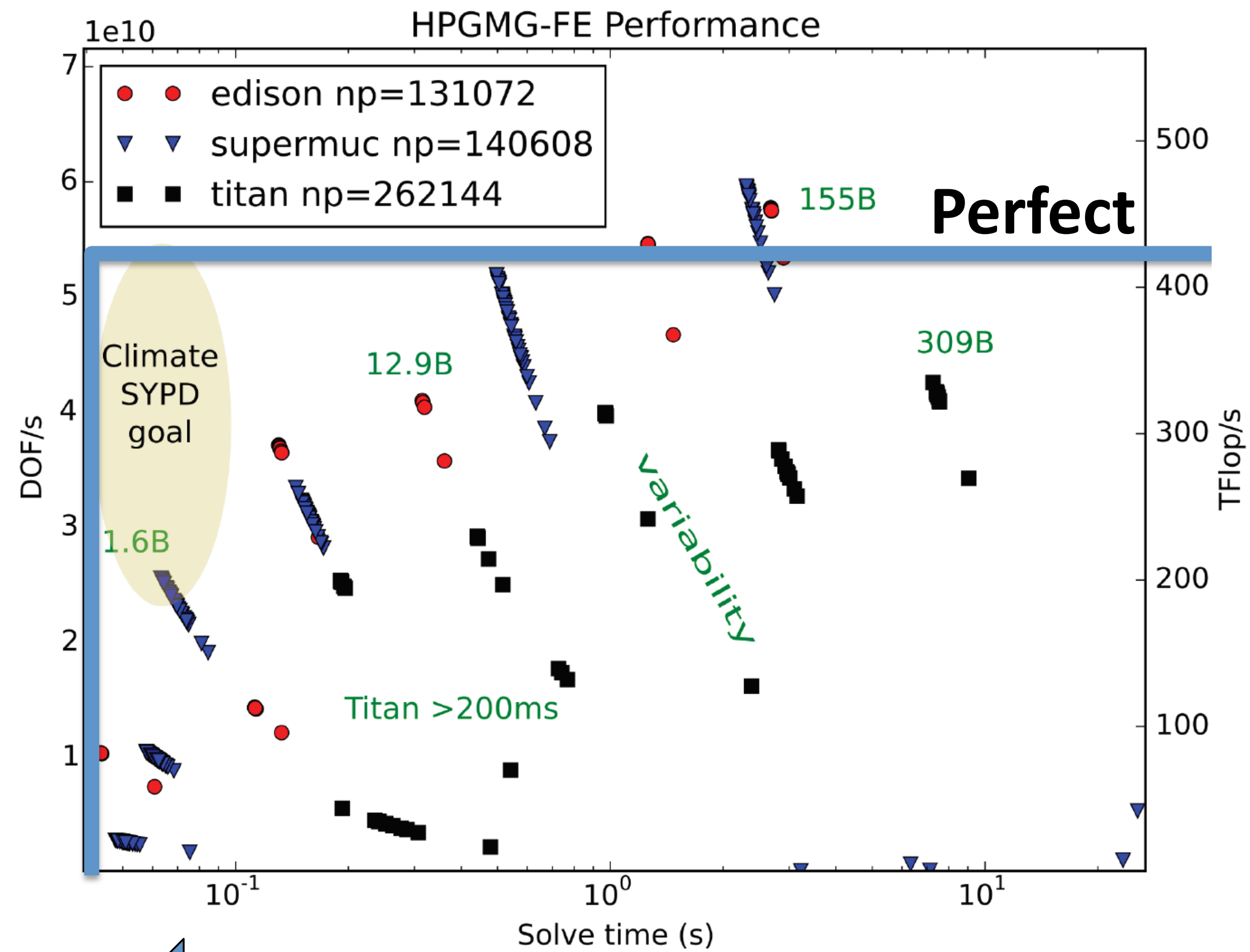
HPGMG-FE on Edison, SuperMUC, Titan

The metric



Machine spectra

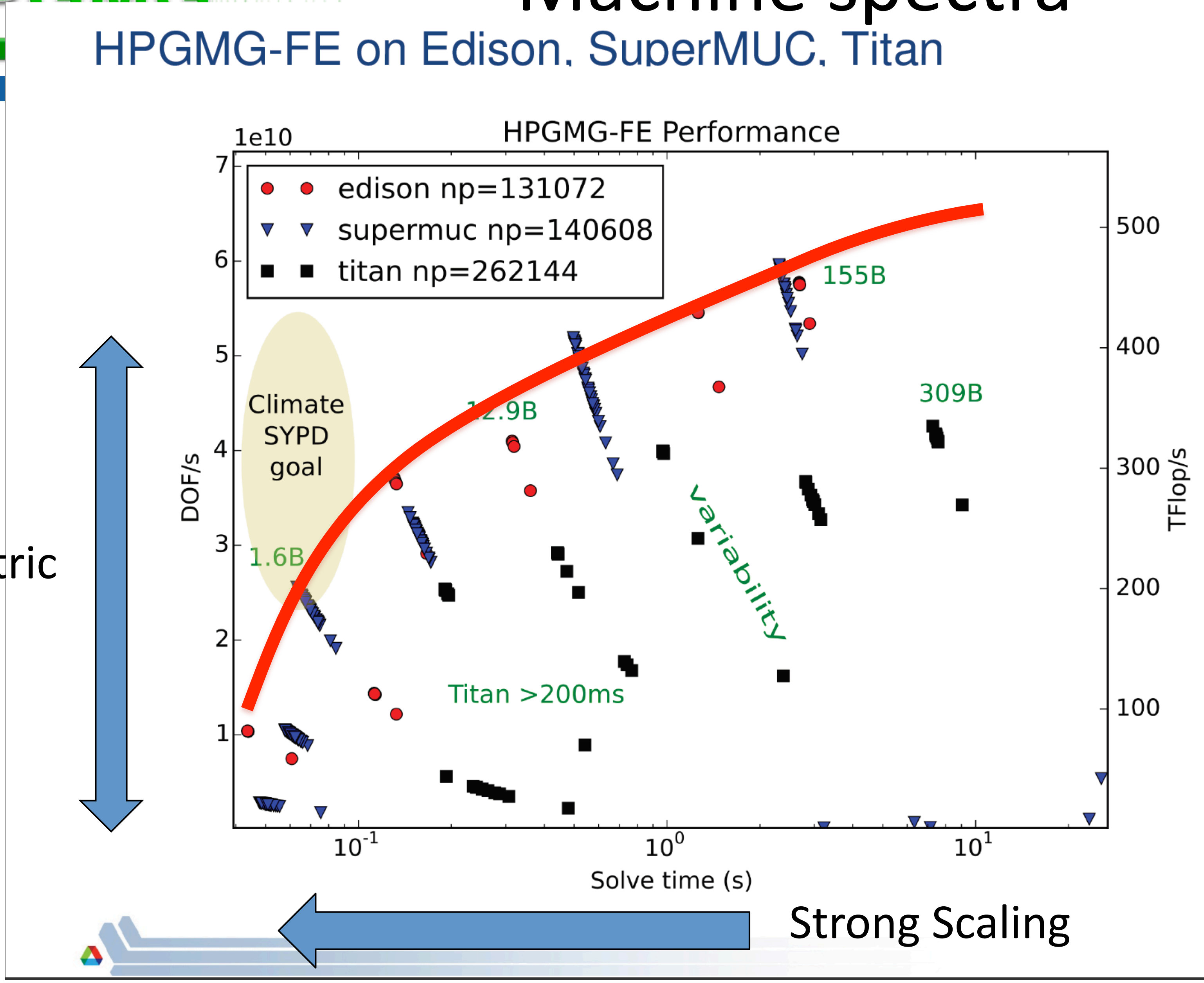
HPGMG-FE on Edison, SuperMUC, Titan



The metric

Strong Scaling

Machine spectra

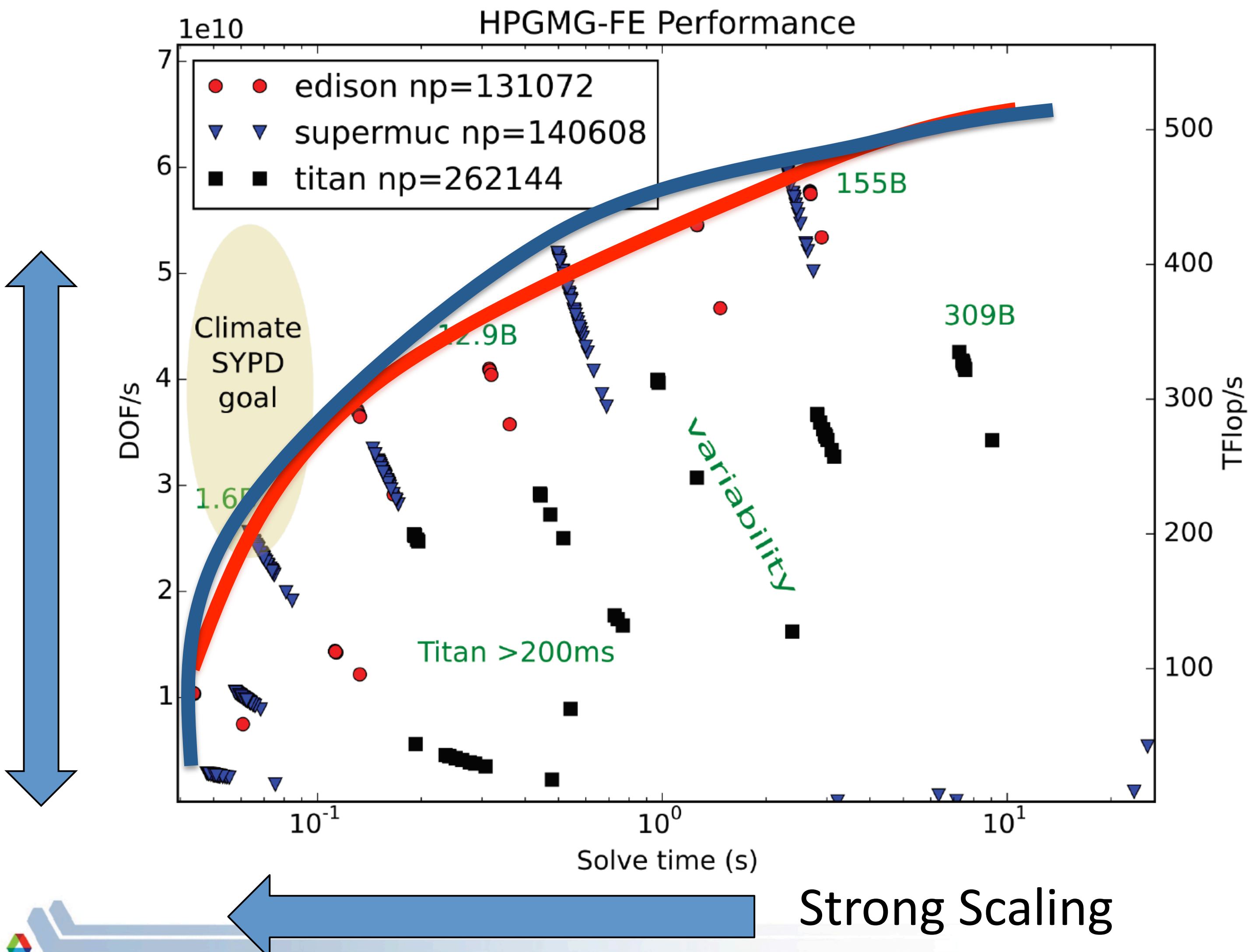


The metric

Strong Scaling

Machine spectra

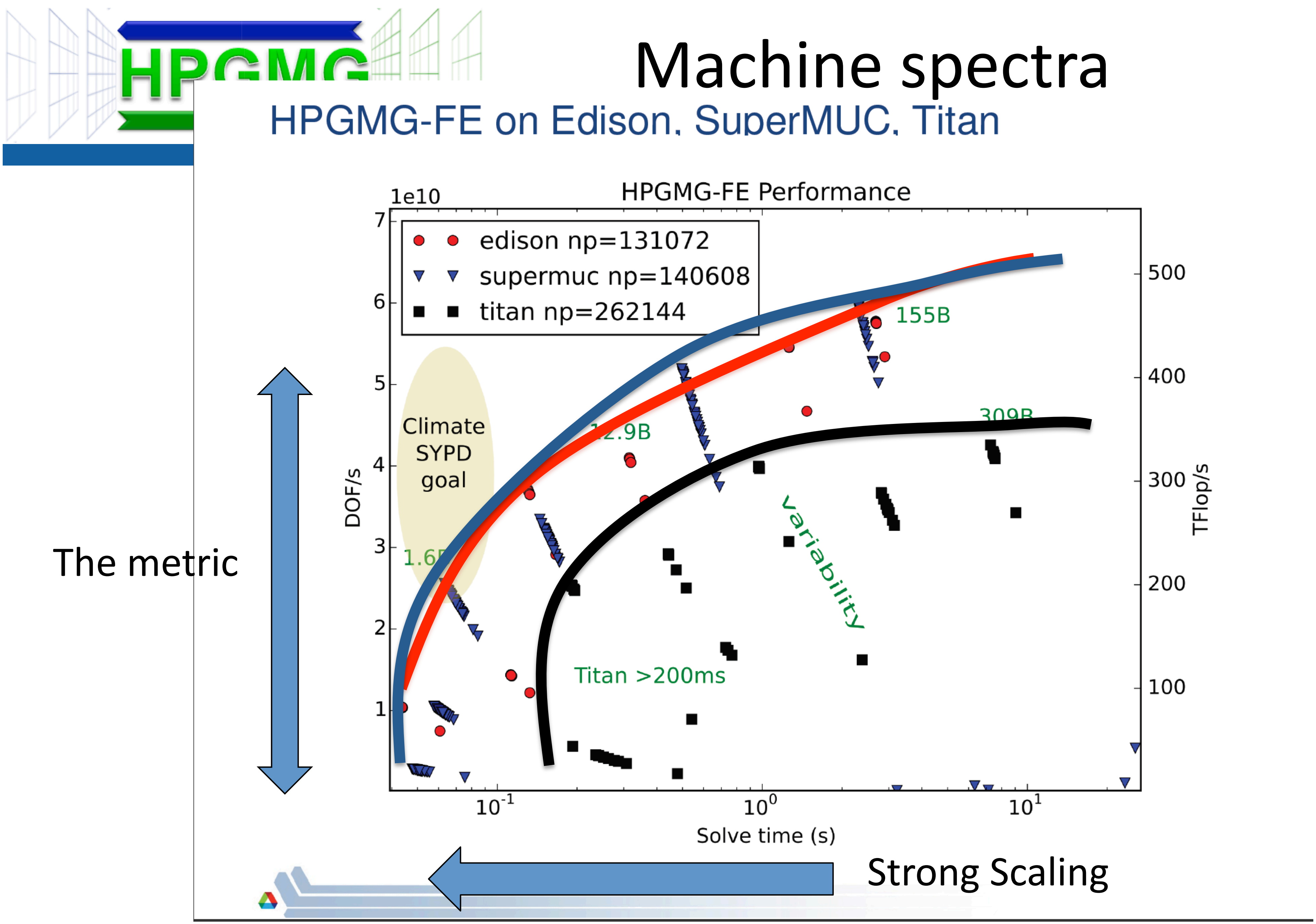
HPGMG-FE on Edison, SuperMUC, Titan



The metric

Strong Scaling

Machine spectra



GMG500: getting ready first release

HPGMG-FV Rank	System	Site	DOF/s	Fraction of System	Parallelization MPI	OMP	DOF per Process	Top500 Rank
1	K	RIKEN	2.83E+12	100%	82944	8	72M	4
2	Mira	Argonne	7.21E+11	100%	49152	64	16M	5
3	Edison	NERSC	3.85E+11	100%	131072	1	4M	24
4	Titan (CPU-only)	Oak Ridge	2.53E+11	88%	32768	8	16M	2
5	Stampede (CPU-only)	TACC	1.49E+11	64%	8192	8	2M	7
6	Hopper	NERSC	1.21E+11	86%	21952	6	2M	44
7	Piz Daint (CPU-only)	CSCS	1.02E+11	78%	4096	8	18M	6
8	SuperMUC	LRZ	7.13E+10	15%	2744	8	16M	14
9	BiFrost	NSC	4.67E+10	98%	1260	16	176M	-
10	Stampede (MIC-only)	TACC	2.16E+10	8%	512	180	16M	7
11	Peregrine (IVB-only)	NREL	1.08E+10	18%	512	12	2M	-
12	Carver	NERSC	1.35E+09	5%	125	4	2M	-
13	Babbage (MIC-only)	NERSC	8.24E+08	30%	27	180	16M	-

hpgmg.org

- Three variants of HPGMG available (% peak FPU)
  - HPGMG-FV: Finite Volume, 2nd order, memory intensive (5)
  - HPGMG-FV: Finite Volume, (new) 4th order, memory/cache intensive (20)
  - HPGMG-FE: Finite Element, 3rd order, cache/floating-point intensive (25)
- Reference Implementations (C + MPI) on https://bitbucket.org/hpgmg/hpgmg
- We welcome community input and involvement:
  - hpgmg-forum@hpgmg.org
- Welcome collaboration on metric design, metric acquisition tools, application pool (eg, CORAL), an analysis of correlation with application pool to rationally design a Top500 like metric