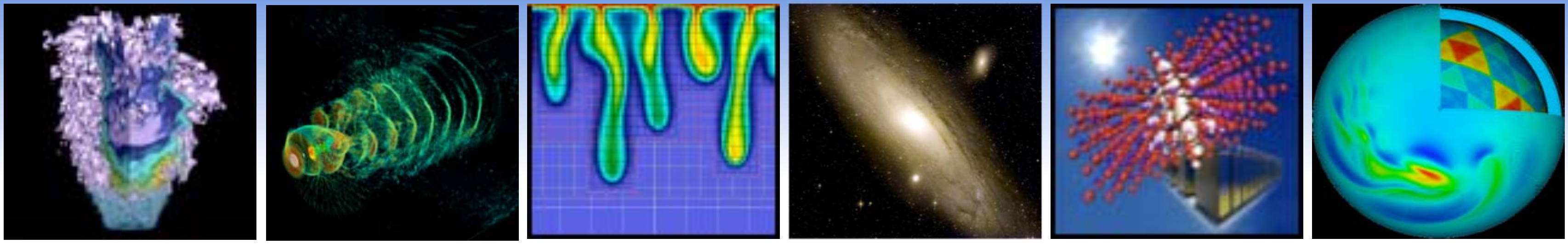


Paving the Road to Exascale

Kathy Yelick
Associate Laboratory Director for Computing Sciences and NERSC Division Director, Lawrence Berkeley National Laboratory

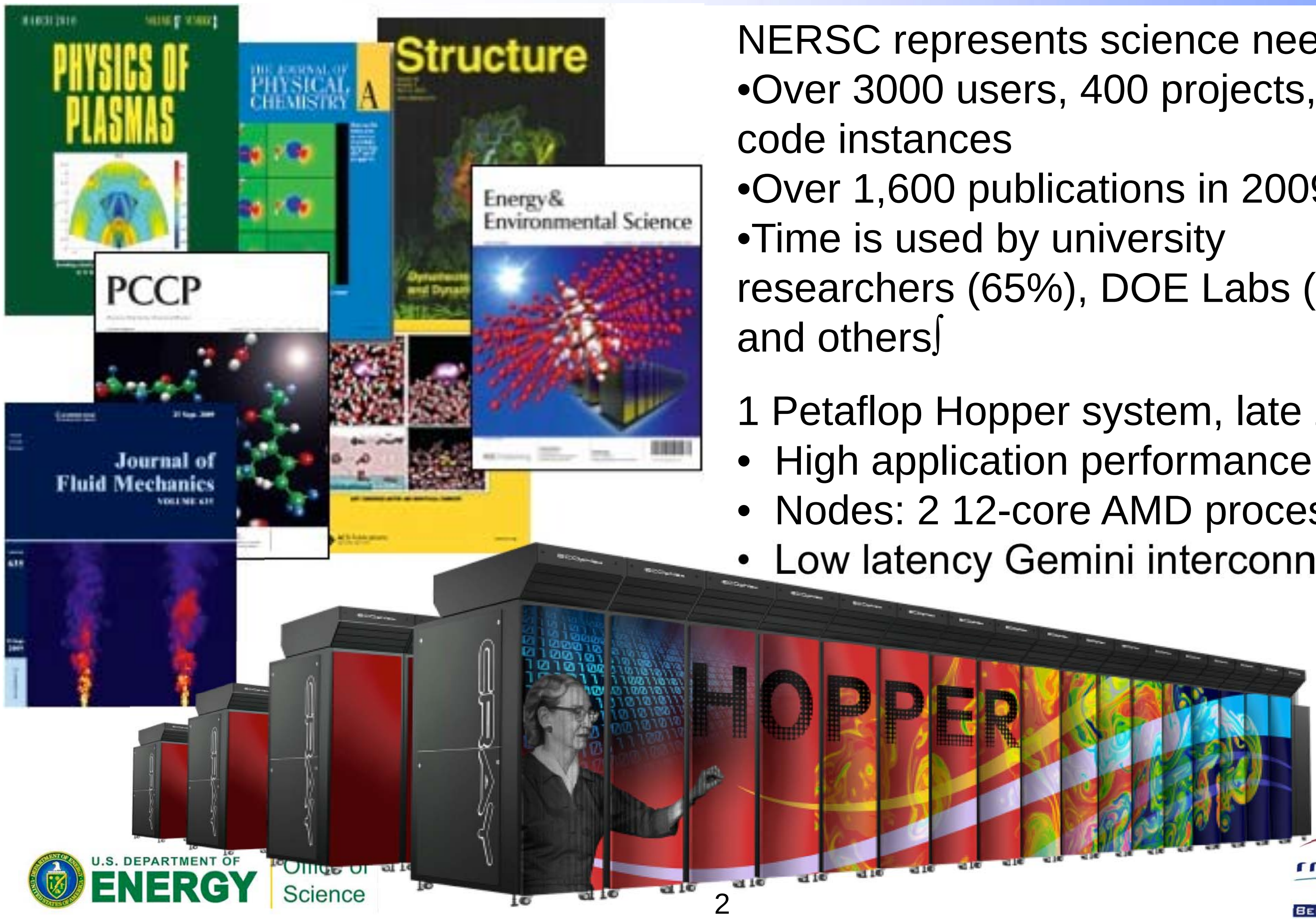
EECS Professor, UC Berkeley

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

BERKELEY LAB

NERSC Overview


PHYSICS OF PLASMAS
THE JOURNAL OF PHYSICAL CHEMISTRY A
Structure
Energy & Environmental Science
PCCP
Journal of Fluid Mechanics
HOPPER
CRAY

NERSC represents science needs

•Over 3000 users, 400 projects, 500 code instances

•Over 1,600 publications in 2009

•Time is used by university researchers (65%), DOE Labs (25%) and others∫

1 Petaflop Hopper system, late 2010

- High application performance
- Nodes: 2 12-core AMD processors
- Low latency Gemini interconnect

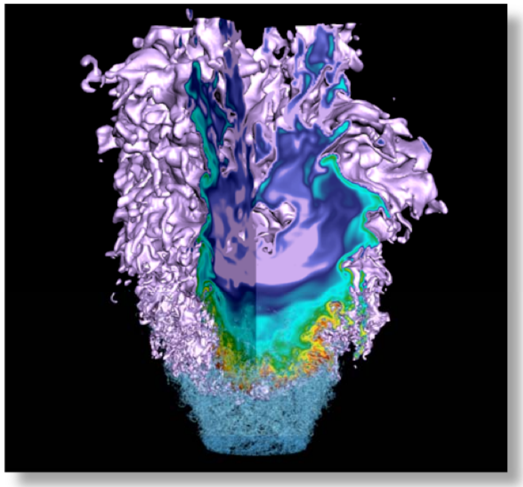
DOE Explores Cloud Computing

FERDINAN MAGELLANVS SVPERATIS
ANTARCTICI FRETI ANGVSTIIS CLARISS

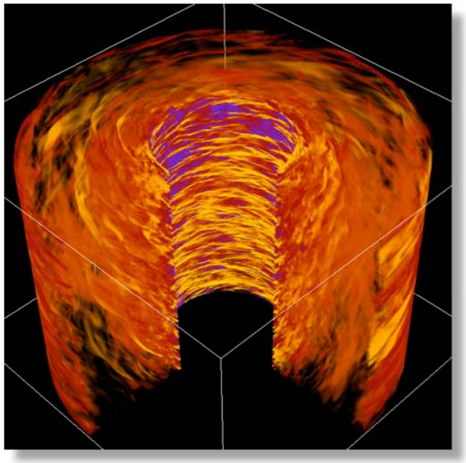
- **In spite of NERSC and other DOE centers**
  - **Many scientists still buy their own clusters**
  - **Not efficient for energy or operations**
  - **Clouds provide centralized resources for diverse workloads, including "private virtual clusters"**
- **Magellan is a "Science Cloud" Testbed for DOE**
  - **Installed in early 2010; iDataplex cluster**
- **Cloud questions to explore on Magellan:**
  - **Can a cloud serve DOE's mid-range computing needs?**
  - **What features (hardware and software) are needed in a Science Cloud?**
  - **What part of the workload benefits from clouds?**
  - **Is a Science Cloud from commercial clouds which serve primarily independent serial jobs?**

NATIONAL ENERGY RESEARCH SCIENTIFIC COMPUTING CENTER

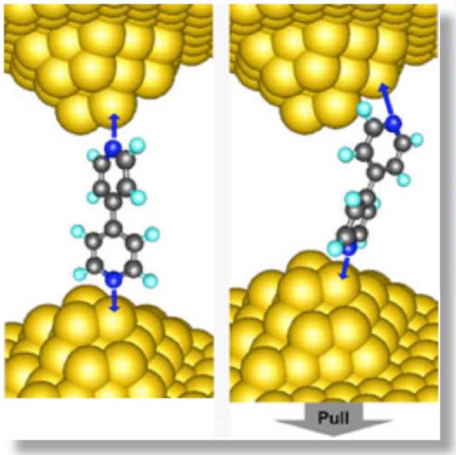
Science at NERSC



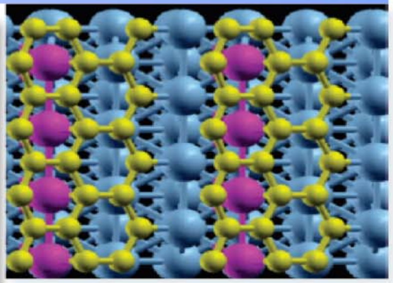
Combustion: New algorithms (AMR) coupled to experiments



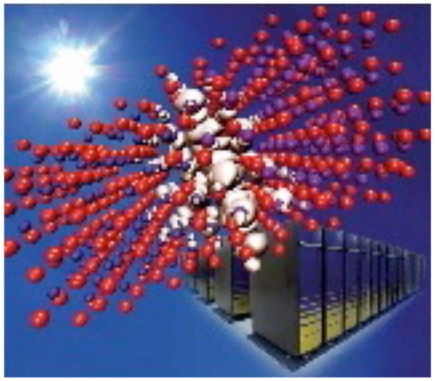
Fusion: Simulations of Fusion devices at ITER scale



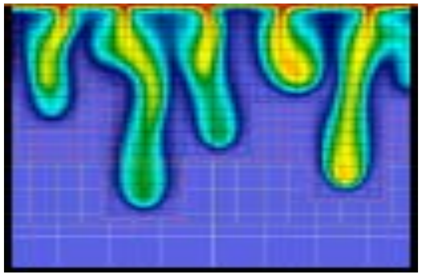
Nano devices: New single molecule switching element



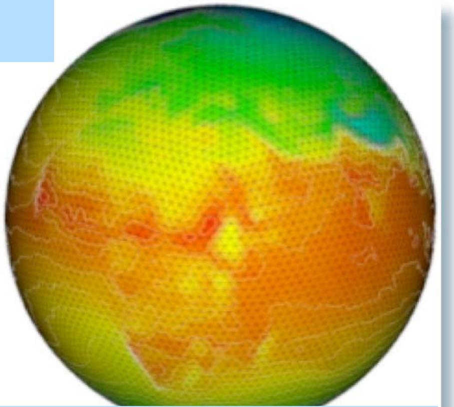
Energy storage: Catalysis for improved batteries and fuel cells



Materials: For solar panels and other applications.



Capture & Sequestration: EFRCs



Climate modeling: Work with users on scalability of cloud-resolving models

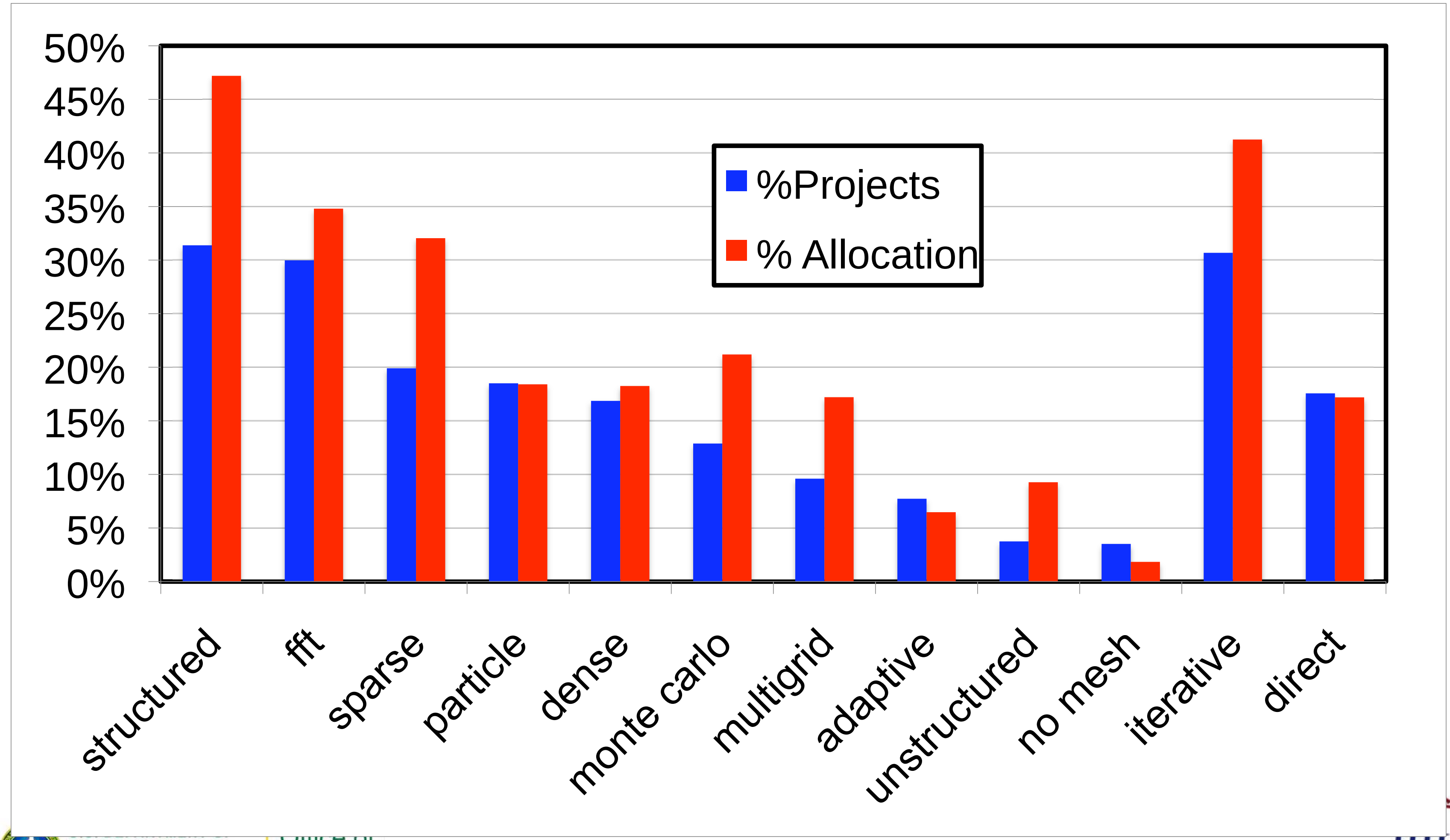
Algorithm Diversity

Science areas	*Dense linear algebra*	*Sparse linear algebra*	*Spectral Methods (FFT)s*	*Particle Methods*	*Structured Grids*	*Unstructured or AMR Grids*
Accelerator Science		X	X	X	X	X
Astrophysics	X	X	X	X	X	X
Chemistry	X	X	X	X		
Climate			X		X	X
Combustion					X	X
Fusion	X	X		X	X	X
Lattice Gauge		X	X	X	X	
Material Science	X		X	X	X	

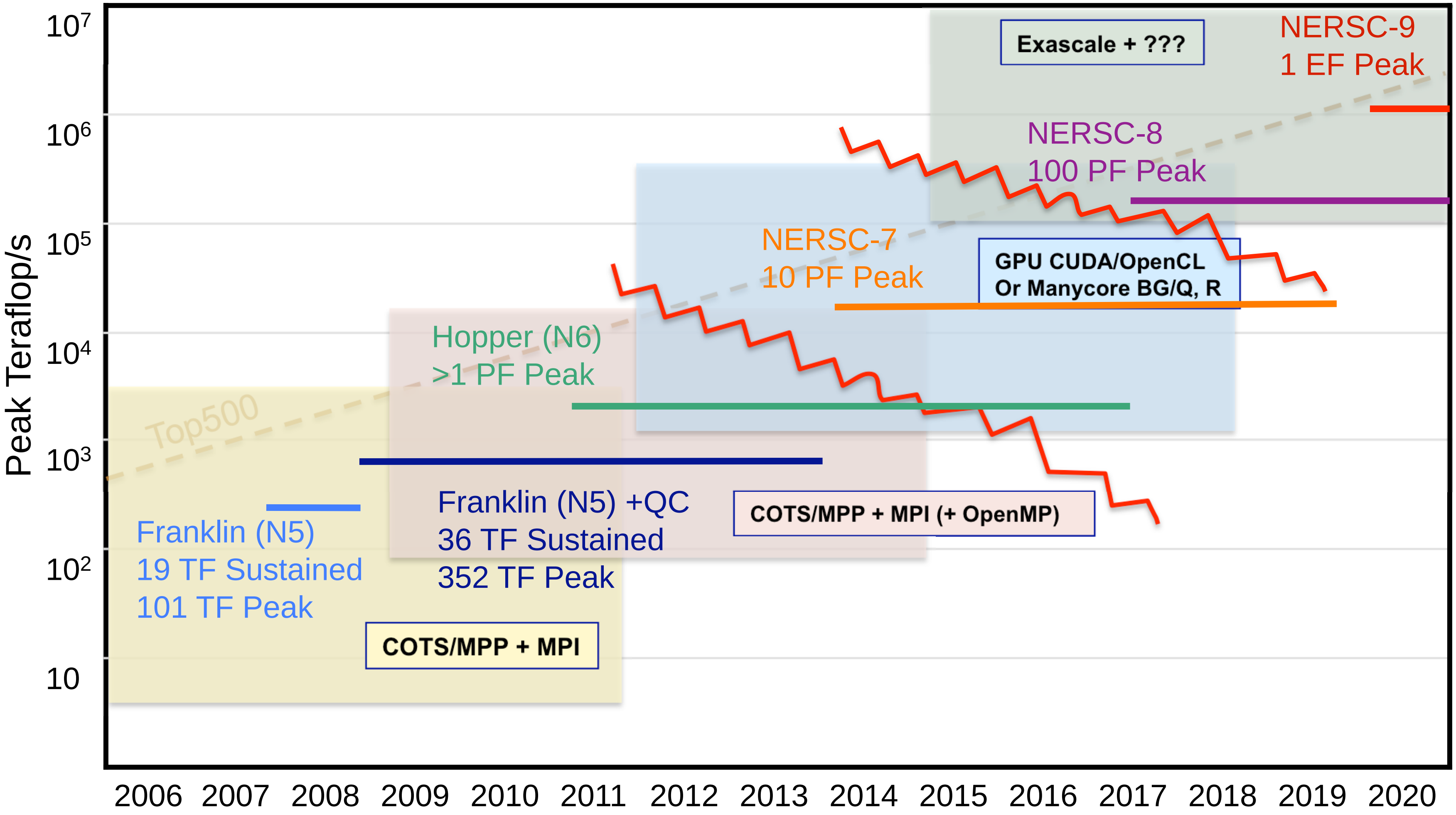
NERSC Qualitative In-Depth Analysis of Methods by Science Area

Numerical Methods at NERSC

- Quantitative (but not so deep) measure of algorithms classes
- Based on hours allocated to a project that the PI claims uses the method



NERSC Interest in Exascale

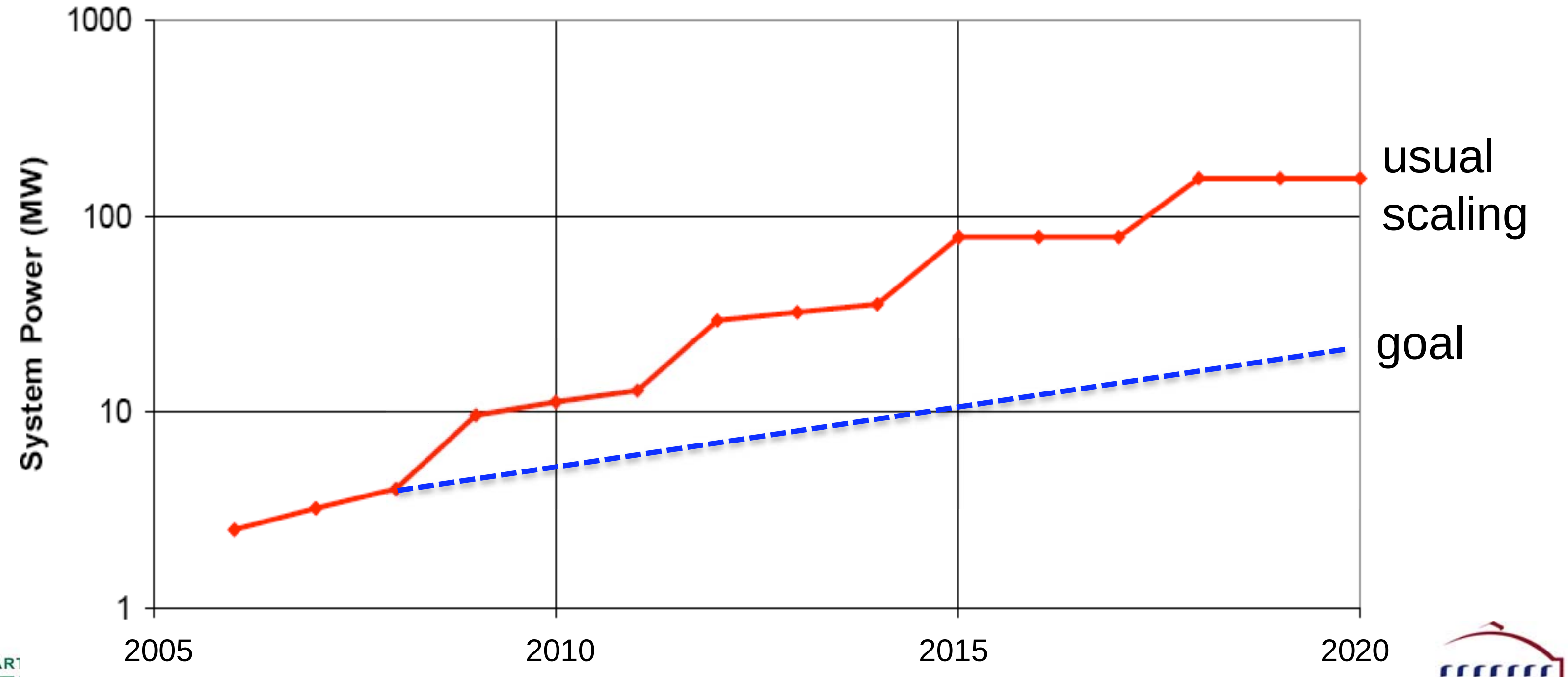


Danger: dragging users into a local optimum for programming

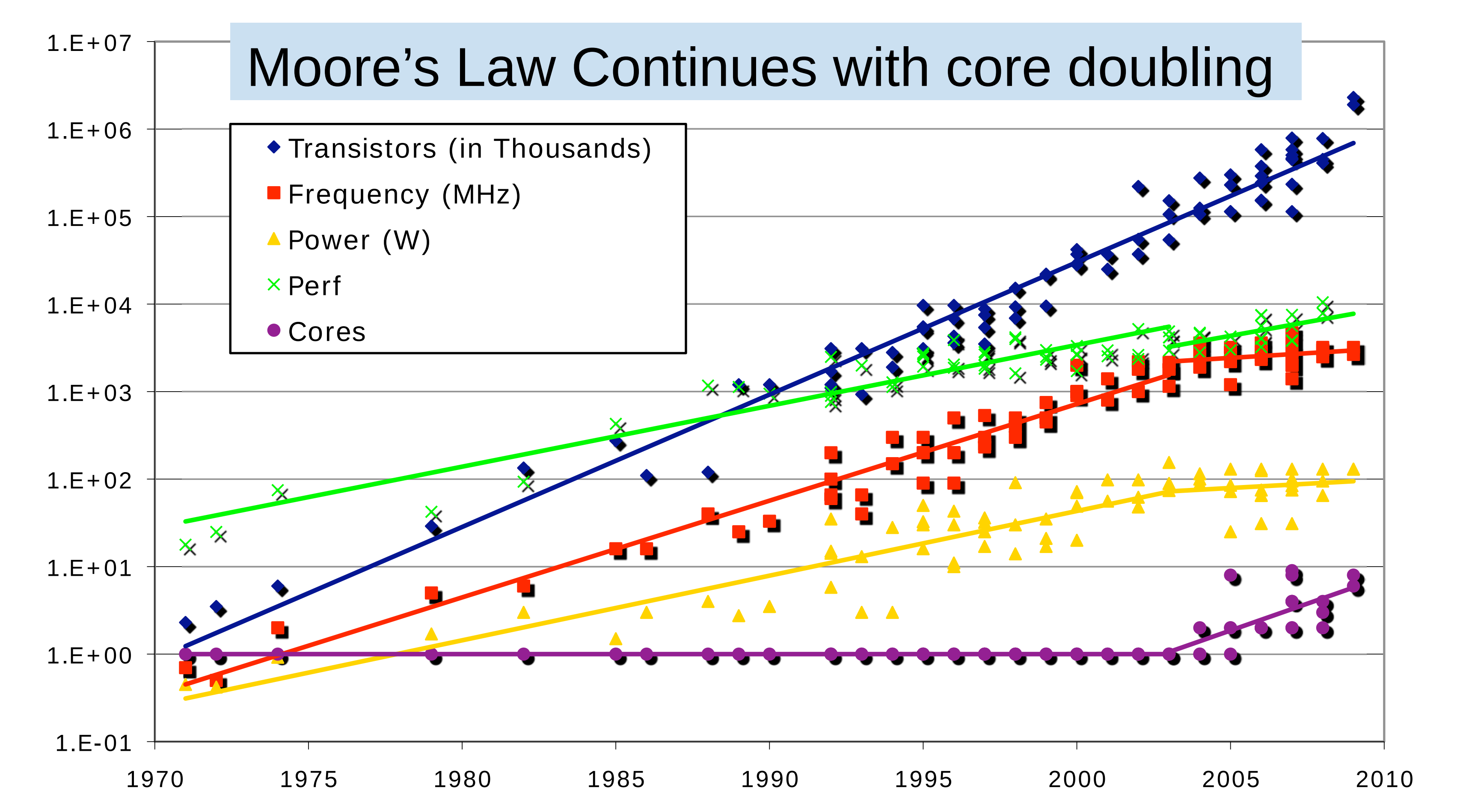
Exascale is really about Energy Efficient Computing

At $1M per MW, energy costs are substantial

- 1 petaflop in 2010 will use 3 MW
- 1 exaflop in 2018 possible in 200 MW with “usual” scaling
- 1 exaflop in 2018 at 20 MW is DOE target



Performance Has Also Slowed, Along with Power

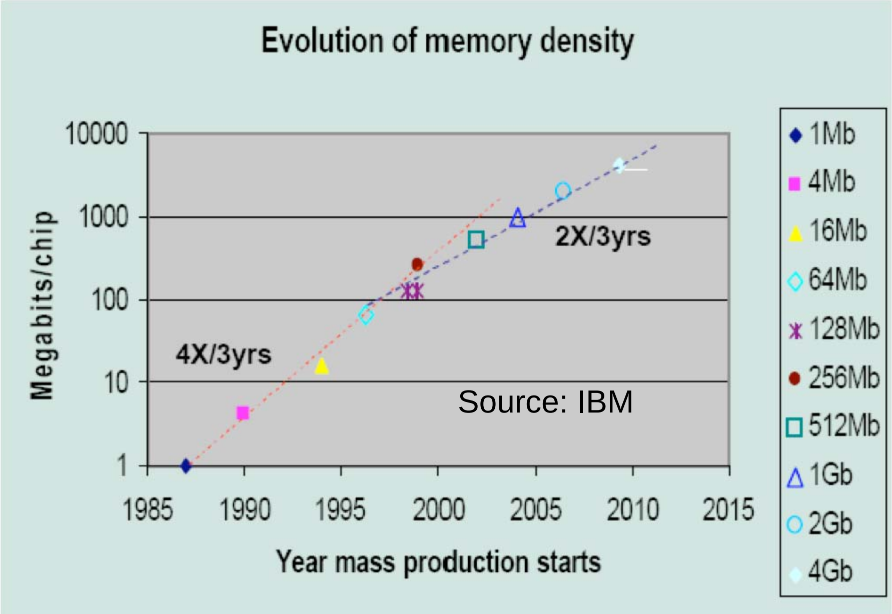


Data from Kunle Olukotun, Lance Hammond, Herb Sutter, Burton Smith, Chris Batten, and Krste Asanoviç

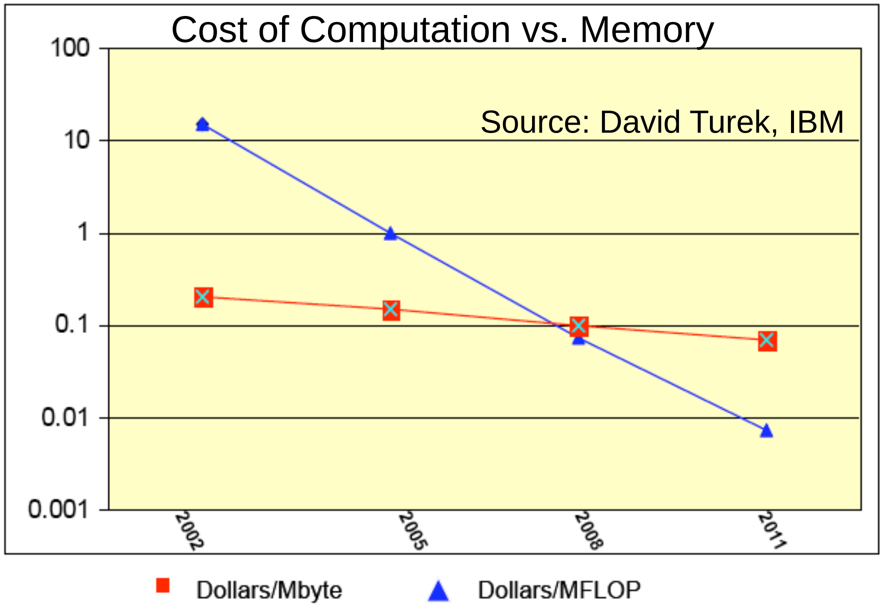
Memory is Not Keeping Pace

Technology trends against a constant or increasing memory per core

- Memory density is doubling every three years; processor logic is every two
- Storage costs (dollars/Mbyte) are dropping gradually compared to logic costs



Source: IBM



Source: David Turek, IBM

The cost to sense, collect, generate and calculate data is declining much faster than the cost to access, manage and store it

Question: *Can you double concurrency without doubling memory?*

The Challenge

- ***Power is the leading design constraint in HPC system design***

- ***How to get build an exascale system without building a nuclear power plant next to my HPC center?***

- ***How can you assure the systems will be balanced for a reasonable science workload?***

- ***How do you make it "programmable?"***

System Balance

- **If you pay 5% more to double the FPUs and get 10% improvement, it’s a win (despite lowering your % of peak performance)**

- **If you pay 2x more on memory BW (power or cost) and get 35% more performance, then it’s a net loss (even though % peak looks better)**

- ***Real example: we can give up ALL of the flops to improve memory bandwidth by 20% on the 2018 system***

- **We have a fixed budget**
  - **Sustained to peak FLOP rate is *wrong* metric if FLOPs are cheap**
  - **Balance involves balancing your checkbook & balancing your power budget**
  - **Requires a application co-design make the right trade-offs**

Anticipating and Influencing the Future

Hardware Design

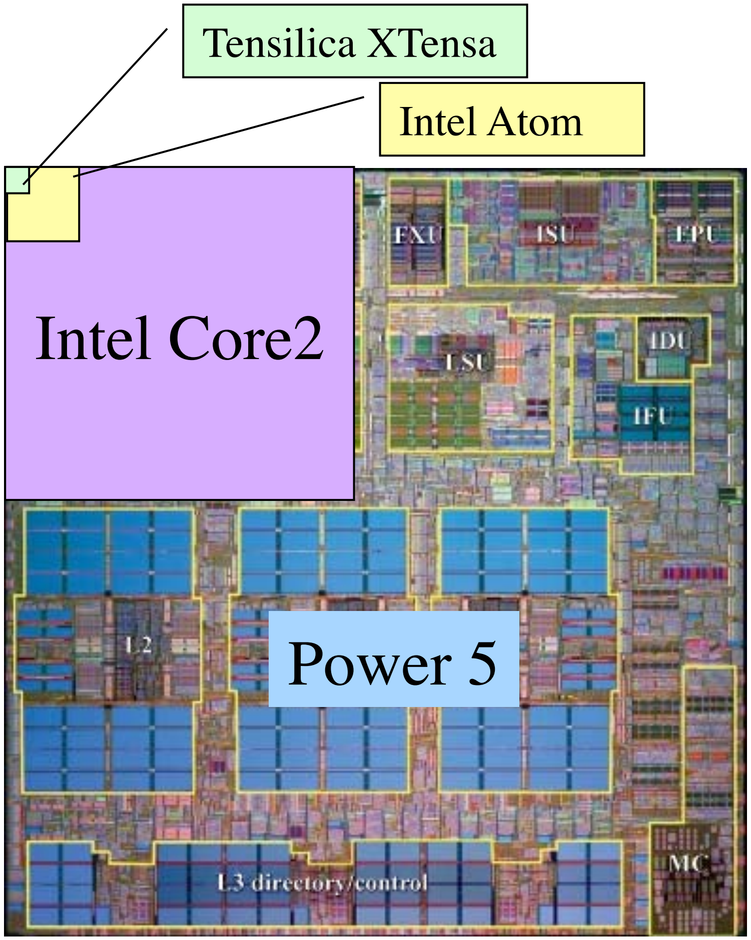
Architecture Paths to Exascale

- Leading Technology Paths *(Swim Lanes)*
  - ~~Multicore: Maintain complex cores, and replicate (x86 and Power7)~~
  - Manycore/Embedded: Use many simpler, low power cores from embedded (BlueGene)
  - GPU/Accelerator*:* Use highly specialized processors from gaming space (NVidia Fermi, Cell)

- Risks in Swim Lane selection
  - Select too soon: users cannot follow
  - Select too late*:* fall behind performance curve
  - Select incorrectly: Subject users to multiple disruptive changes

- Users must be deeply engaged in this process
  - Cannot leave this up to vendors alone

Manycore/Embedded Swim Lane

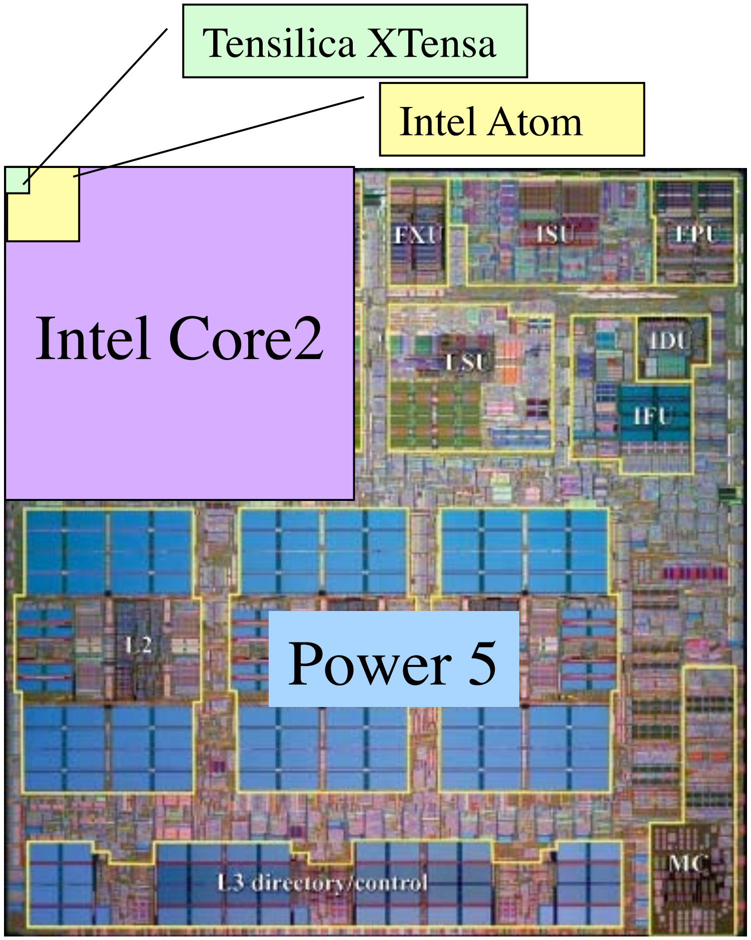


- Cubic power improvement with lower clock rate due to V^2F
- Slower clock rates enable use of simpler cores
- Simpler cores use less area (lower leakage) and reduce cost
- Tailor design to application to REDUCE WASTE

This is how iPhones and MP3 players are designed to maximize battery life and minimize cost

Slide by John Shalf,
Green Flash Project PI

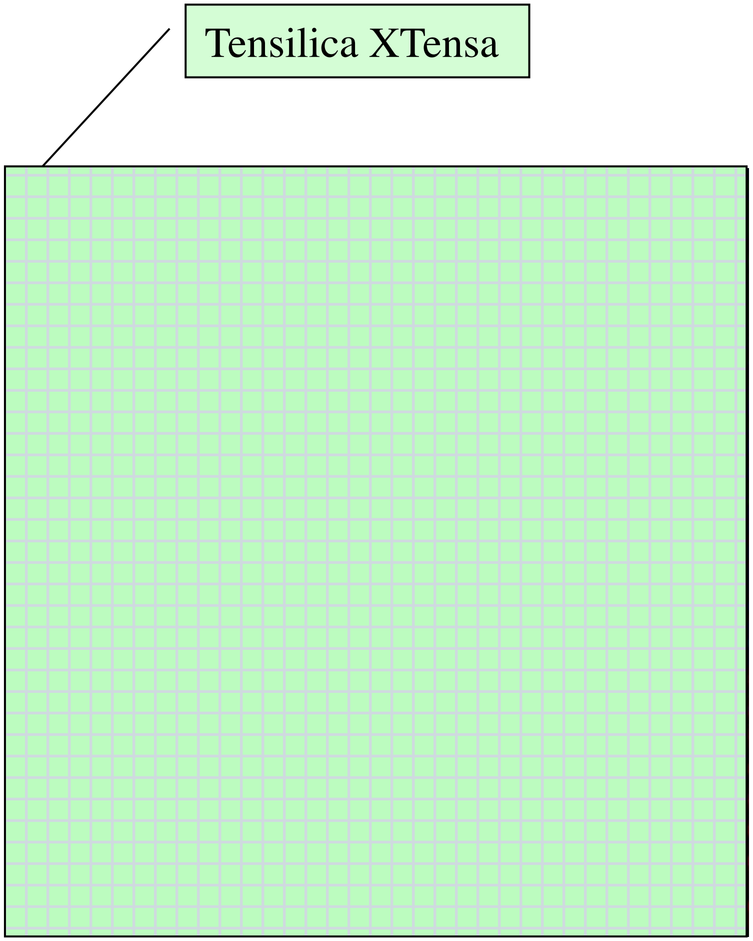
Manycore/Embedded Swim Lane



- **Power5 (server)**
  - **120W@1900MHz**
  - **Baseline**
- **Intel Core2 sc (laptop) :**
  - **15W@1000MHz**
  - ***4x more FLOPs/watt than baseline***
- **Intel Atom (handhelds)**
  - **0.625W@800MHz**
  - **80x more**
- **Tensilica XTensa DP (Moto Razor) :**
  - **0.09W@600MHz**
  - **400x more (80x-100x sustained)**

Slide by John Shalf,
Green Flash Project PI

Manycore/Embedded Swim Lane



- **Power5 (server)**
  - **120W@1900MHz**
  - **Baseline**
- **Intel Core2 sc (laptop) :**
  - **15W@1000MHz**
  - ***4x more FLOPs/watt than baseline***
- **Intel Atom (handhelds)**
  - **0.625W@800MHz**
  - **80x more**
- **Tensilica XTensa DP (Moto Razor) :**
  - **0.09W@600MHz**
  - **400x more (80x-100x sustained)**

Even if each simple core is 1/4th as computationally efficient as complex core, you can fit hundreds of them on a single chip and still be 100x more power efficient.

Slide by John Shalf,
Green Flash Project PI

Technology Investment Trends

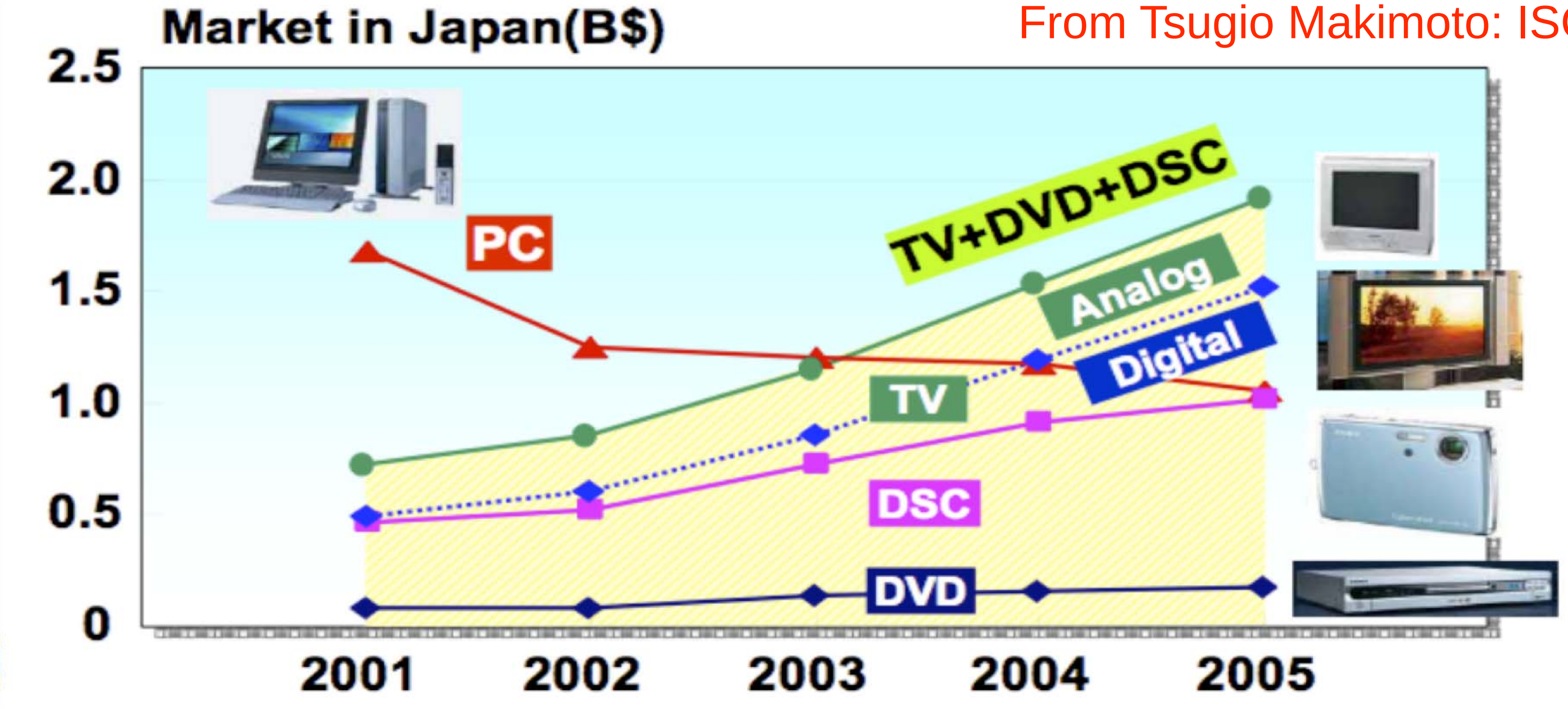
1990s: Computing R&D dominated by desktop/COTS

- Learned to use COTS technology for HPC

2010s: Computing R&D moving to consumer electronics

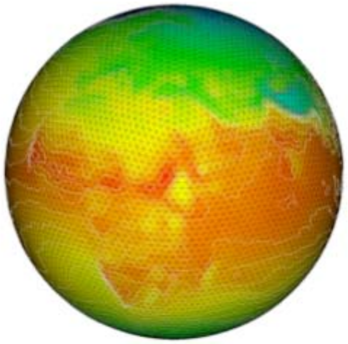
- Need to leverage embedded/consumer technology for HPC

From Tsugio Makimoto: ISC2006

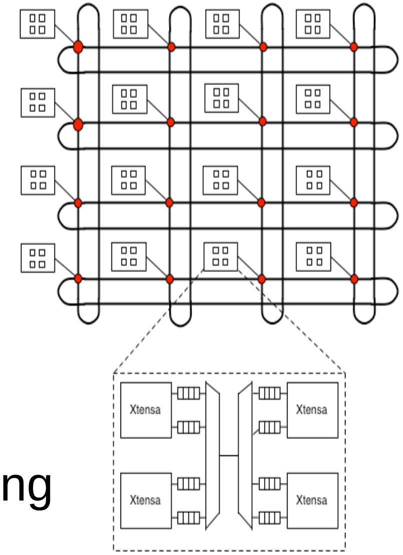


Co-Design in the Green Flash Project

- **Demonstrated during SC '09**
- **CSU atmospheric model ported to low-power core design**
  - **Dual Core Tensilica processors running atmospheric model at 25MHz**
  - **MPI Routines ported to custom Tensilica Interconnect**
- **Memory and processor Stats available for performance analysis**
- **Emulation performance advantage**
  - **250x Speedup over merely function software simulator**
- ***Actual code running - not representative benchmark***



Icosahedral mesh for algorithm scaling



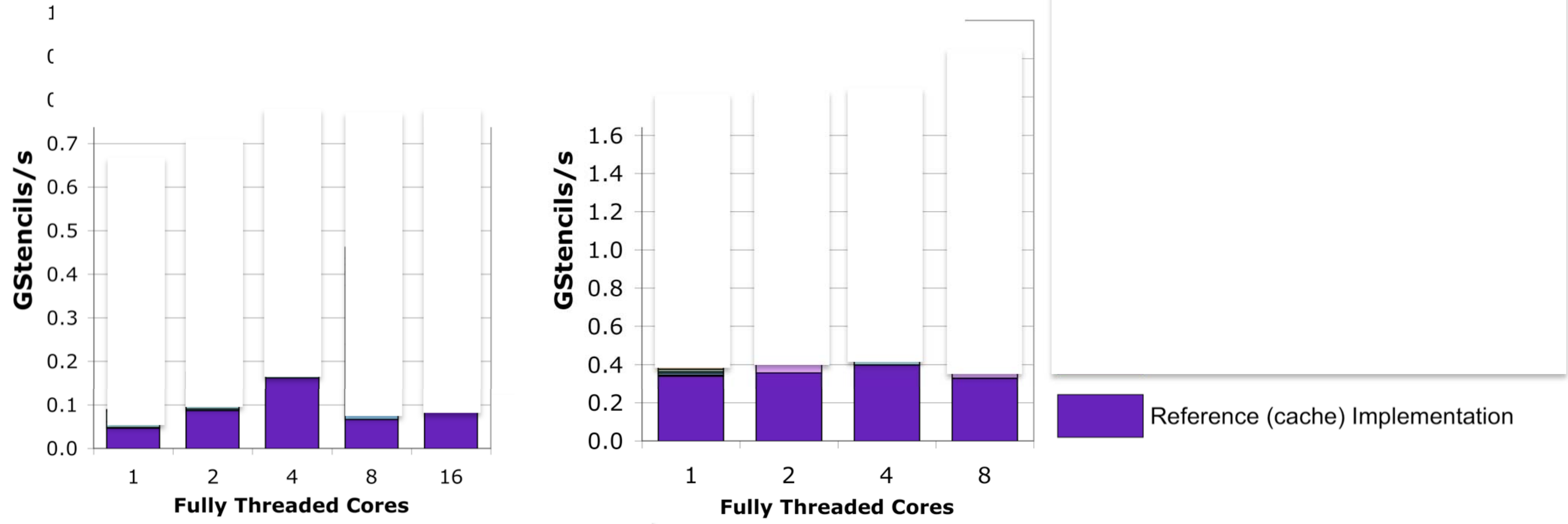
John Shalf, Dave Donofrio, Lenny Oliker, Michael Wehner, Marghoob Mohiyuddin, Shoaib Kamil

Autotuning: Write Code Generators for Nodes

Nearest-neighbor 7point stencil on a 3D array

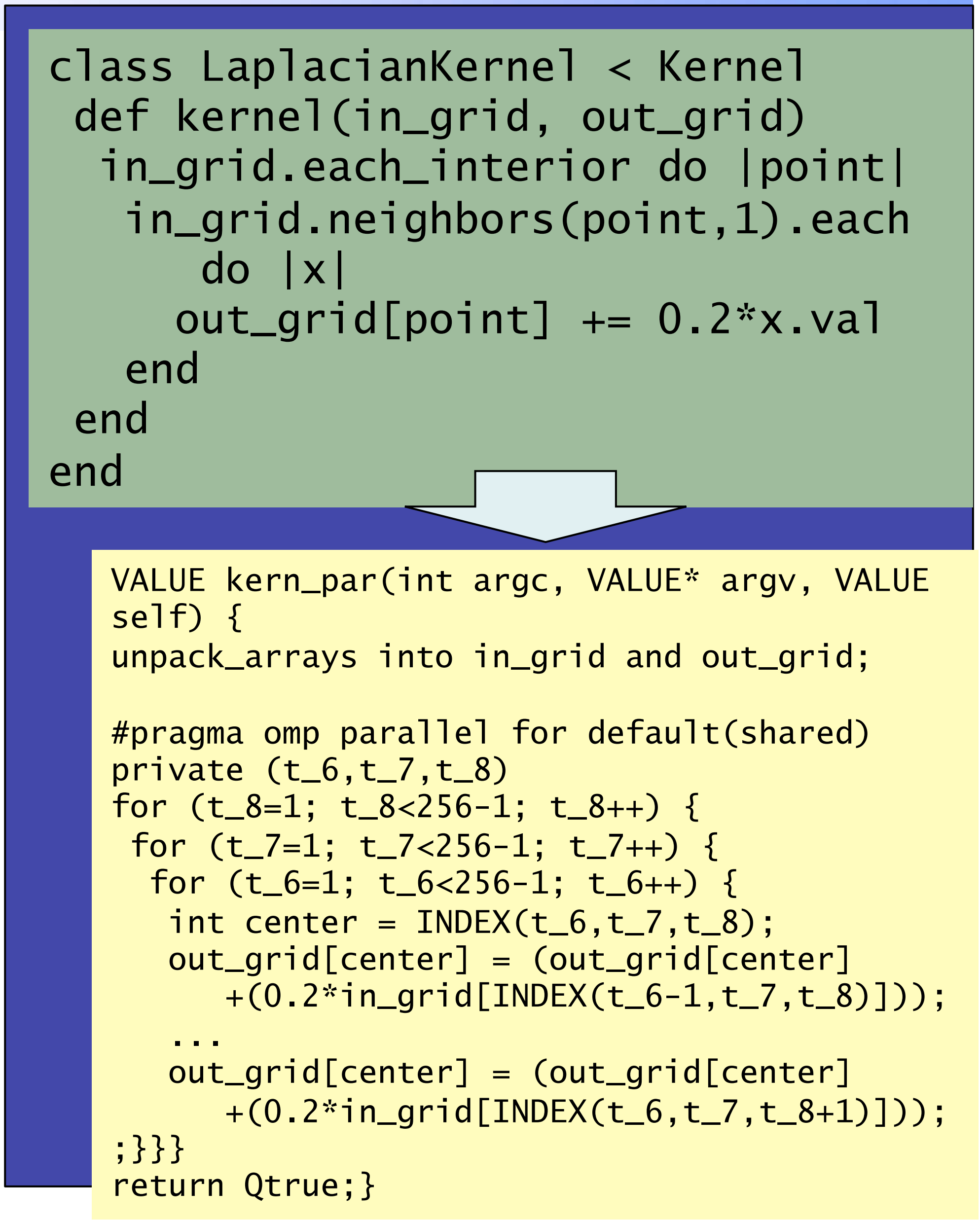
Use Autotuning!

Write code generators and let computers do tuning



Example pattern-specific compiler: Structured grid in Ruby

- **Ruby class encapsulates SG pattern**
  - **body of anonymous lambda specifies filter function**
- **Code generator produces OpenMP**
  - **~1000-2000x faster than Ruby**
  - **Minimal per-call runtime overhead**



```
class LaplacianKernel < Kernel
 def kernel(in_grid, out_grid)
  in_grid.each_interior do |point|
   in_grid.neighbors(point,1).each
      do |x|
     out_grid[point] += 0.2*x.val
   end
 end
end
```

```
VALUE kern_par(int argc, VALUE* argv, VALUE
self) {
unpack_arrays into in_grid and out_grid;

#pragma omp parallel for default(shared)
private (t_6,t_7,t_8)
for (t_8=1; t_8<256-1; t_8++) {
 for (t_7=1; t_7<256-1; t_7++) {
  for (t_6=1; t_6<256-1; t_6++) {
   int center = INDEX(t_6,t_7,t_8);
   out_grid[center] = (out_grid[center]
      +(0.2*in_grid[INDEX(t_6-1,t_7,t_8)]));
   ...
   out_grid[center] = (out_grid[center]
      +(0.2*in_grid[INDEX(t_6,t_7,t_8+1)]));
;}}}
return Qtrue;}
```

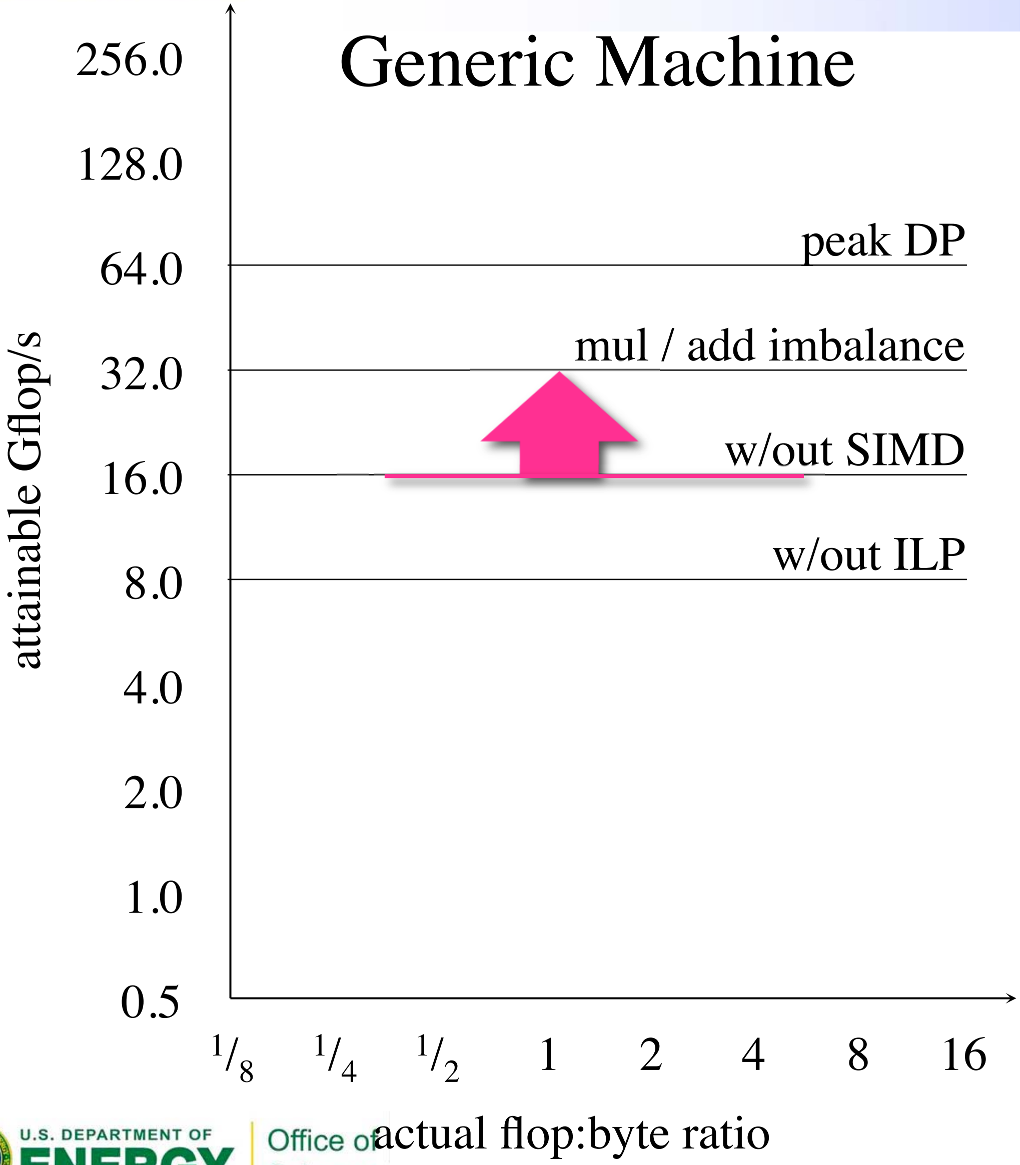
Shoaib Kamil, Armando Fox, John Shalf,

Understand your machine limits

The “roofline” model

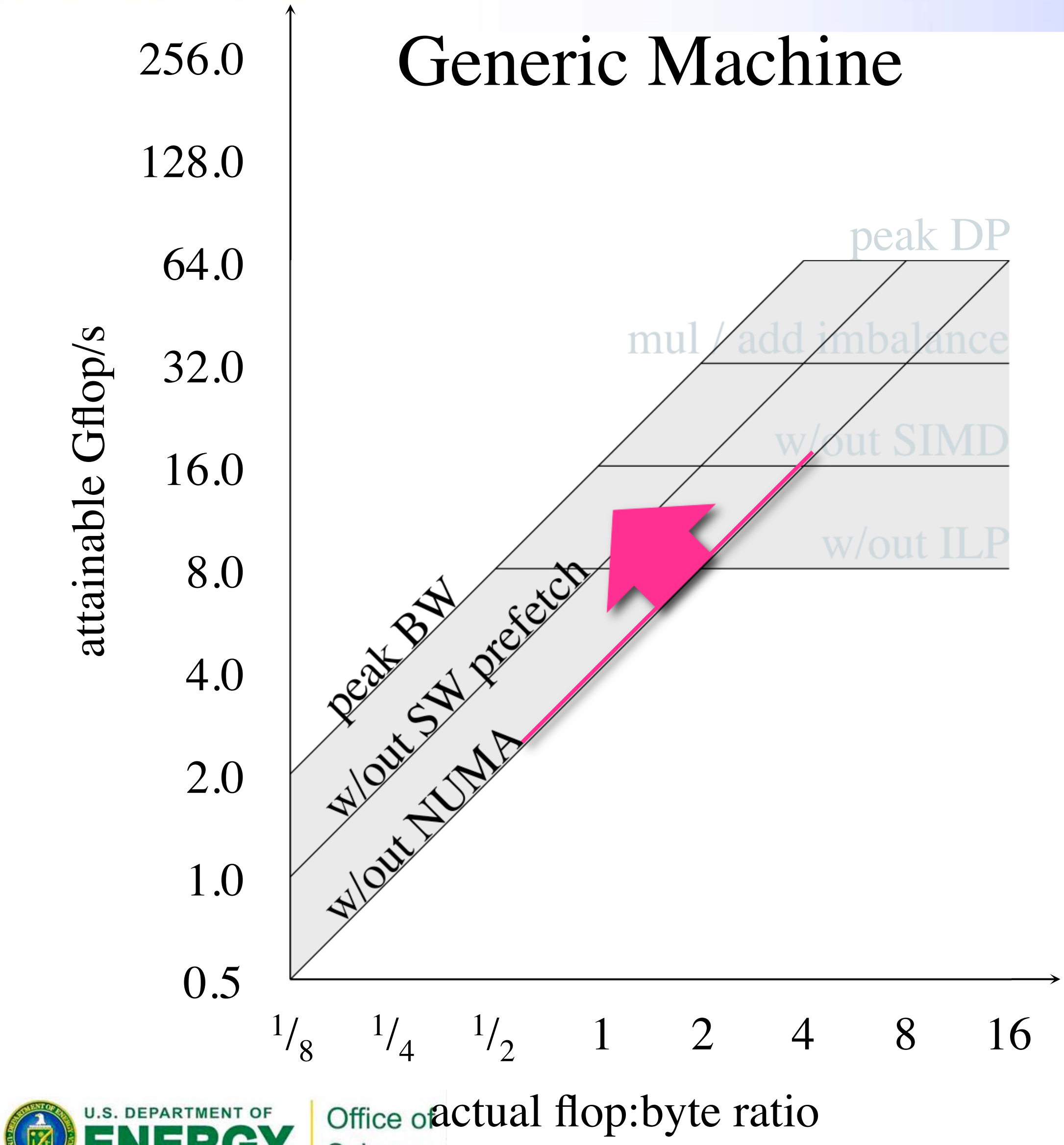
S. Williams, D. Patterson, L. Oliker, J. Shalf, K. Yelick

The Roofline Performance Model



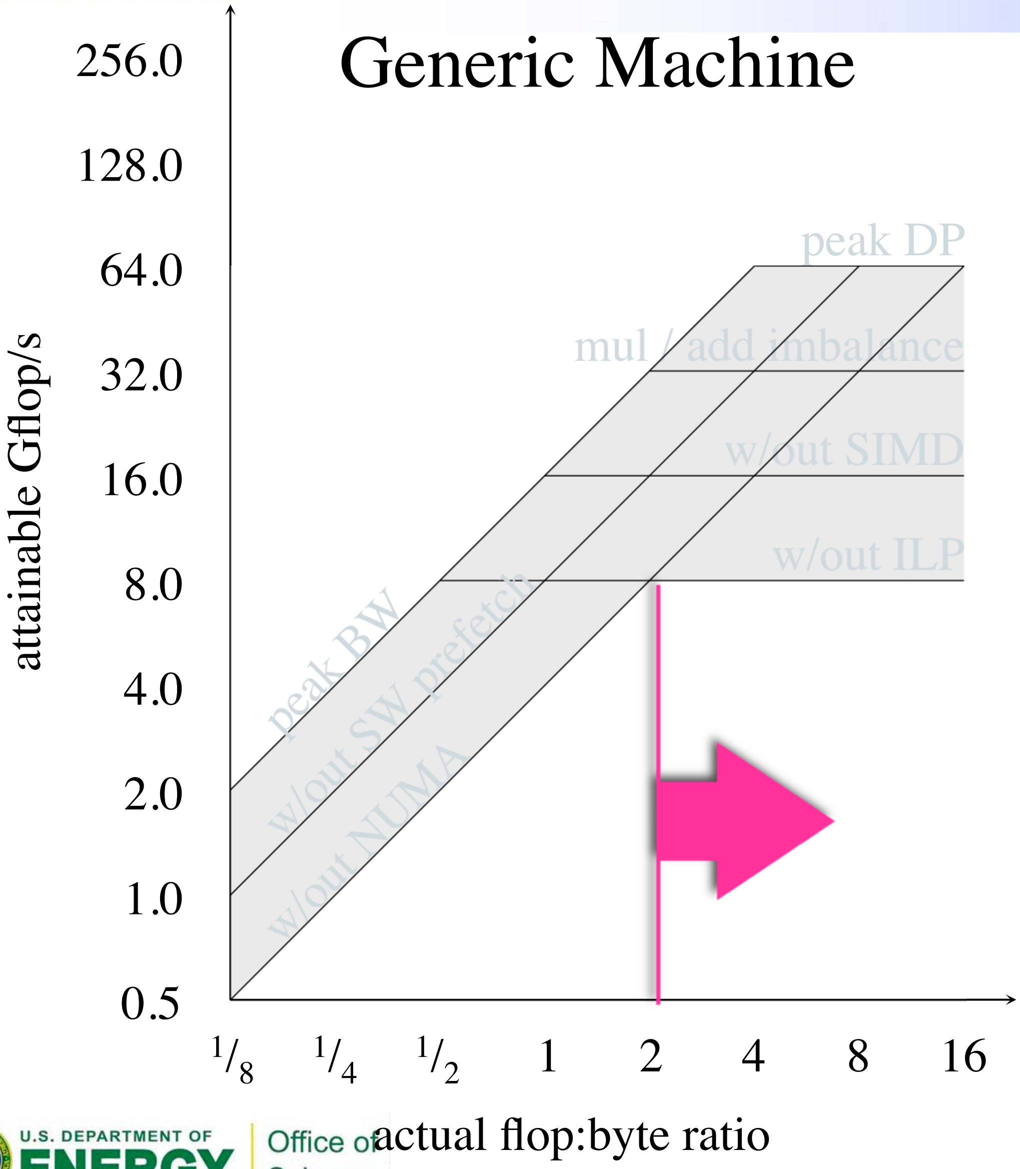
- **The top of the roof is determined by peak computation rate (Double Precision floating point, DP for these algorithms)**
- **The instruction mix, lack of SIMD operations, ILP or failure to use other features of peak will lower attainable**

The Roofline Performance Model



- The sloped part of the roof is determined by peak DRAM bandwidth (STREAM)
- Lack of proper prefetch, ignoring NUMA, or other things will reduce attainable bandwidth

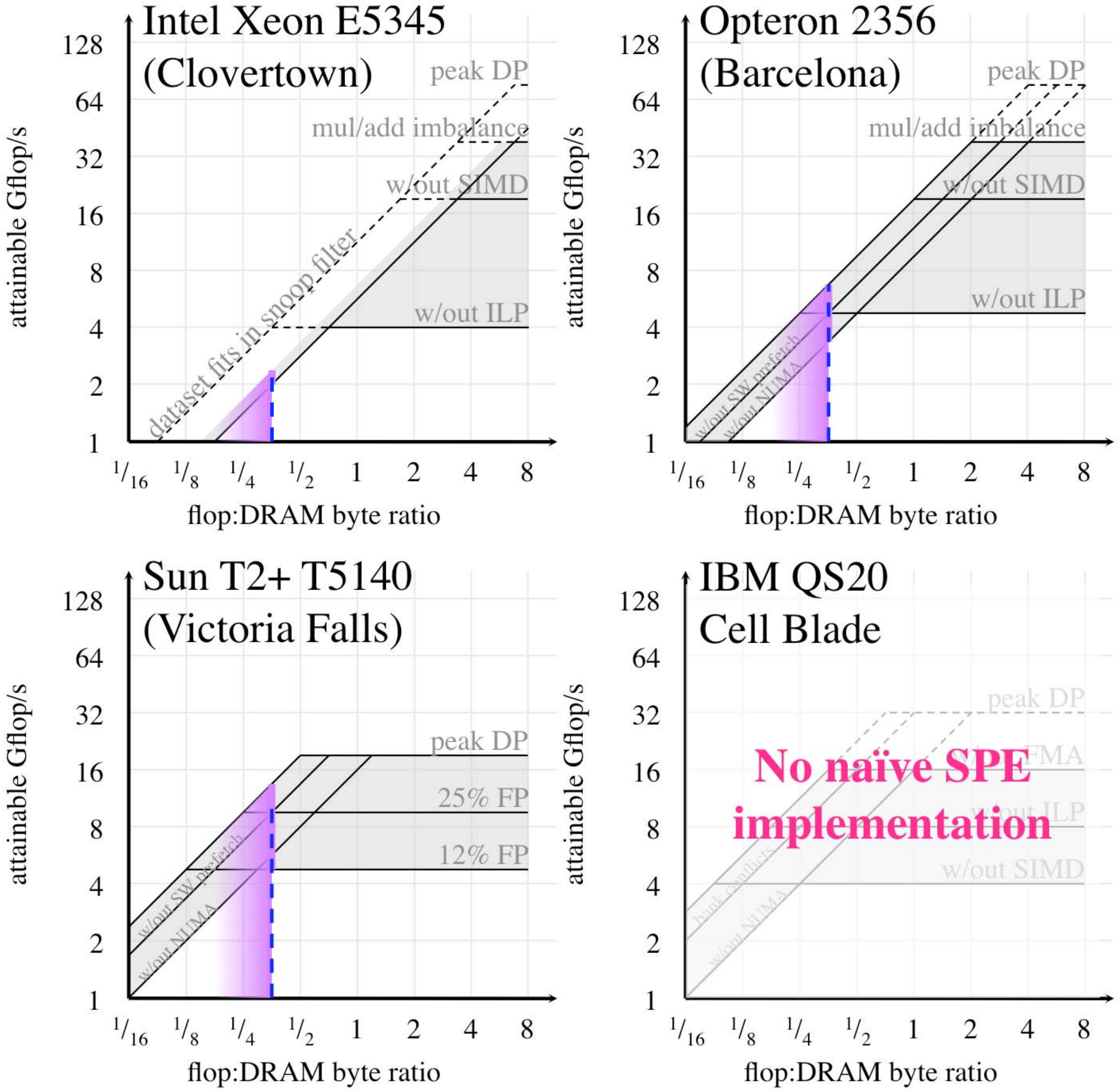
The Roofline Performance Model



- Locations of posts in the building are determined by algorithmic intensity
- Will vary across algorithms and with bandwidth-reducing optimizations, such as better cache re-use (tiling), compression techniques

Roofline model for Stencil

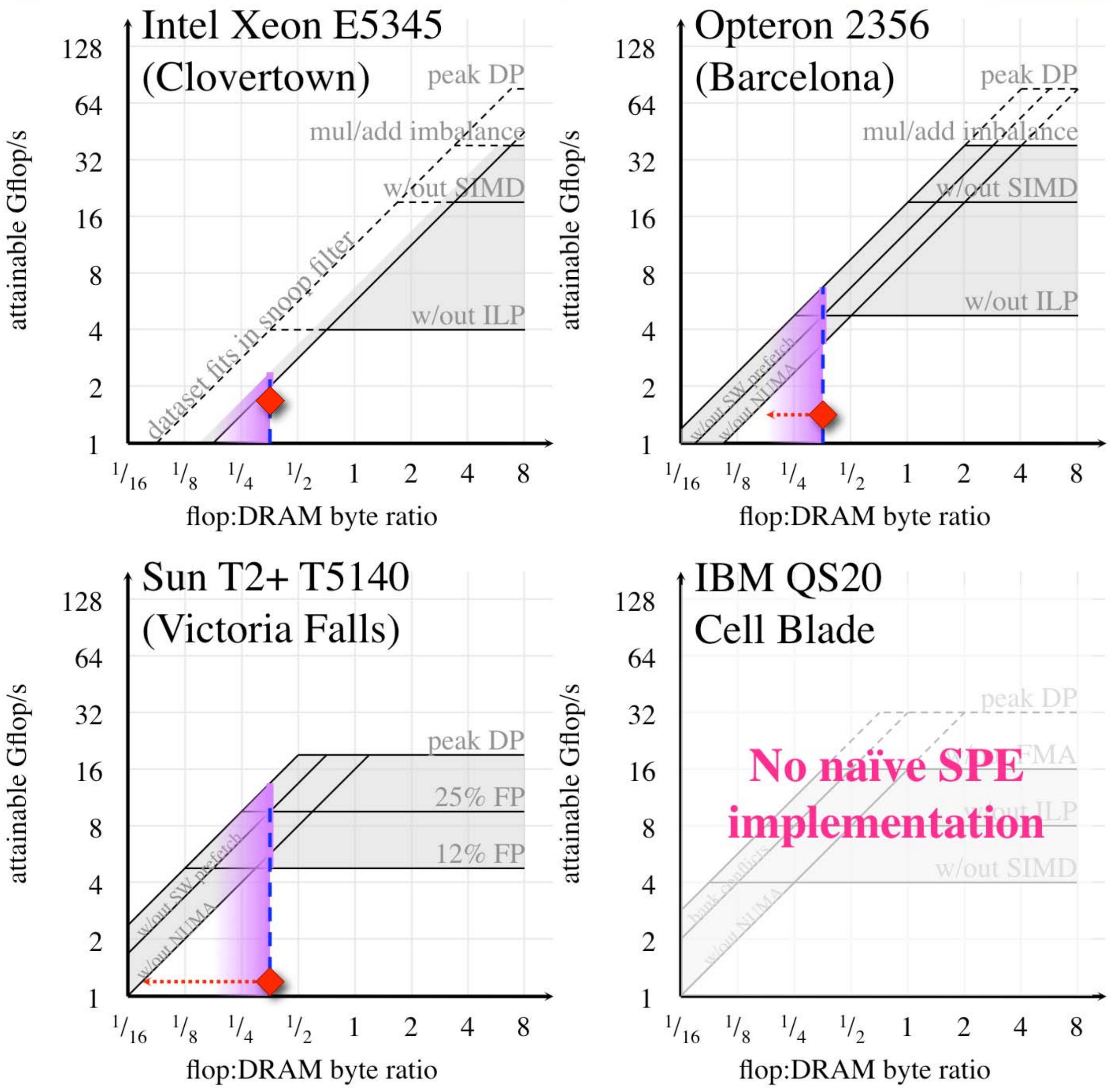
(out-of-the-box code)



- Large datasets
- 2 unit stride streams
- No NUMA
- Little ILP
- No DLP
- Far more adds than multiplies (imbalance)
- Ideal flop:byte ratio $^1/_3$
- High locality requirements
- Capacity and conflict misses will severely impair flop:byte ratio

Roofline model for Stencil

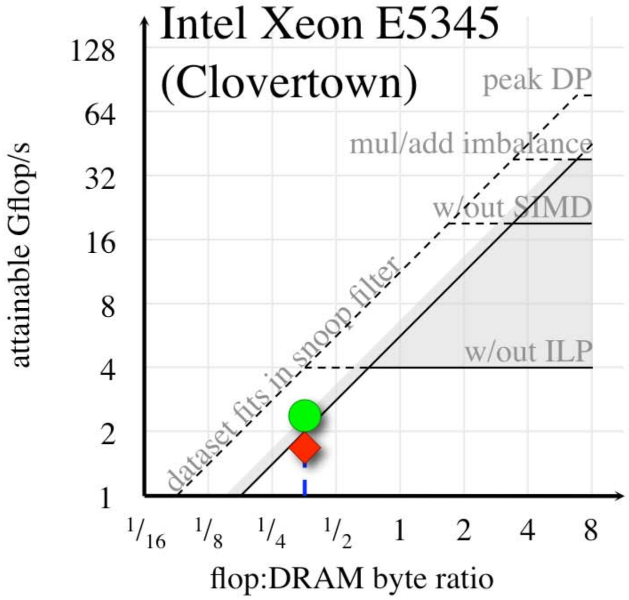
(out-of-the-box code)

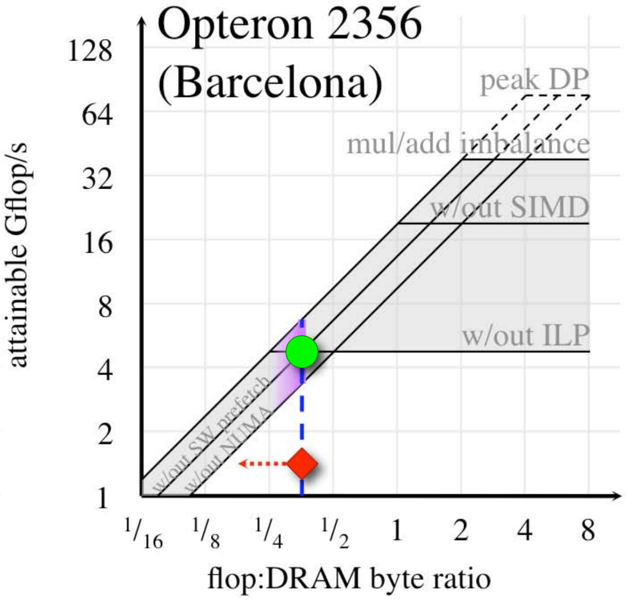


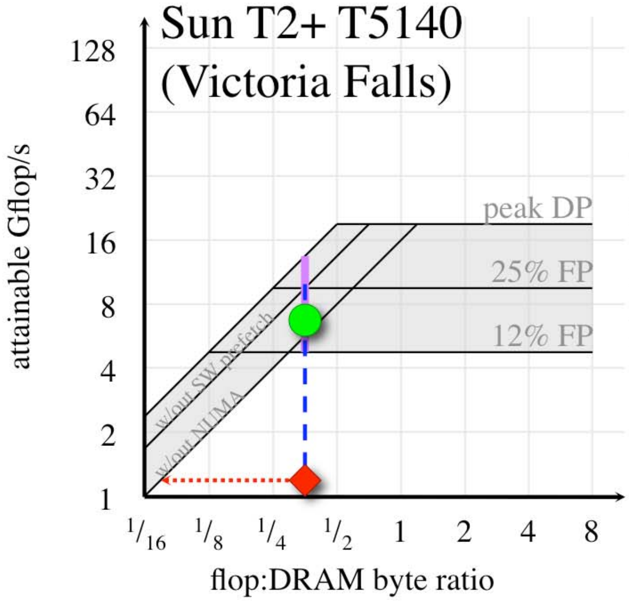
- Large datasets
- 2 unit stride streams
- No NUMA
- Little ILP
- No DLP
- Far more adds than multiplies (imbalance)
- Ideal flop:byte ratio $^1/_3$
- High locality requirements
- Capacity and conflict misses will severely impair flop:byte ratio

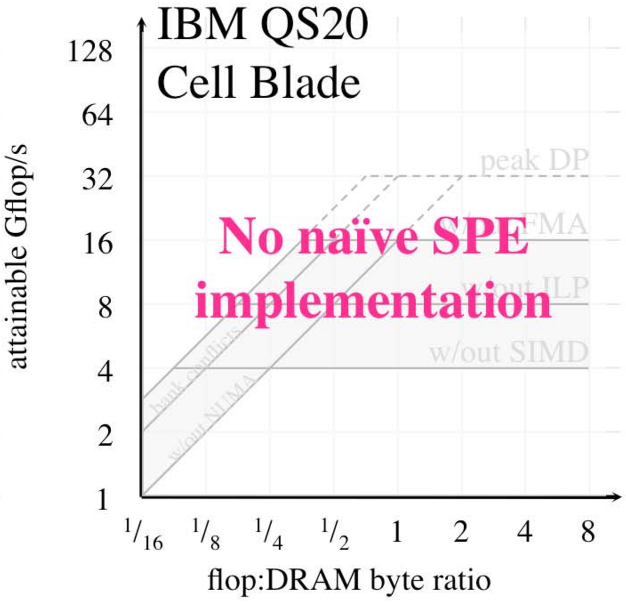
Roofline model for Stencil

(NUMA, cache blocking, unrolling, prefetch, …)





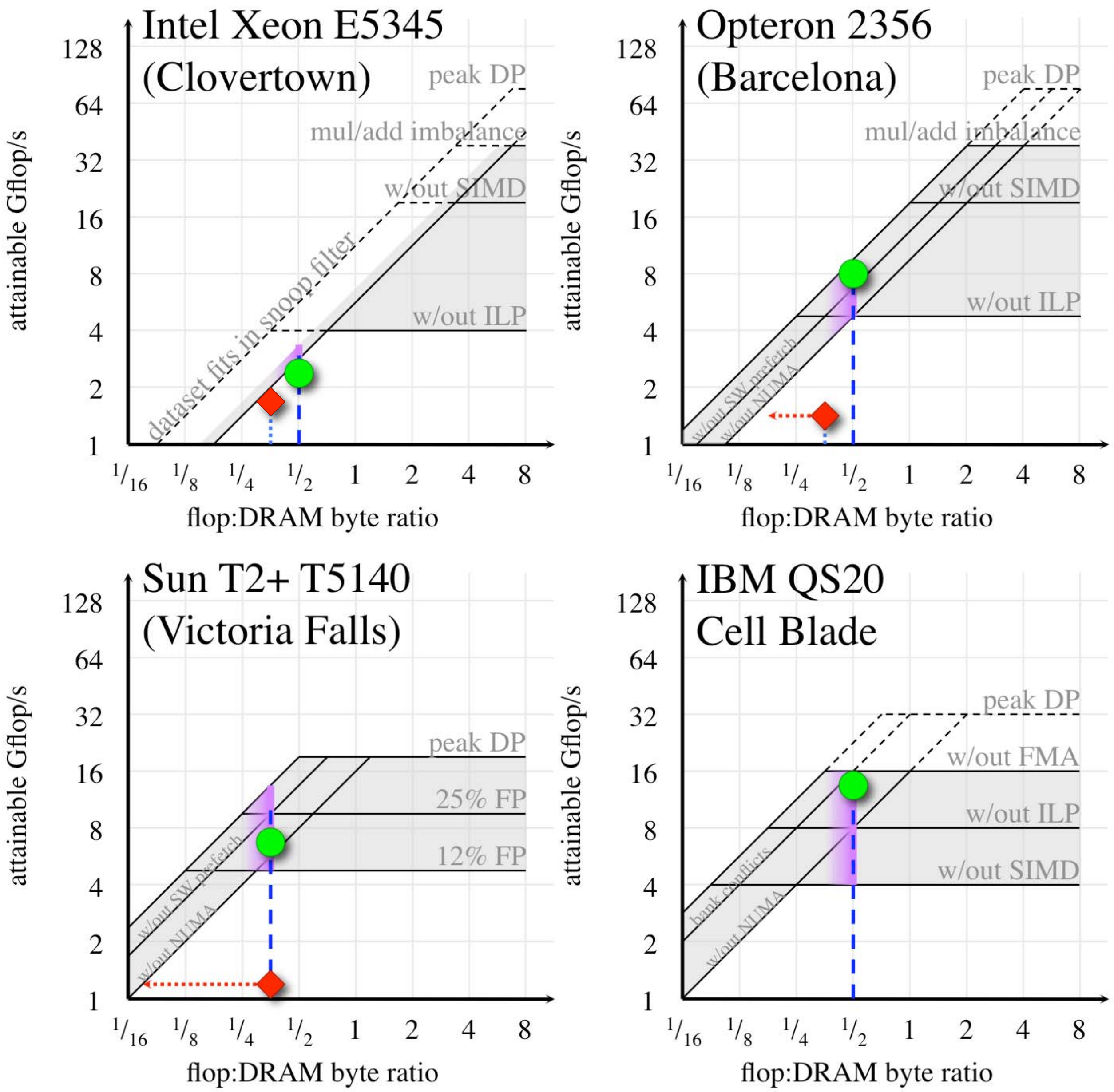




- Cache blocking helps ensure flop:byte ratio is as close as possible to $^1/_3$
- Clovertown has huge caches but is pinned to lower BW ceiling
- Cache management is essential when capacity/thread is low

Roofline model for Stencil

(SIMDization + cache bypass)



- ❖ Make SIMDization explicit
- ❖ Use cache bypass instruction: ***movntpd***
- ❖ Increases flop:byte ratio to ~0.5 on x86/Cell

Programming Models that Match Machines

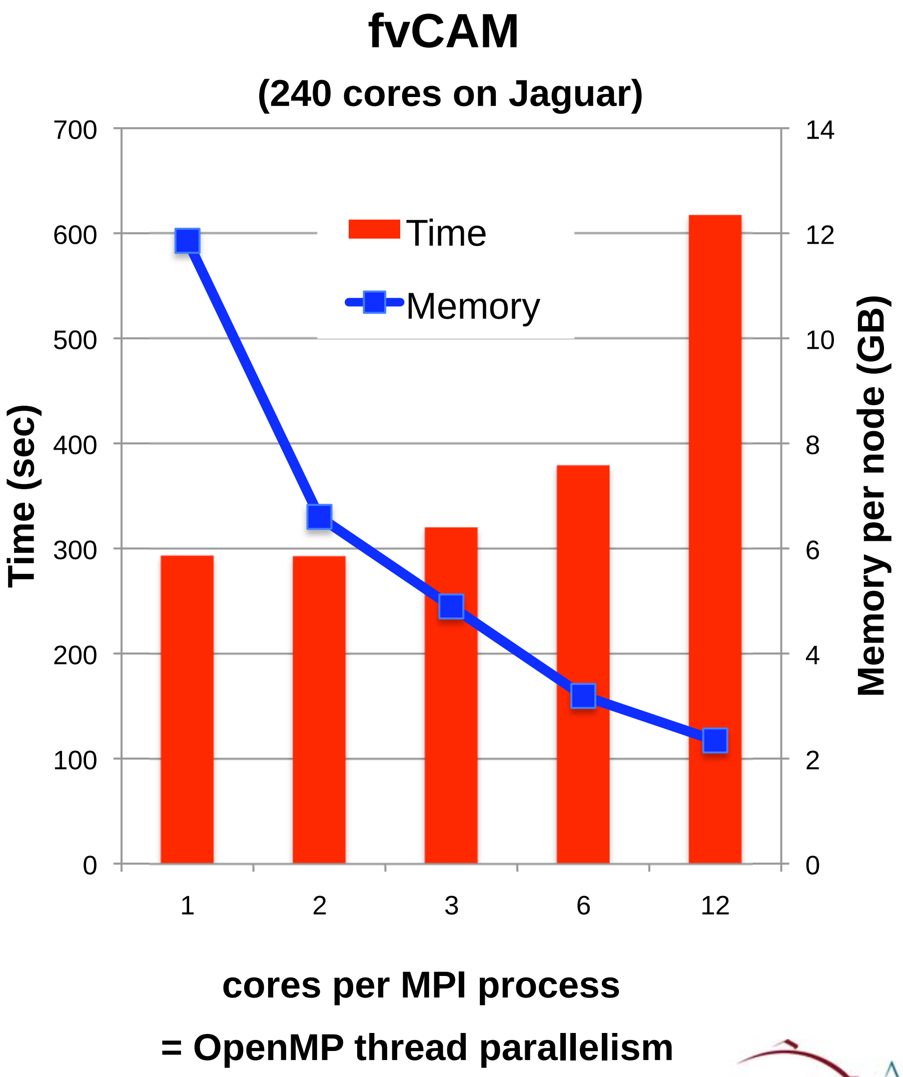
Develop Best Practices in Multicore Programming

NERSC/Cray Programming Models "Center of Excellence" combines:

- LBNL strength in languages, tuning, performance analysis
- Cray strength in languages, compilers, benchmarking

Goals:

- Immediate goal is training material for Hopper users: hybrid OpenMP/MPI
- Long term input into exascale programming model



Work by Nick Wright and John Shalf with Cray

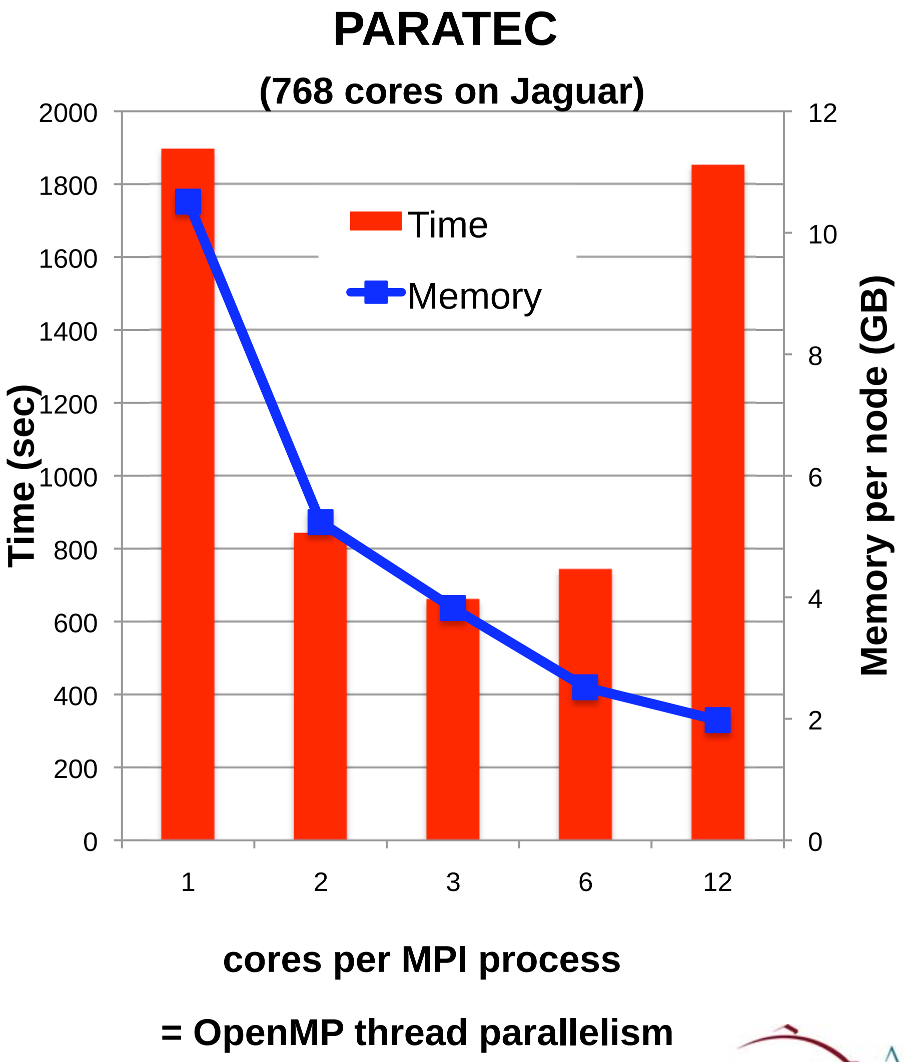
Develop Best Practices in Multicore Programming

Conclusions so far:

- Mixed OpenMP/MPI saves significant memory
- Running time impact varies with application
- 1 MPI process per socket is often good

Run on Hopper next:

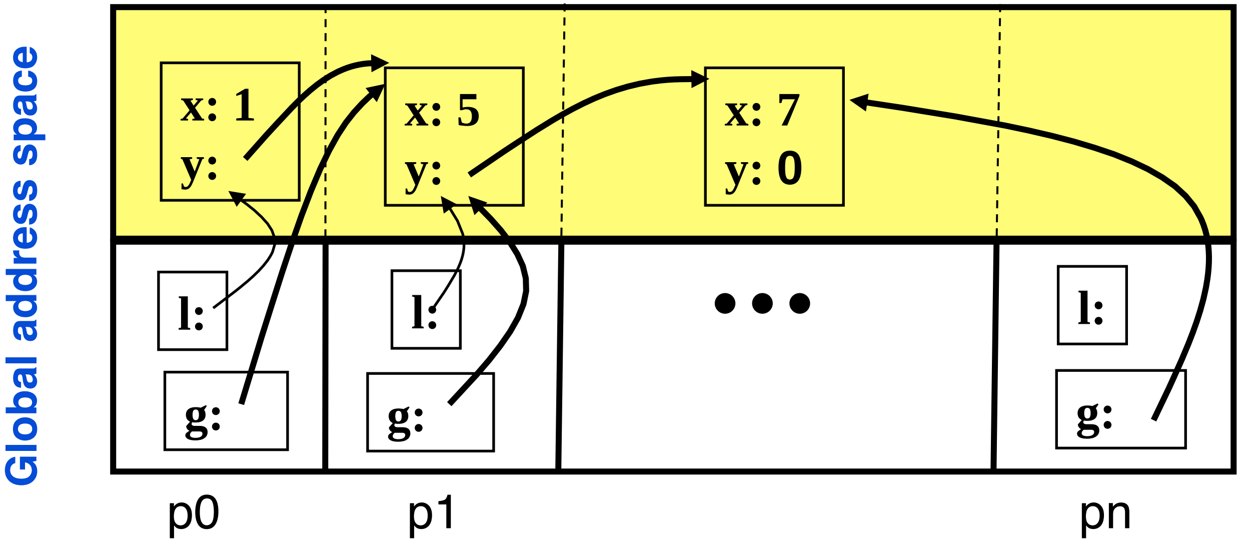
- 12 vs 6 cores per socket
- Gemini vs. Seastar



Work by Nick Wright and John Shalf with Cray

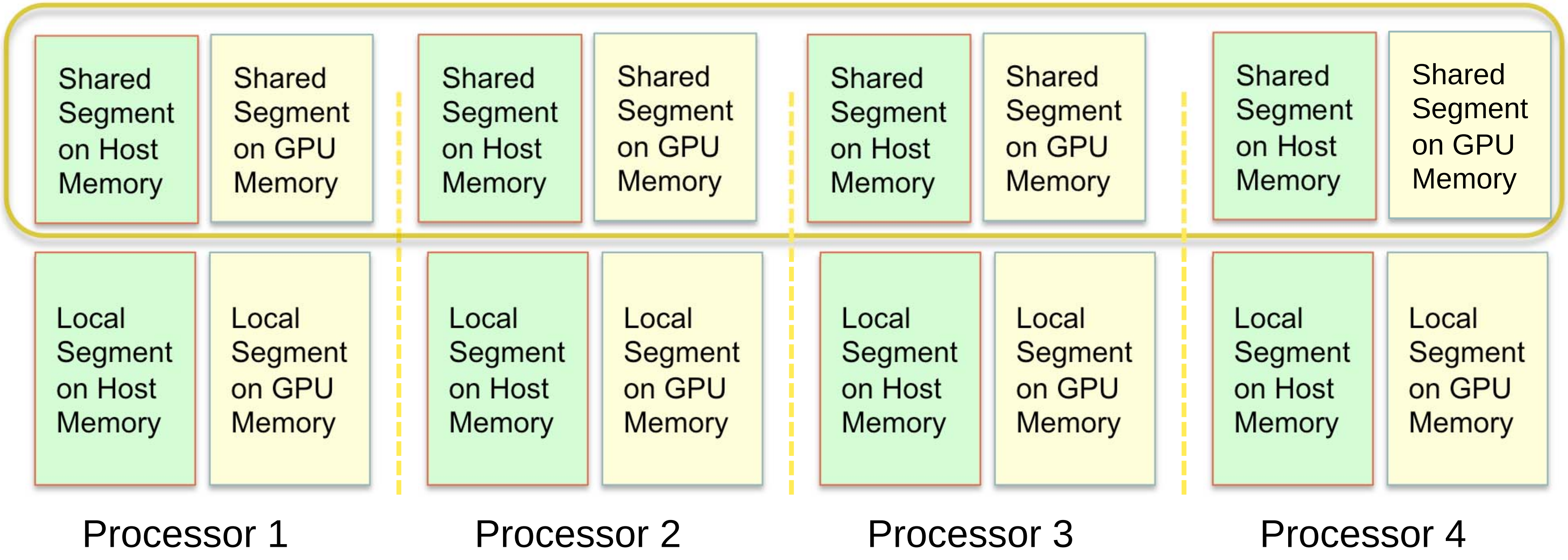
PGAS Languages: Why use 2 Languages (MPI+X) when 1 will do?

Global address space: **thread may directly read/write remote data**
Partitioned: **data is designated as local or global**



Remote put and get: never have to say “receive”
No less scalable than MPI!
Permits sharing, whereas MPI rules it out!
Gives affinity control, useful on shared and distributed memory

Hybrid Partitioned Global Address Space



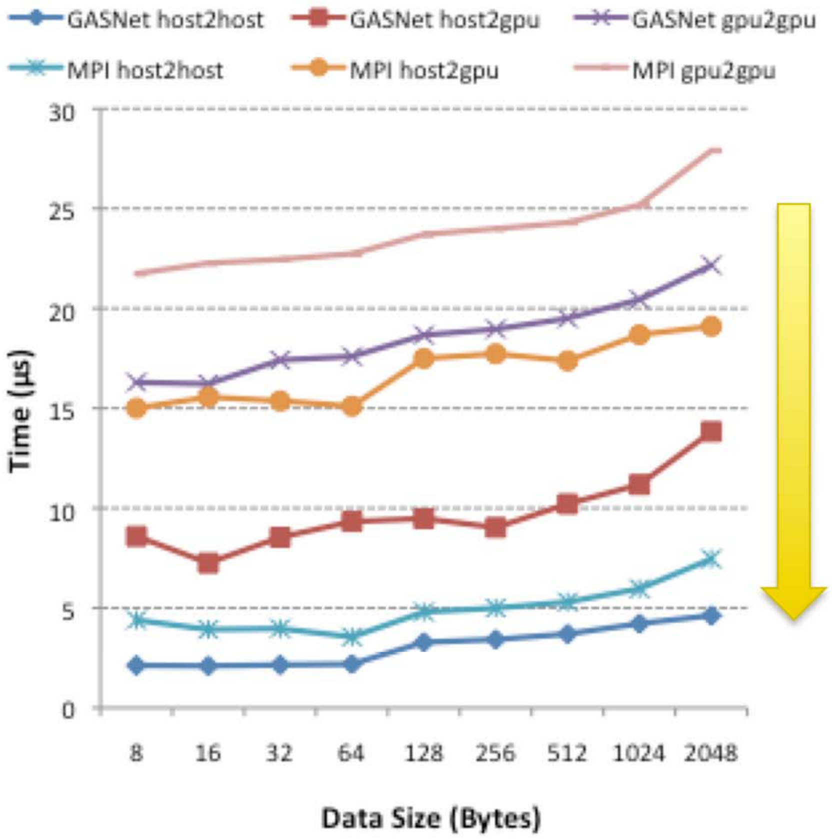
Processor 1		Processor 2		Processor 3		Processor 4	
Shared Segment on Host Memory	Shared Segment on GPU Memory	Shared Segment on Host Memory	Shared Segment on GPU Memory	Shared Segment on Host Memory	Shared Segment on GPU Memory	Shared Segment on Host Memory	Shared Segment on GPU Memory
Local Segment on Host Memory	Local Segment on GPU Memory	Local Segment on Host Memory	Local Segment on GPU Memory	Local Segment on Host Memory	Local Segment on GPU Memory	Local Segment on Host Memory	Local Segment on GPU Memory

- Each thread has only two shared segments
- Decouple the memory model from execution models; one thread per CPU, vs. one thread for all CPU and GPU “cores”
- Caveat: type system and therefore interfaces blow up with different parts of address space

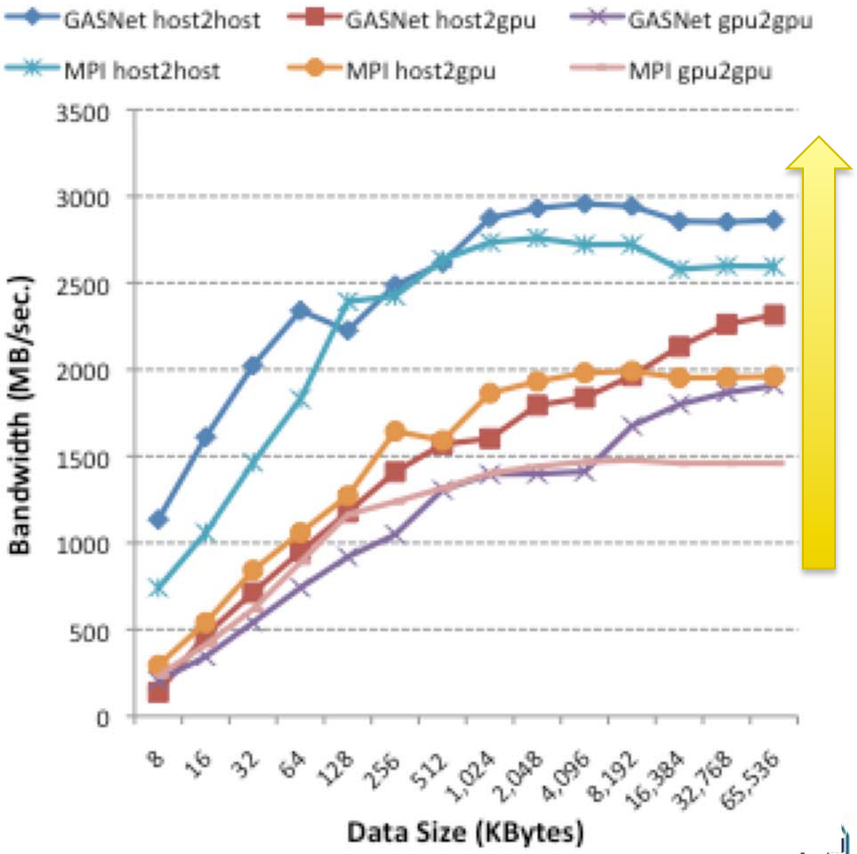
Work by Yili Zheng

GASNet GPU Extension Performance

Latency



Bandwidth

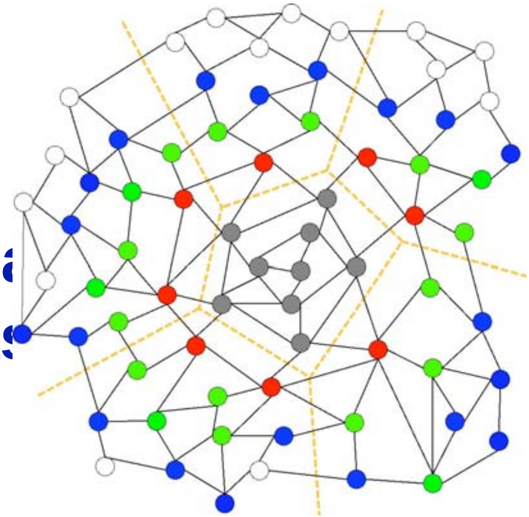


Work by Yili Zheng

Algorithms to Optimize for Communication

Communication-Avoiding Algorithms

- Consider Sparse Iterative Methods
  - Nearest neighbor communication on a mesh
  - Dominated by time to read matrix (edges) from DRAM
  - And (small) communication and global synchronization events at each step



Can we lower data movement costs?

- Take *k* steps “at once” with one matrix rea from DRAM and one communication phas

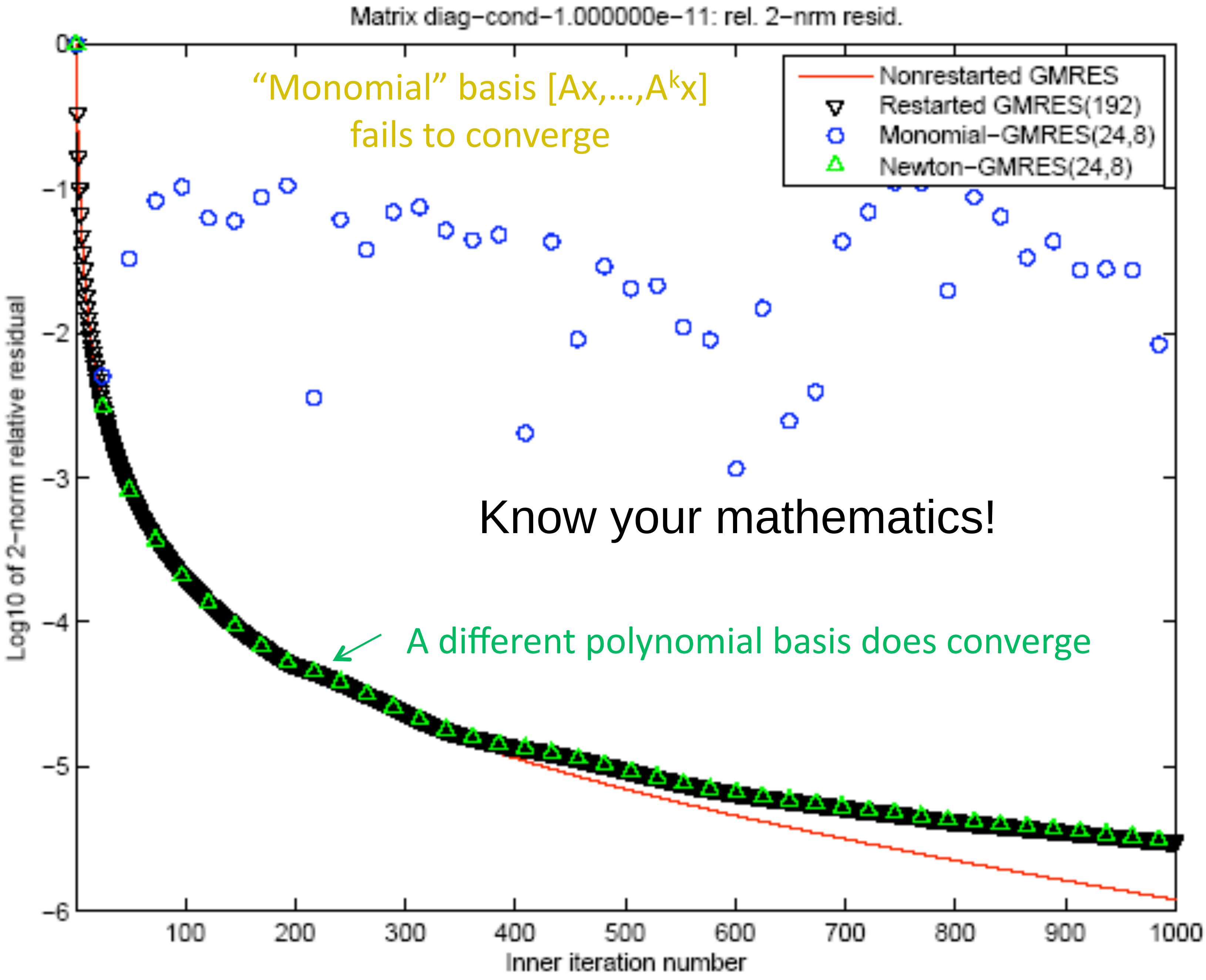
– Parallel implementation

O(log p) messages vs. O(k log p)

– Serial implementation

O(1) moves of data moves vs. O(k)

Joint work with Jim Demmel, Mark Hoemman, Marghoob Mohiyuddin

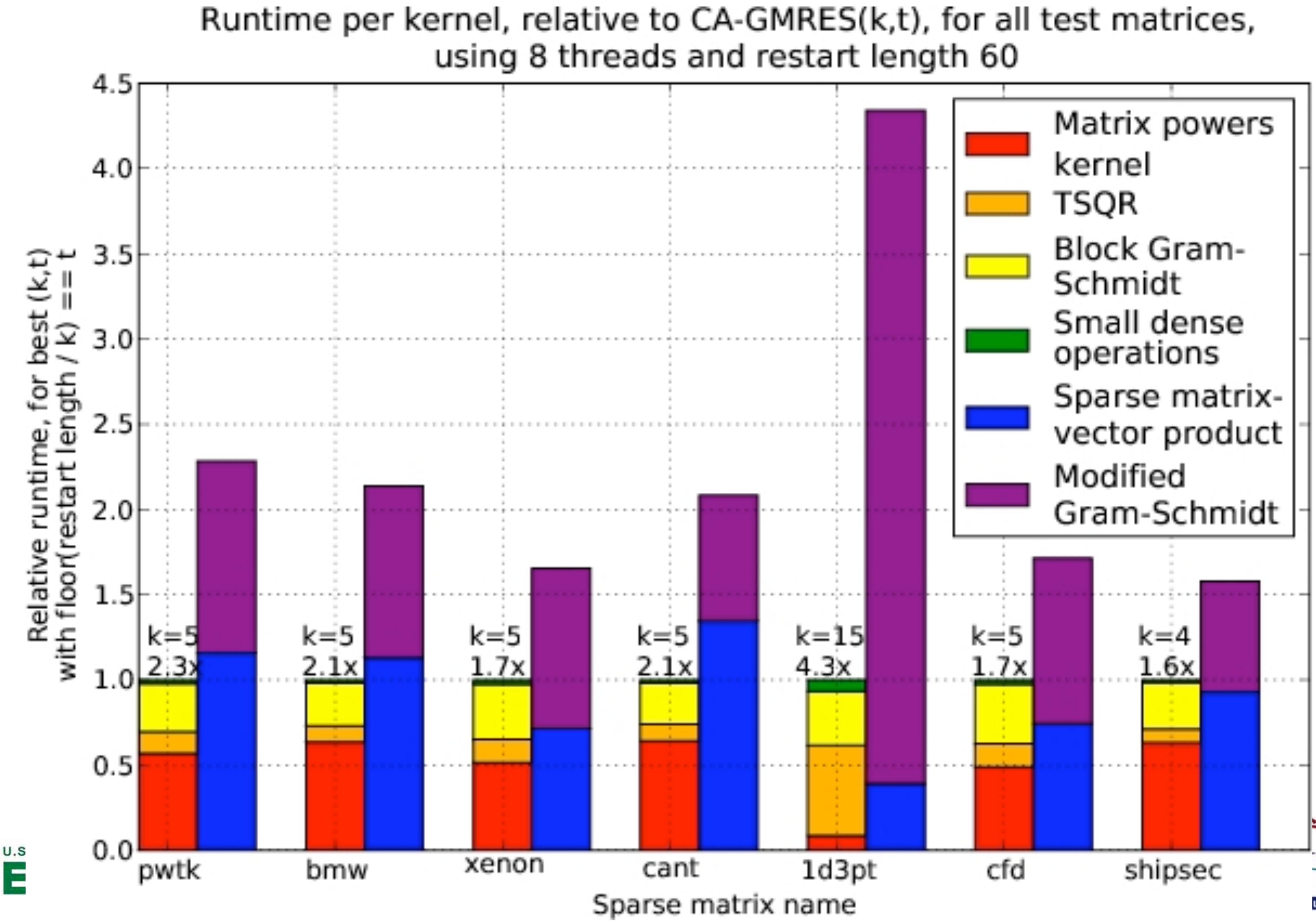


“Monomial” basis [$Ax,\ldots,A^kx$] fails to converge

Know your mathematics!

A different polynomial basis does converge

Communication-Avoiding GMRES on 8-core Clovertown



Runtime per kernel, relative to CA-GMRES(k,t), for all test matrices, using 8 threads and restart length 60

General Lessons

- Early intervention with hardware designs
- Optimize for what is important:

  energy → data movement
- Anticipating and changing the future
  - Influence hardware designs
  - Understand hardware limits
  - Write code generators / autotuners
  - Use programming models that match machines
  - Redesign algorithms for communication

Questions?

See jobs.lbl.gov or send mail if you're interested in joining the team.