Challenges and Strategies for High End Computing

Kathy Yelick

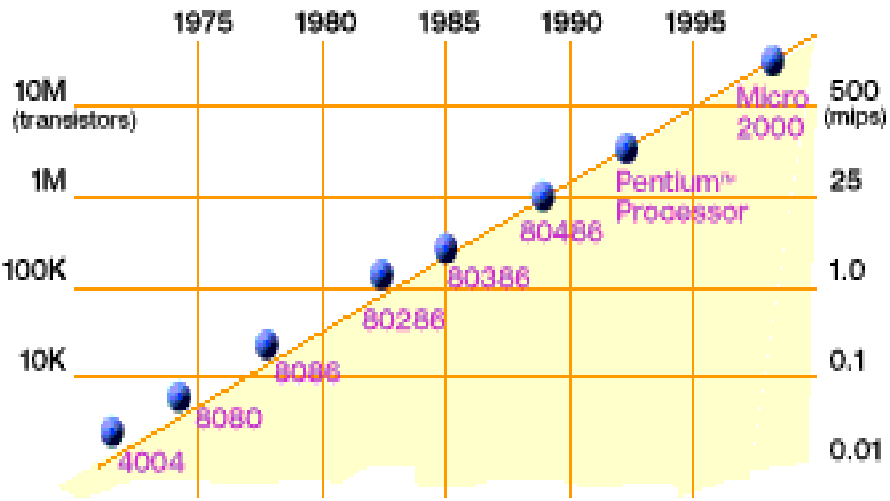
EECS Professor, U.C. Berkeley

NERSC Division Director, LBNL

Major Challenges in High End Computing

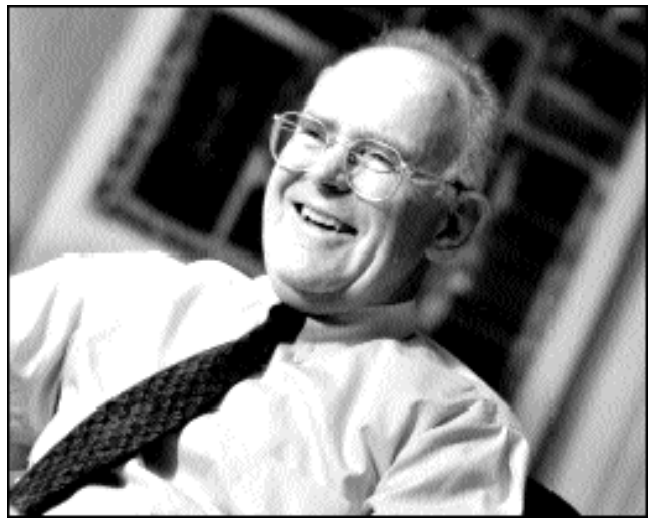
- Shift to multicore complicates programming
- Driven by power density within a chip
- Power consumption of centers is another major challenge

Moore's Law is Alive and Well



2X transistors/Chip Every 1.5 years
Called "Moore's Law"

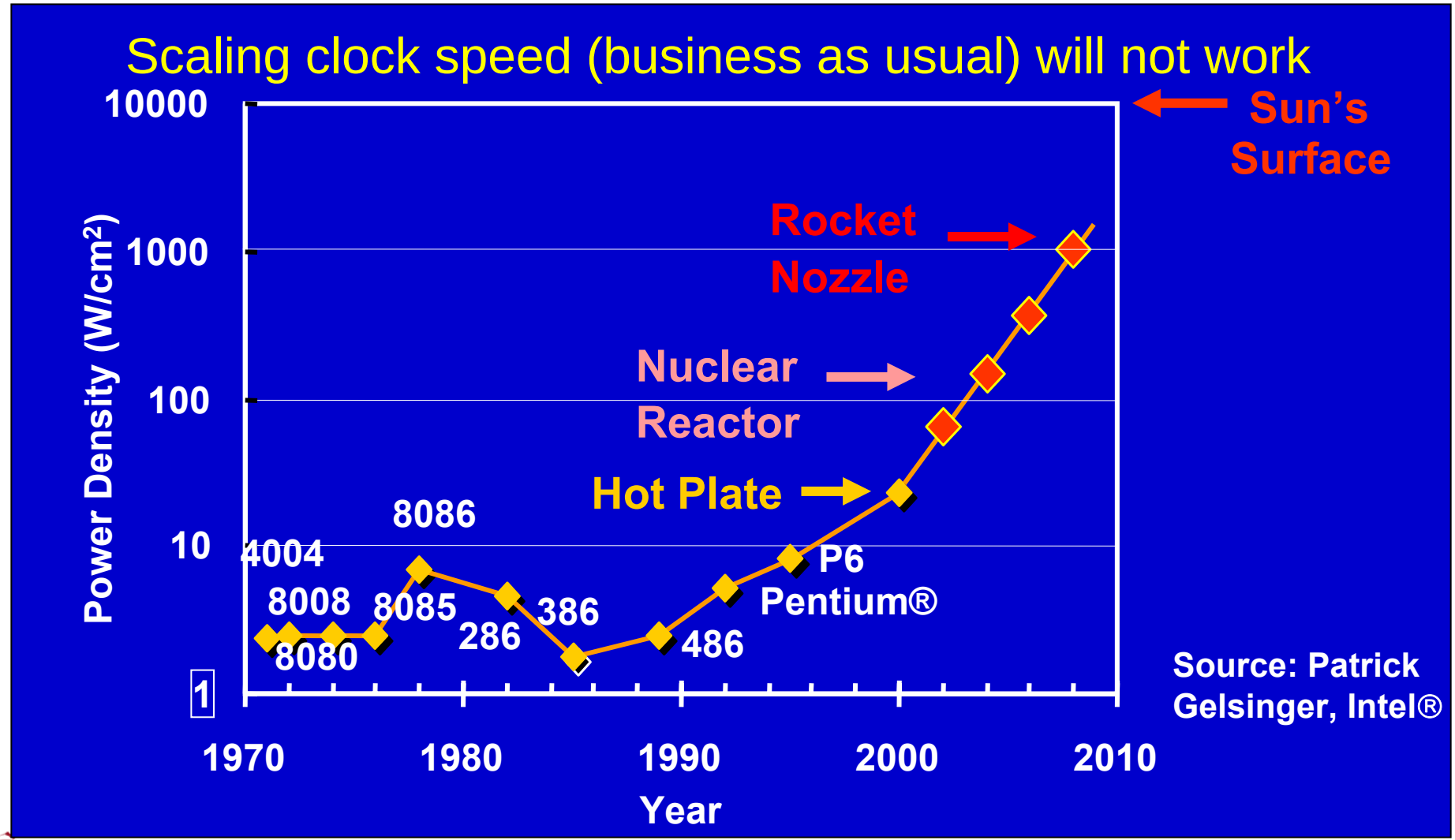
Microprocessors have become smaller, denser, and more powerful.



Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Slide source: Jack Dongarra

Clock Scaling Hits Power Density Wall

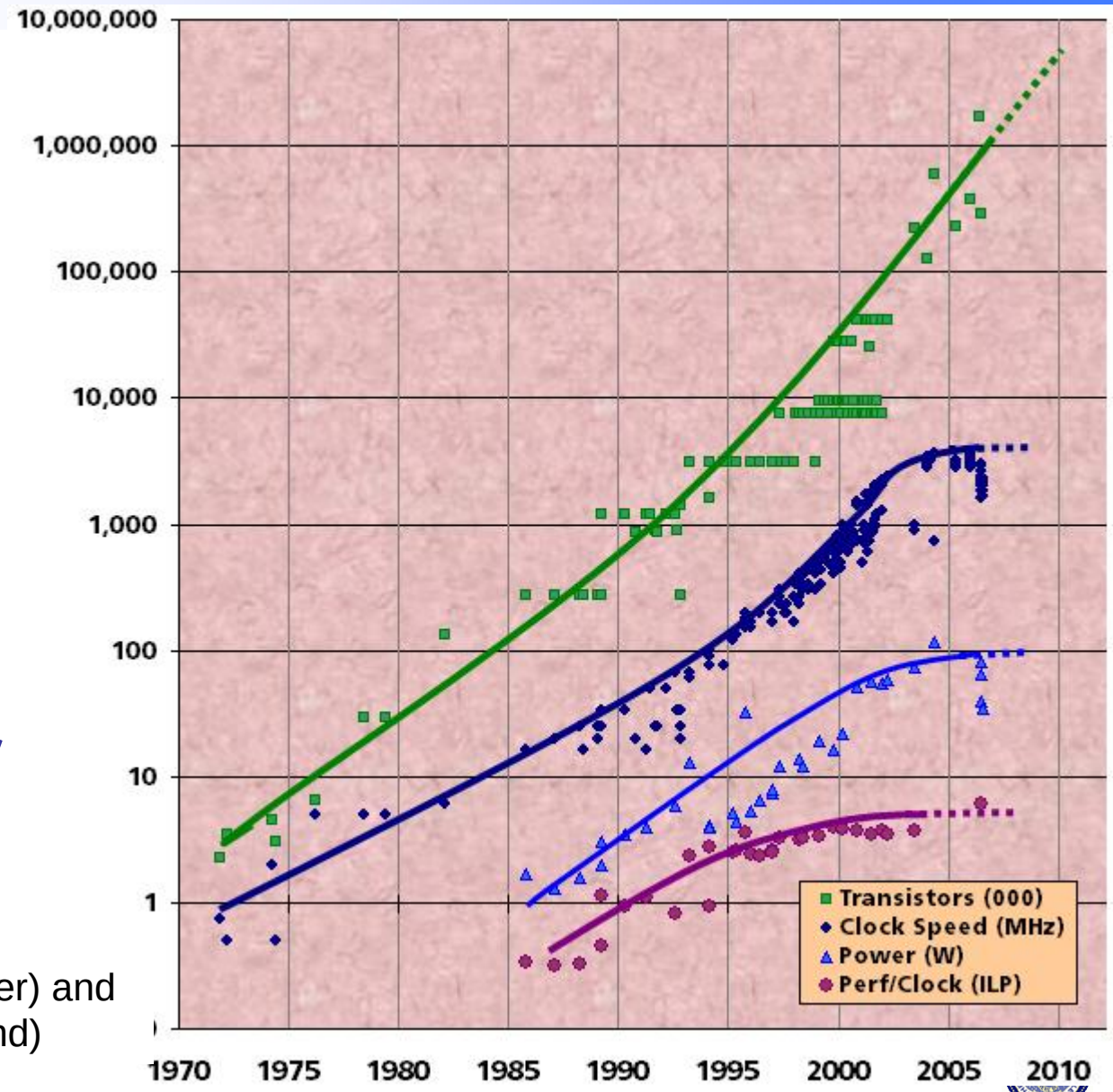


Concurrency for Low Power

- **Highly concurrent systems are more power efficient**
  - ***Dynamic power is proportional to V^2fC***
  - ***Increasing frequency (f) also increases supply voltage (V): more than linear effect***
  - ***Increasing cores increases capacitance (C) but has only a linear effect***
- **Hidden concurrency burns power**
  - **Speculation, dynamic dependence checking, etc.**
  - **Push parallelism discover to software (compilers and application programmers) to save power**
- **Challenge: *Can you double the concurrency in your algorithms every 2 years?***

Revolution is Happening Now

- Chip density is continuing increase ~2x every 2 years
  - Clock speed is not
  - Number of processor cores may double instead
- There is little or no hidden parallelism (ILP) to be

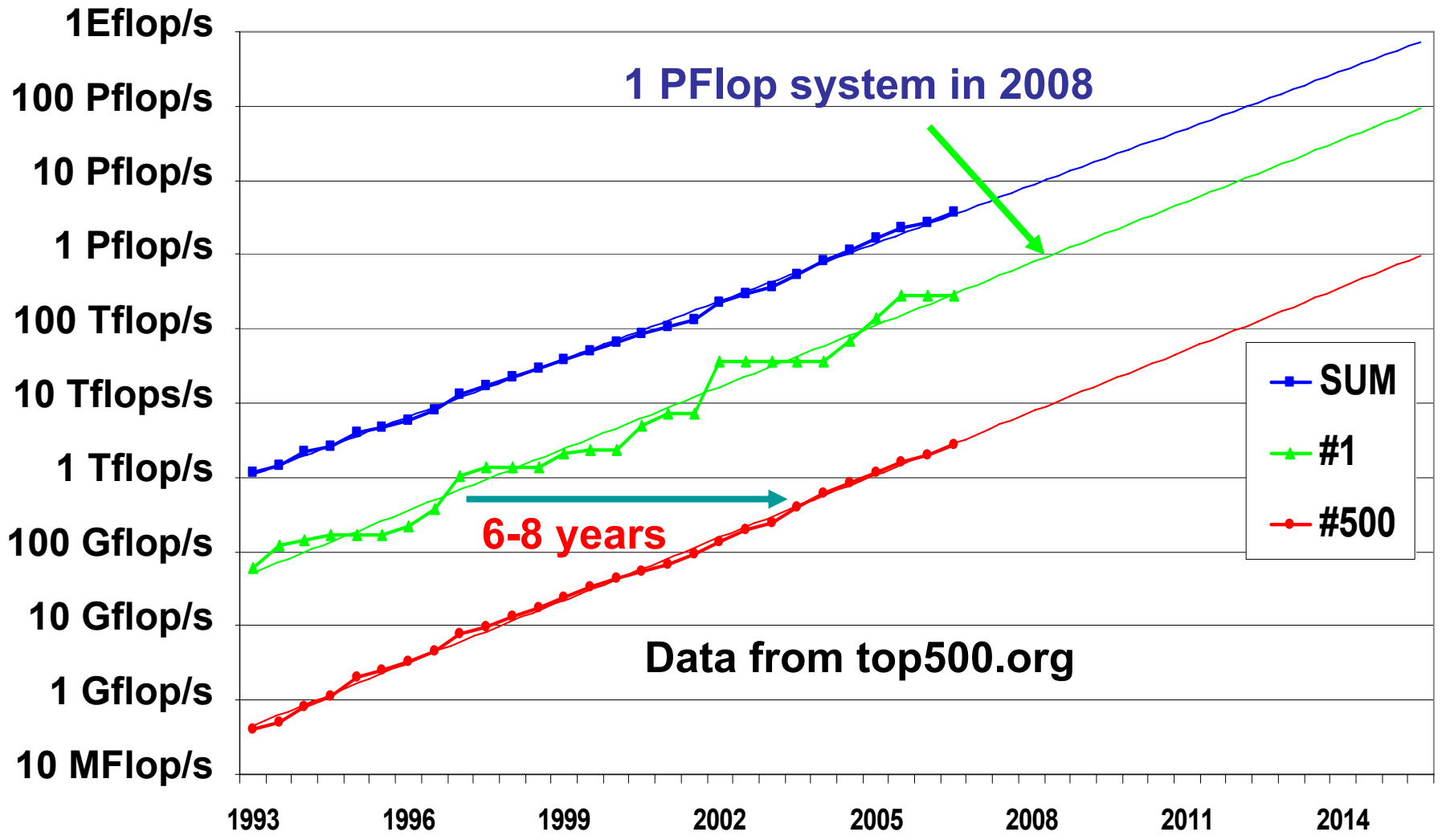


Source: Intel, Microsoft (Sutter) and Stanford (Olukotun, Hammond)

- Parallelism

Petaflop with ~1M Cores

Common by 2015?



Slide source Horst Simon, LBNL

Need a Fundamentally New Approach

- Rethink hardware
  - What limits performance
  - How to build efficient hardware
- Rethink software
  - Massive parallelism
  - Eliminate scaling bottlenecks replication, synchronization
- Rethink algorithms
  - Massive parallelism and locality
  - Counting Flops is the wrong measure

Rethink Hardware

Debunking some Hardware Myths

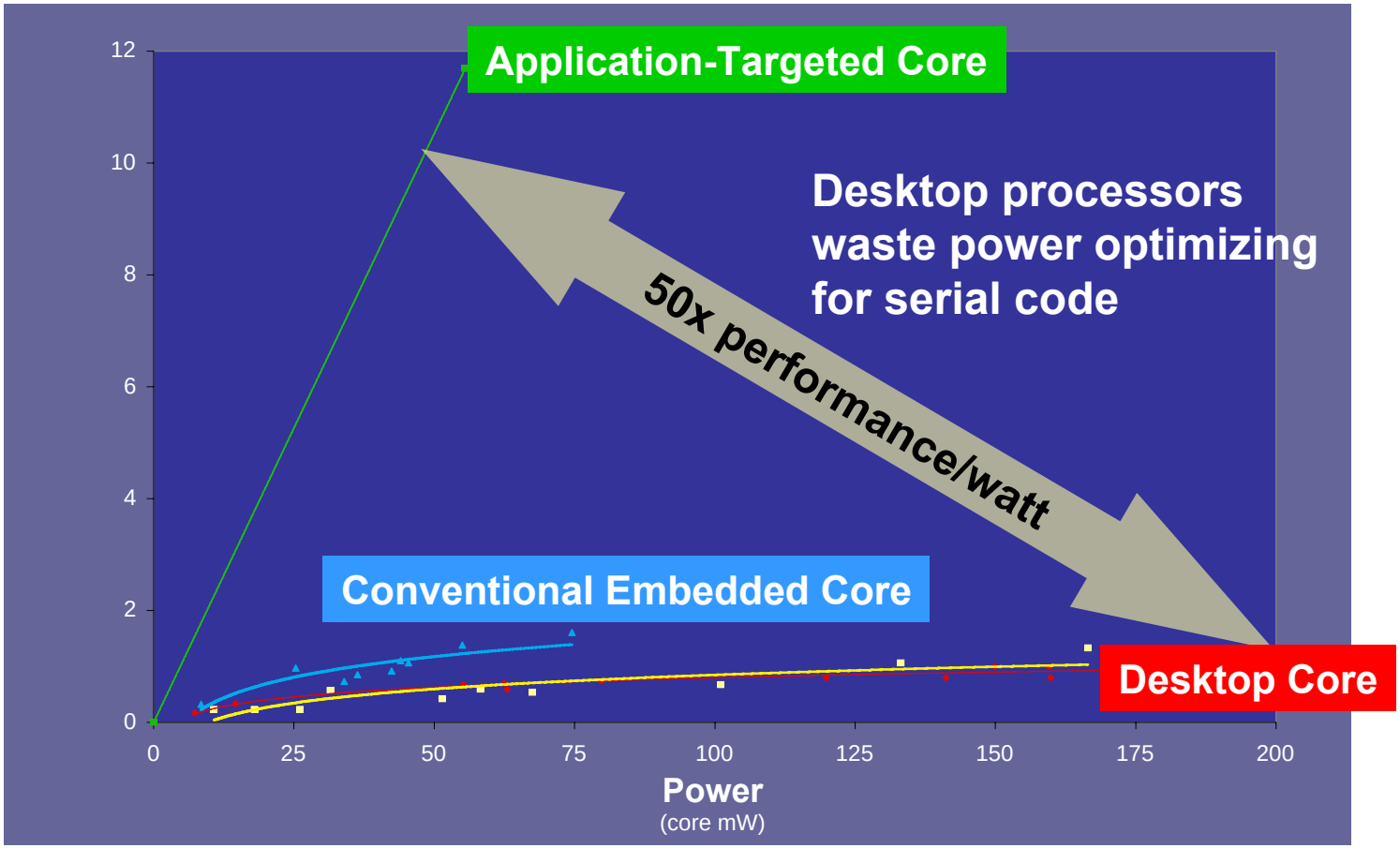
Power Demands Threaten to Limit the Future Growth of Computational Science

- LBNL Study for Climate Modeling in 2008 (Shalf, Wehner, Oliker)
  - Extrapolation of Blue Gene and AMD design trends
  - Estimate: 20 MW for BG and 179 MW for AMD
- DOE E3 Report
  - Extrapolation of existing design trends
  - Estimate: 130 MW
- DARPA Exascale Study
  - More detailed assessment of component technologies
    - Power-constrained design for 2014 technology
    - 3 TF/chip, new memory technology, optical interconnect
  - Estimate: 20 MW for memory alone, 60 MW aggregate so far
- NRC Study
  - Power and multicore challenges are not just an HPC problem

Processor Power and Performance

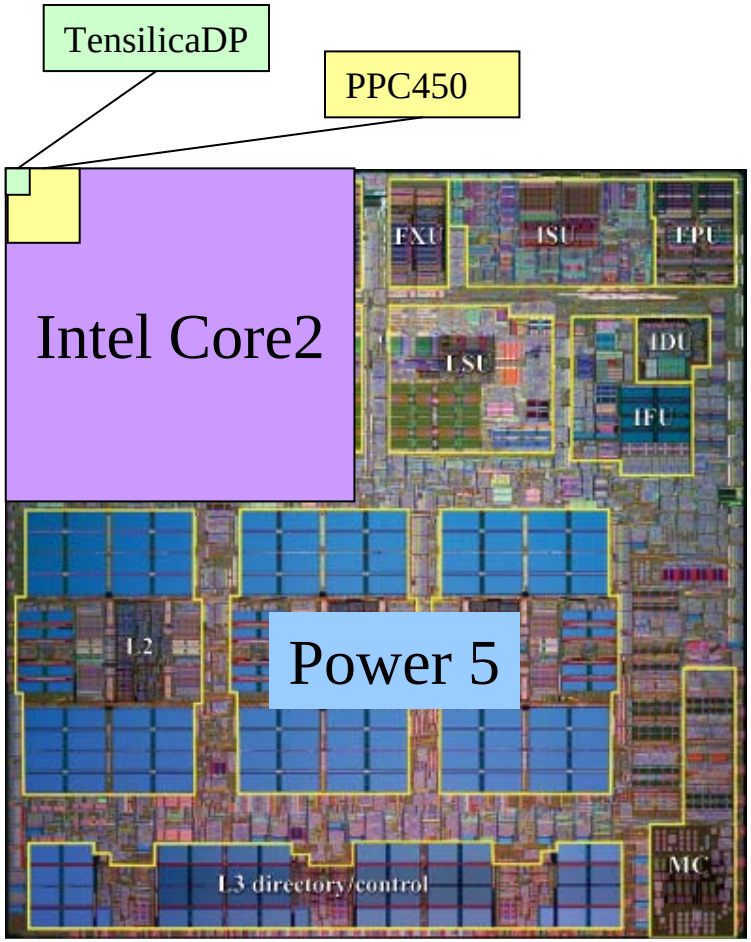
Embedded Application-Specific Cores

Graph courtesy of Chris Rowen, Tensilica Inc.



Performance on EEMBC benchmarks aggregate for Consumer, Telecom, Office, Network, based on ARM1136J-S (Freescale i.MX31), ARM1026EJ-S, Tensilica Diamond 570T, T1050 and T1030, MIPS 20K, NECVR5000). MIPS M4K, MIPS 4Ke, MIPS 4Ks, MIPS 24K, ARM 968E-S, ARM 966E-S, ARM926EJ-S, ARM7TDMI-S scaled by ratio of Dhrystone MIPS within architecture family. All power figures from vendor websites, 2/23/2006.

How Small Is "Small"?



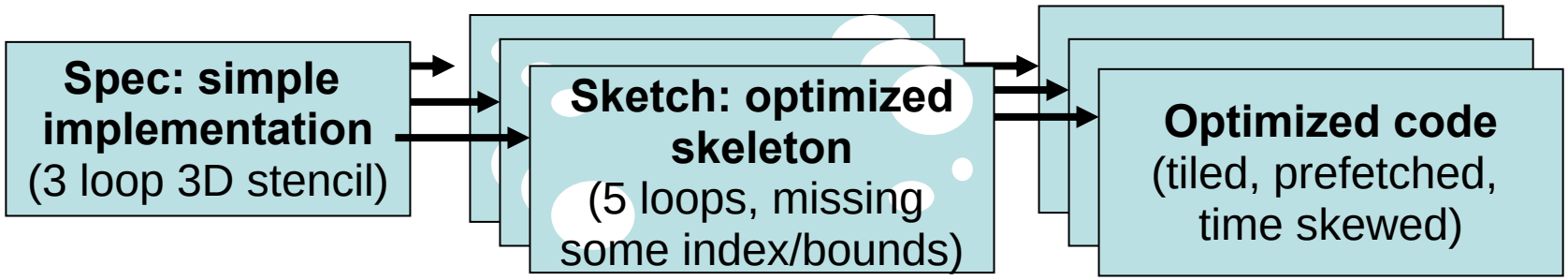
- **Power5 (Server)**
  - 389 mm²
  - 120 W @ 1900 MHz
- **Intel Core2 sc (Laptop)**
  - 130 mm²
  - 15 W @ 1000 MHz
- **PowerPC450 (BlueGene/P)**
  - 8 mm²
  - 3 W @ 850 MHz
- **Tensilica DP (cell phones)**
  - 0.8 mm²
  - 0.09 W @ 650 MHz

Each core operates at 1/3 to 1/10th efficiency of largest chip, but you can pack 100x more cores onto a chip and consume 1/20 the power!

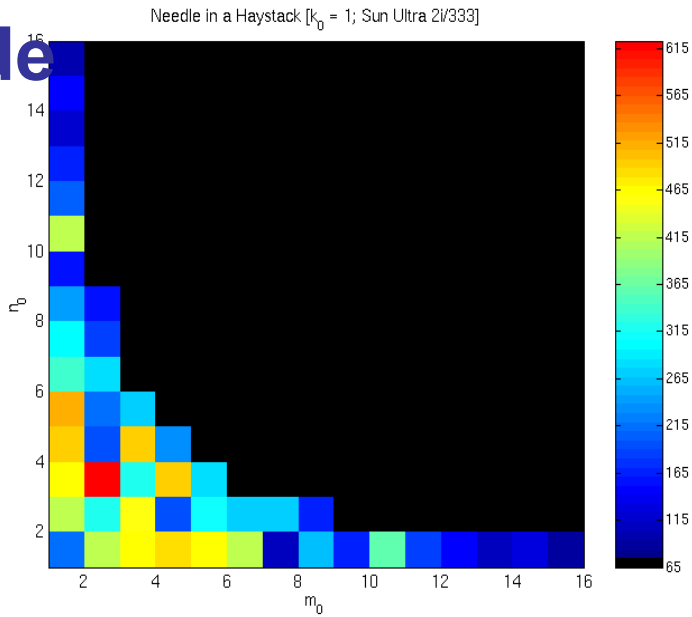
Rethink Software

Program Synthesis

- Needs extensive tuning knobs for writing basic code
- Don't do this by hand: tools for tuning

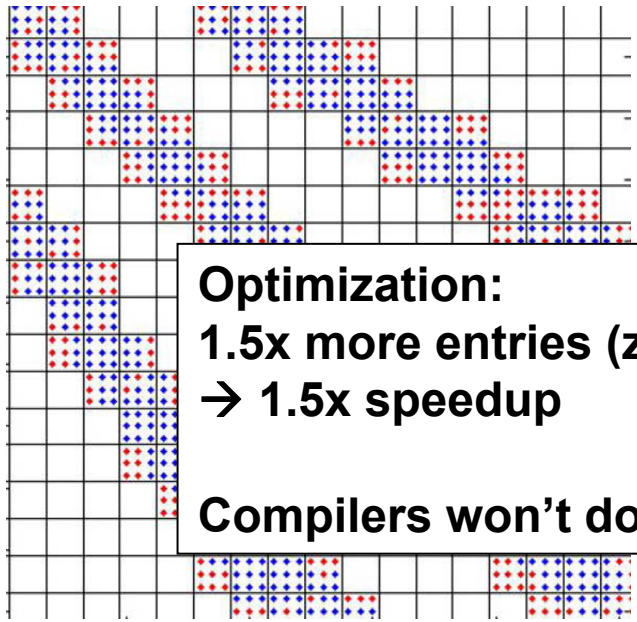


- Autotuning: self-tuning code
  - Can select from algorithms/data structures changes not producible by compiler transform



Tools for Efficiency: Autotuning

- Automatic performance tuning
  - Use machine time in place of human time for tuning
  - Search over possible implementations
  - Use performance models to restrict search space
  - Autotuned libraries for dwarfs (up to 10x speedup)
    - Spectral (FFTW, Spiral)
    - Dense (PHiPAC, Atlas)
    - Sparse (Sparsity, OSKI)
    - Stencils/structured grids
  - Are these compilers?
    - Don't transform source
    - There are compilers that use this kind of search
    - But not for the sparse case (transform matrix)



Optimization:
1.5x more entries (zeros)
→ 1.5x speedup

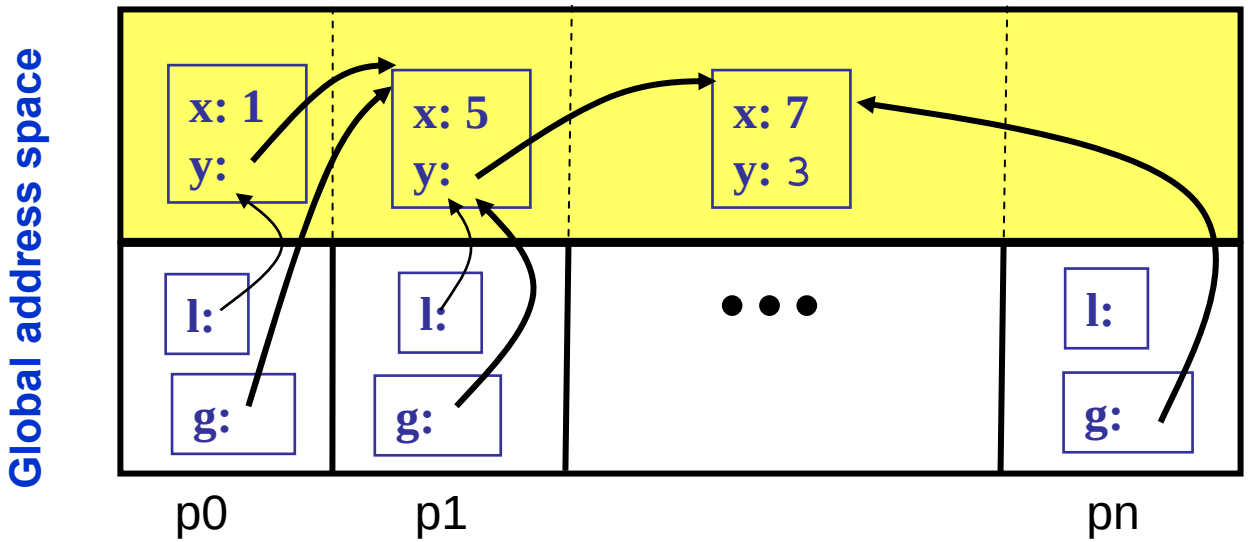
Compilers won't do this!

Sparse Matrix * Vector on Multicore

	intel	AMD	IBM
Name	Clovertown	Opteron	Cell
Chips*Cores	2*4 = 8	2*2 = 4	1*8 = 8
Architecture	4-/3-issue, 2-/1-SSE3, OOO, caches, prefetch		2-VLIW, SIMD, local RAM, DMA
Clock Rate	2.3 GHz	2.2 GHz	3.2 GHz
Peak MemBW	21.3 GB/s	21.3	25.6 GB/s
Peak GFLOPS	74.6 GF	17.6 GF	14.6 (DP Fl. Pt.)
Naïve SpMV (median of many matrices)	1.0 GF	0.6 GF	--
Efficiency %	1%	3%	--

Do New Machines Need New Languages?

- *Global address space:* any thread/process may directly read/write data allocated by another
- *Partitioned:* programmer controls layout
- One model for shared and distributed memory



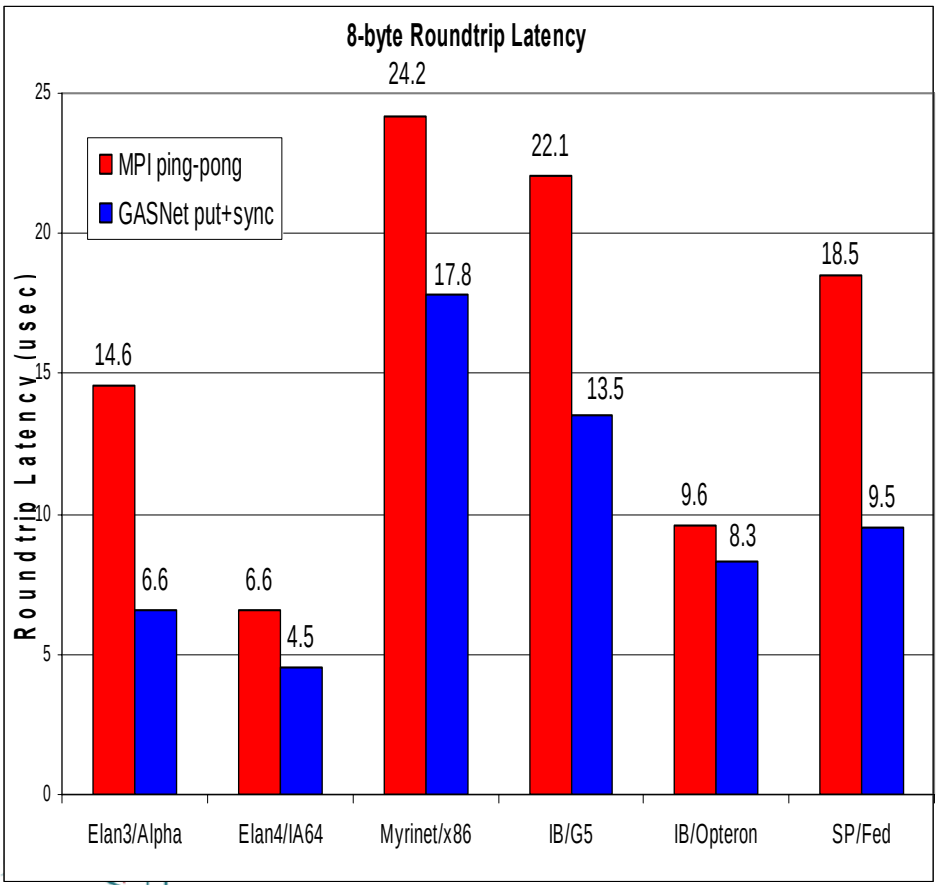
By default:

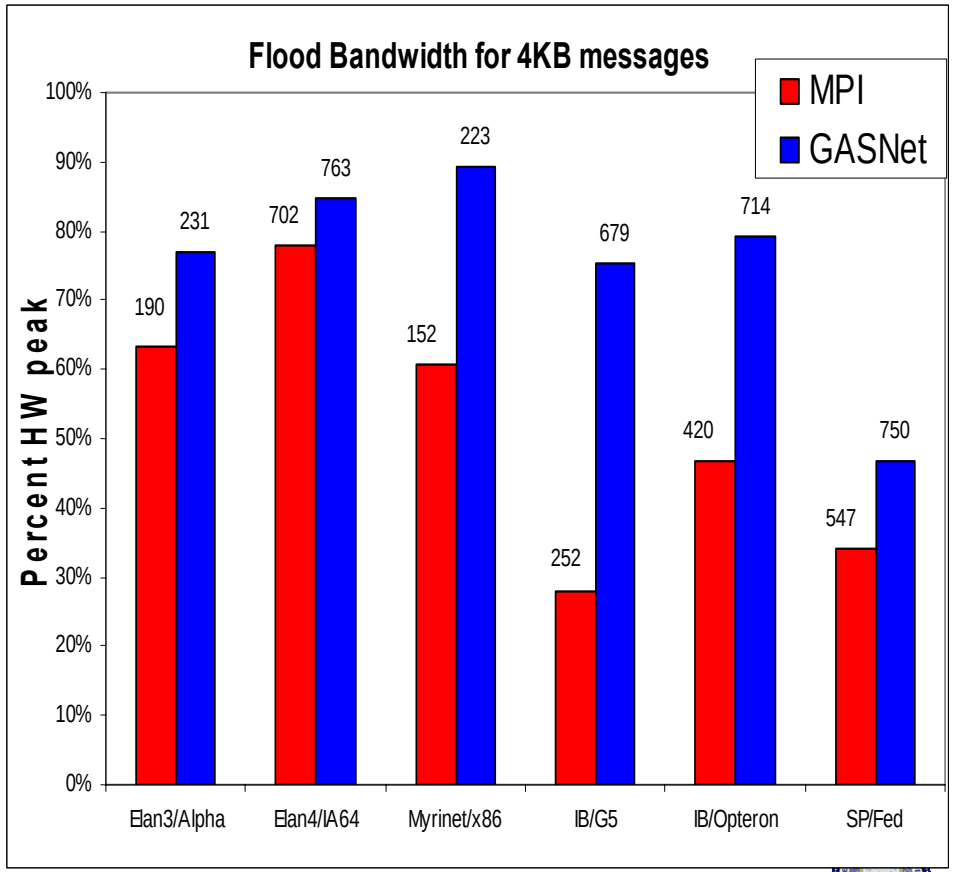
- Object heaps are shared
- Program stacks are private

- *3 Current languages:* UPC, CAF, and Titanium

How to Waste Machine $

- Global Address space allows for sharing (reduce footprint)
- And it gives lower latency and higher bandwidth than two-sided MPI



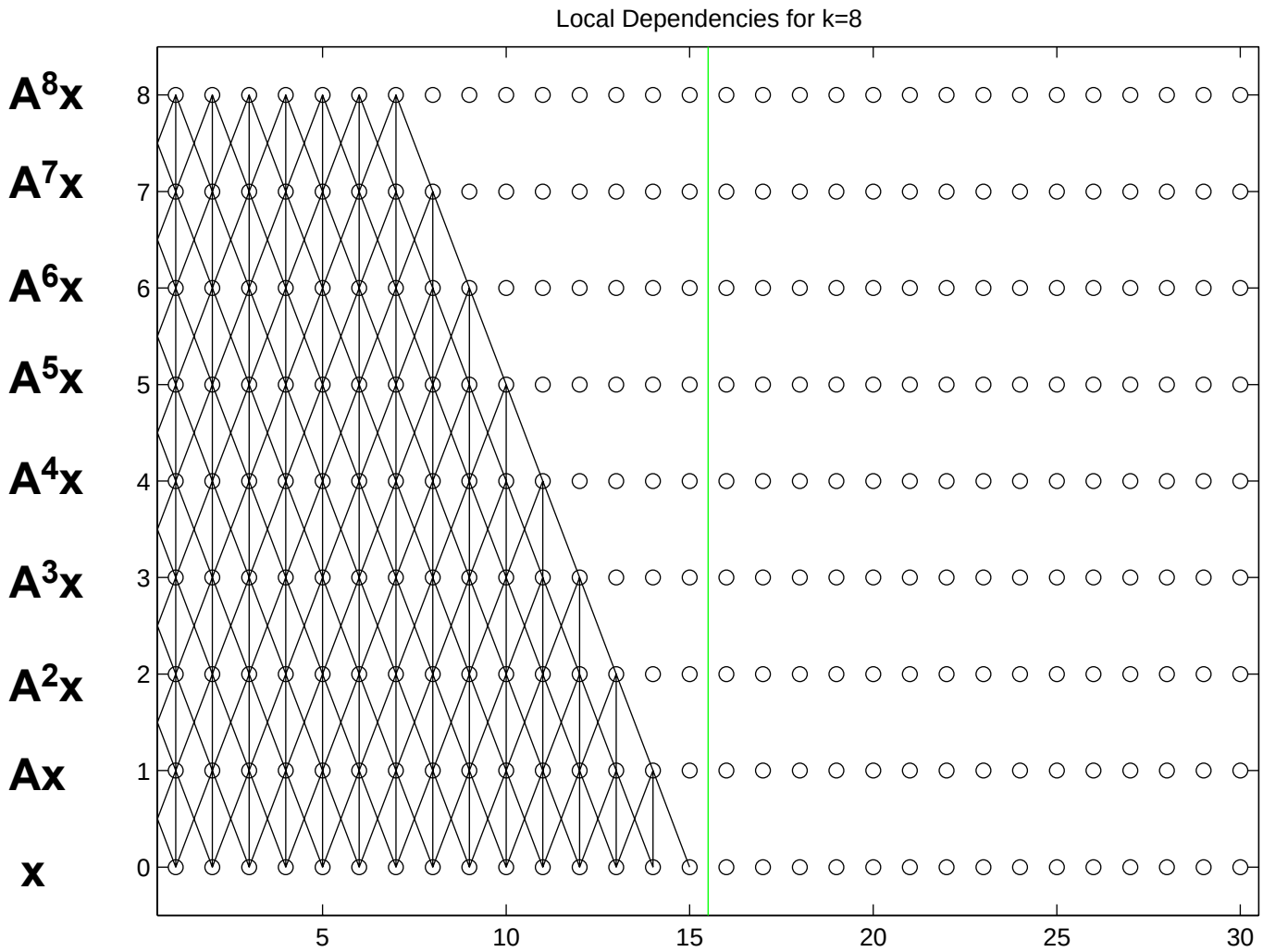


Rethink Algorithms

Latency and Bandwidth-Avoiding

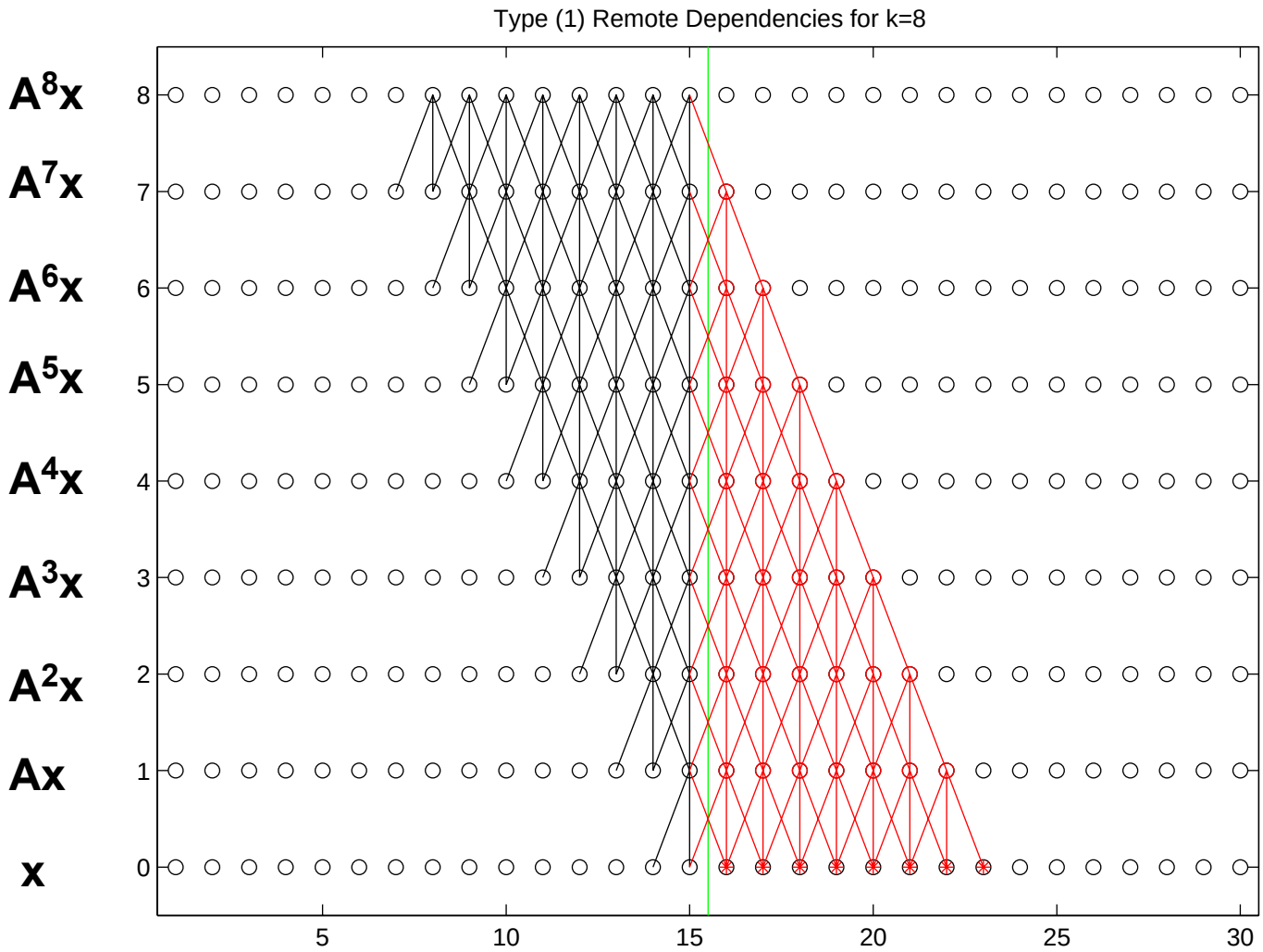
- New optimal ways to implement Krylov subspace methods on parallel and sequential computers
  - Replace $x \rightarrow Ax$ by $x \rightarrow [Ax, A^2x, \ldots A^kx]$
  - Change GMRES, CG, Lanczos, … accordingly
- Theory
  - Minimizes network latency costs on parallel machine
  - Minimizes memory bandwidth and latency costs on sequential machine
- Performance models for 2D problem
  - Up to 7x (overlap) or 15x (no overlap) speedups on BG/P
- Measure speedup: 3.2x for out-of-core

Locally Dependent Entries for [x,Ax,…,A^{8}x], A tridiagonal



Can be computed without communication
k=8 fold reuse of A

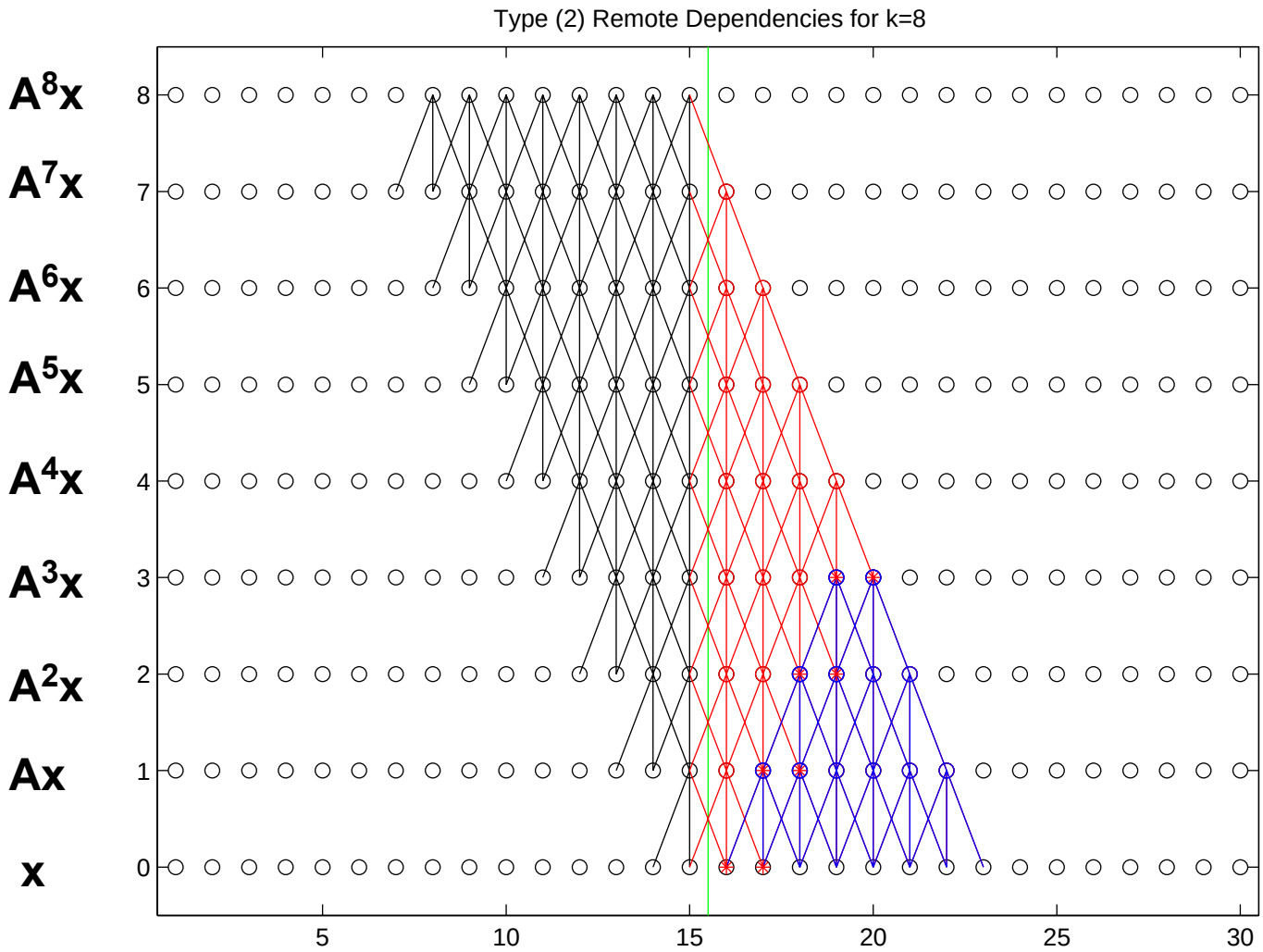
Remotely Dependent Entries for [x,Ax,…,A^8x], A tridiagonal



One message to get data needed to compute remotely dependent entries, not k=8

Price: redundant work

Fewer Remotely Dependent Entries for [x,Ax,…,A^8x], A tridiagonal

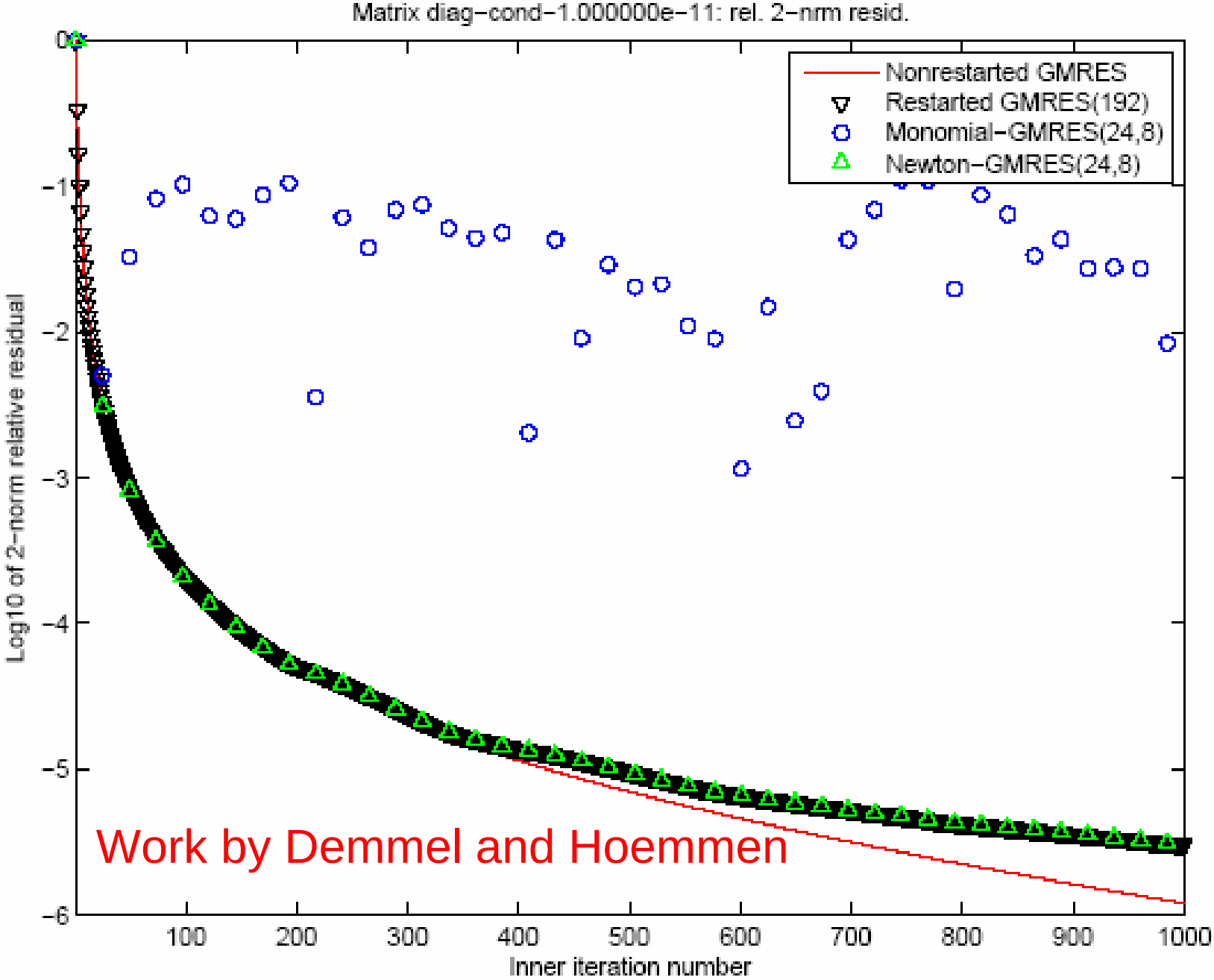


Reduce redundant work by half

Latency Avoiding Parallel Kernel for $[x, Ax, A^2x, \ldots, A^kx]$

- Compute locally dependent entries needed by neighbors
- Send data to neighbors, receive from neighbors
- Compute remaining locally dependent entries
- Wait for receive
- Compute remotely dependent entries

Can use Matrix Power Kernel, but change Algorithms



Work by Demmel and Hoemmen

Predictions and Conclusions

- Parallelism will explode
  - Number of cores will double every 18-24 months
  - Petaflop (million processor) machines will be common in HPC by 2015 (all top 500 machines will have this)
- Performance will become a software problem
  - Parallelism and locality are fundamental; can save power by pushing these to software
- Locality will continue to be important
  - On-chip to off-chip as well as node to node
  - Need to design algorithms for what counts (communication not computation)
- Massive parallelism required (including pipelining and overlap)

Question 1

Question 2

Question 3

Question 4

Question 5