How can we quantify performance versatility?

Jed Brown jedbrown@mcs.anl.gov (ANL and CU Boulder)

JointLab, Chicago, 2014-11-24

This talk: http://59A2.org/files/20141124-Versatility.pdf

Why do we need an exascale computer?

- Science & engineering demands
  - Model fidelity: resolution, multi-scale, coupling
  - Inversion/data assimilation
  - Optimization, control
  - Quantify uncertainty, risk-aware decisions
  - Sequence of forward simulations, each needing more time steps
- External requirements on time-to-solution
  - Policy: 5 SYPD for climate model to inform IPCC
  - Weather: 250x faster than real-time
  - Supply chain dynamics, manufacturing
  - Field studies, disaster response
  - Transient simulation is not weak scaling
- “weak scaling” [. . .] will increasingly give way to “strong scaling”

  [The International Exascale Software Project Roadmap, 2011]

Is the tail wagging the dog?

- Creative thinking about science/engineering problems
  - Guide software and hardware choices
  - Scientist: “your code doesn’t scale”
  - Center: “your machine is inappropriate for my application”
- Find corners of “science” that can use the machines
  - Incentivize solving problems hardware is good at
    - funding, allocations
    - “if your code doesn’t run on machine X, I’m not paying”
  - “The easiest way to make software scalable is to make it sequentially inefficient” – Gropp (1999)
    - Suboptimal modeling/algorithms are subtle inefficiency
  - Fragmenting high-end from low-end, no middle
  - Opportunity in advancing low-end to medium scale

Versatility

- Solve problems of maximum science/engineering interest
- At practical accuracy
- With desired turn-around time
- On available hardware
- Using modular, extensible software
- Reliably, debuggable
- Automate everything

Why a new benchmark?

Goodhart's Law

When a measure becomes a target, it ceases to be a good measure.

- But surely we can do better than HPL
  - Every feature stressed by benchmark should be **necessary** for an important application
  - Good performance on the benchmark should be **sufficient** for good performance on most applications

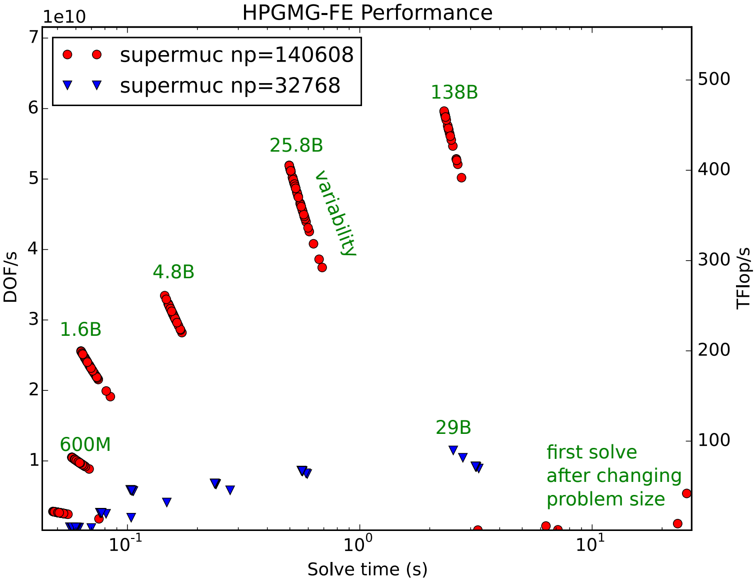
HPGMG: a new benchmarking proposal

- `https://hpgmg.org`, hpgmg-forum@hpgmg.org mailing list
- Mark Adams, Sam Williams (finite-volume), Jed Brown (finite-element), John Shalf, Brian Van Straalen, Erich Strohmeier, Rich Vuduc
- Building momentum, BoF at SC14
- Implementations

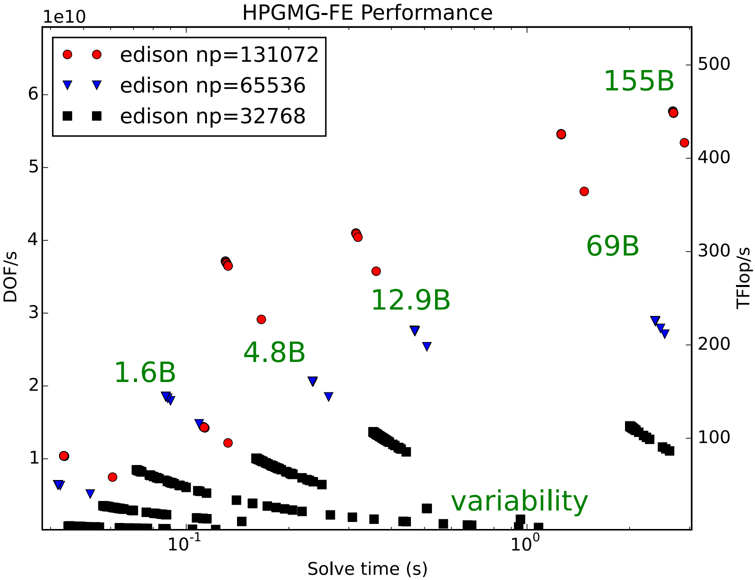
  Finite Volume memory bandwidth intensive, simple data dependencies

  Finite Element compute- and cache-intensive, vectorizes, overlapping writes
- Full multigrid, well-defined, scale-free problem
- Best-known algorithms, no “fat” left to trim
- Representative of structure-exploiting algorithms

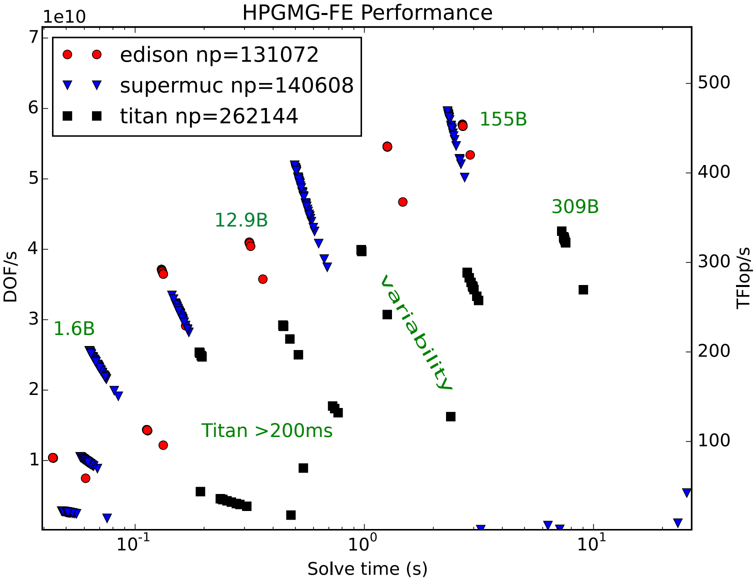
SuperMUC (FDR 10, E5-2680)



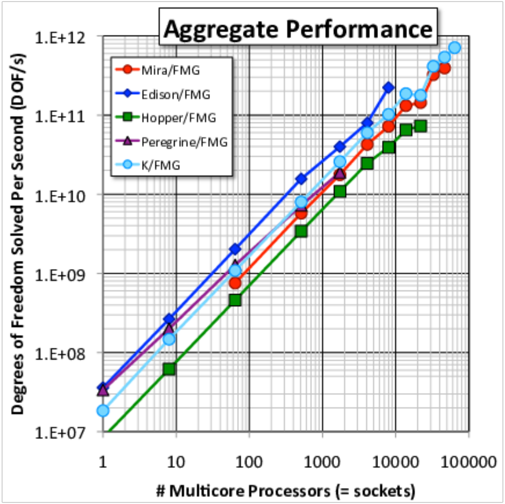
Edison (Aries, E5-2695v2)



Edison, SuperMUC, Titan

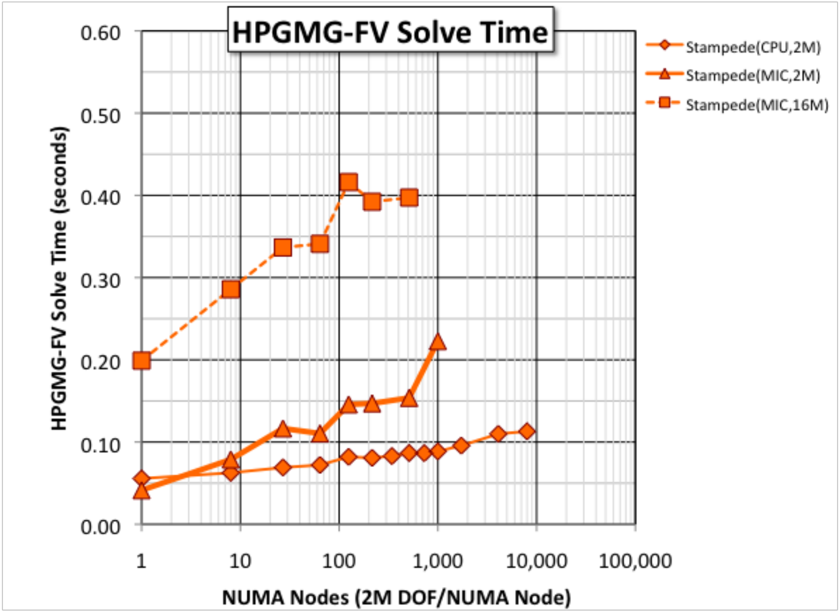


HPGMG distinguishes networks at 1M dofs/core



- Peregrine and Edison have identical node architecture
- Peregrine has 5:1 tapered IB, Edison has Aries dragonfly topology

MIC communication bottlenecks on Stampede



Hardware Arithmetic Intensity

Operation	Arithmetic Intensity (flops/B)
Sparse matrix-vector product	1/6
Dense matrix-vector product	1/4
Unassembled matrix-vector product, residual	$\gtrsim 8$

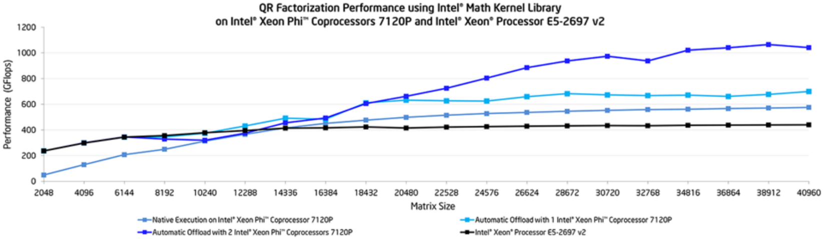
Processor	STREAM Triad (GB/s)	Peak (GF/s)	Balance (F/B)
E5-2680 8-core	38	173	4.5
E5-2695v2 12-core	45	230	5.2
E5-2699v3 18-core	60	660	11
Blue Gene/Q node	29.3	205	7
Kepler K20Xm	160	1310	8.2
Xeon Phi SE10P	161	1060	6.6
KNL (DRAM)	100	3000	30
KNL (MCDRAM)	500	3000	6

How much parallelism out of how much cache?

Processor	v width	threads	F/inst	latency	L1D	L1D/#par
Nehalem	2	1	2	5	32 KiB	1638 B
Sandy Bridge	4	2	2	5	32 KiB	819 B
Haswell	4	2	4	5	32 KiB	410 B
BG/P	2	1	2	6	32 KiB	1365 B
BG/Q	4	4	2	6	32 KiB	682 B
KNC	8	4	4	5	32 KiB	205 B
Tesla K20	32	*	2	10	64 KiB	102 B

- Most “fast” algorithms do about $O(n)$ flops on n data
- DGEMM and friends do $O(n^{3/2})$ flops on n data
- Exploitable parallelism limited by cache and register load/store
- L2/L3 performance highly variable between architectures

Where we are now: *QR* factorization with MKL on MIC



- Figure compares two CPU sockets (230W TDP) to one MIC (300W TDP plus host)
- Performance/Watt only breaks even at largest problem sizes
- Haswell-EP doubles performance within same power envelope
- $10^4 \times 10^4$ matrix takes 667 GFlops: about 2 seconds
- This is an $O(n^{3/2})$ operation on n data
- MIC cannot strong scale, no more energy efficient/cost effective
- “hard to program” versus “architecture ill-suited for problem”?

Outlook

- How can we measure versatility?
  - Opportunity cost of avoiding problems that “don’t scale”
- What is the impact of performance variability?
  - Allocation budgeting, coupling, load balancing
- We should strive to put ourselves out of business