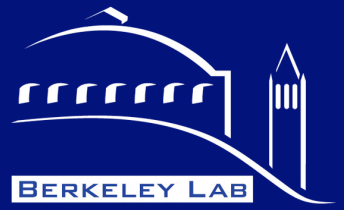
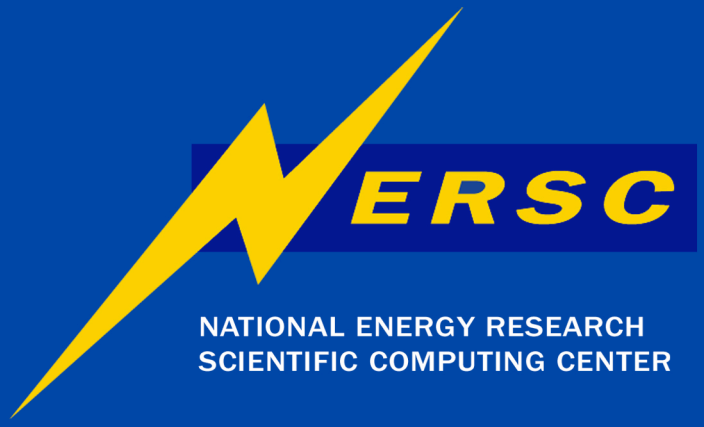




National Energy Research Scientific Computing Center (NERSC)

The New Landscape of Computer Architecture

John Shalf
NERSC Center Division, LBNL

SciDAC2007, Boston
June 25, 2007

Traditional Sources of Performance Improvement are Flat-Lining

- **New Constraints**
  - 15 years of *exponential* clock rate growth has ended
- **But Moore's Law continues!**
  - How do we use all of those transistors to keep performance increasing at historical rates?
  - Industry Response: #cores per chip doubles every 18 months *instead* of clock frequency!

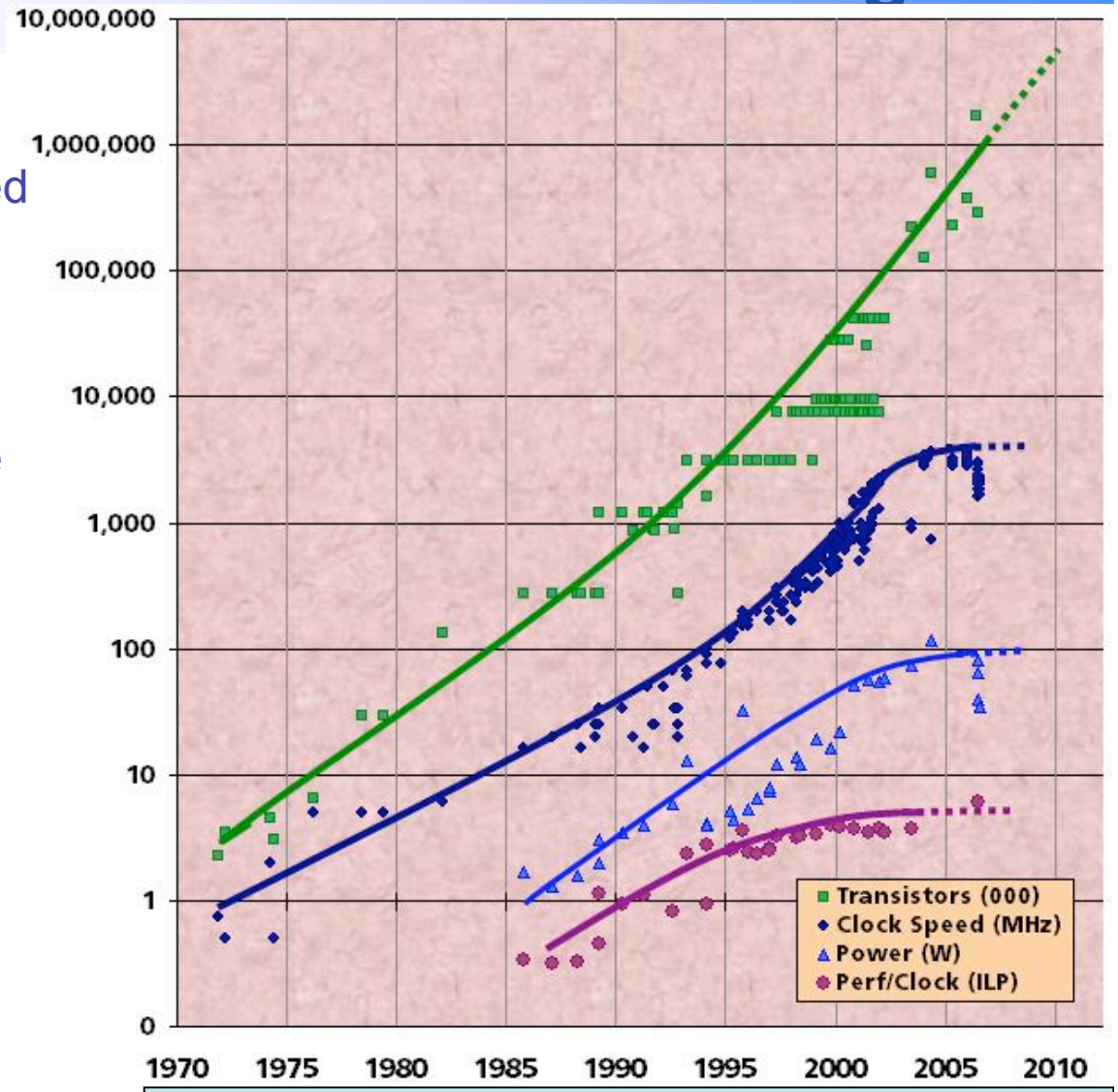
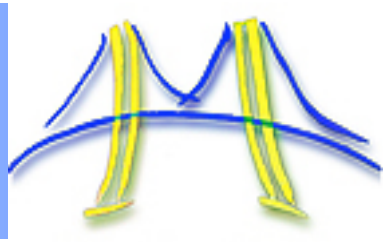


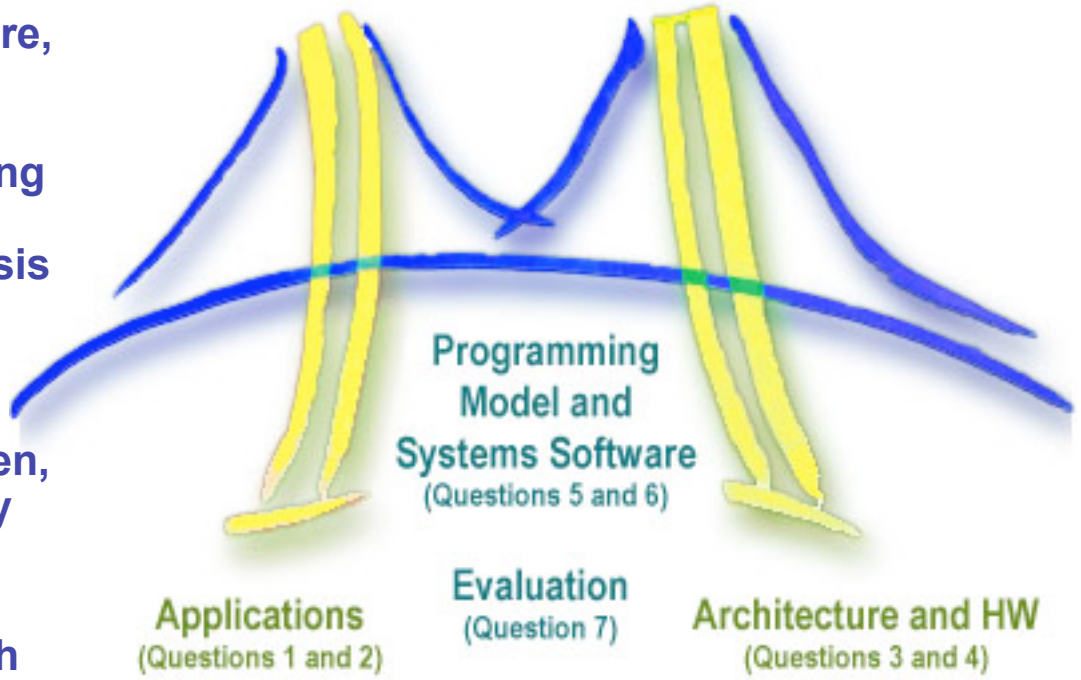
Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

Landscape of Parallel Computing Architecture



Berkeley researchers from many backgrounds meeting since Feb. 2005 to discuss parallelism

- Circuit design, computer architecture, massively parallel computing, computer-aided design, embedded hardware and software, programming languages, compilers, scientific programming, and numerical analysis
- Krste Asanovic, Ras Bodik, Jim Demmel, John Kubiatowicz, Kurt Keutzer, Edward Lee, George Necula, Dave Patterson, Koushik Sen, John Shalf, John Wawrzynek, Kathy Yelick, …
- Tried to learn from successes in parallel embedded (BWRC) and high performance computing (LBNL)
- "Berkeley View" Tech. Report: see http://view.eecs.berkeley.edu



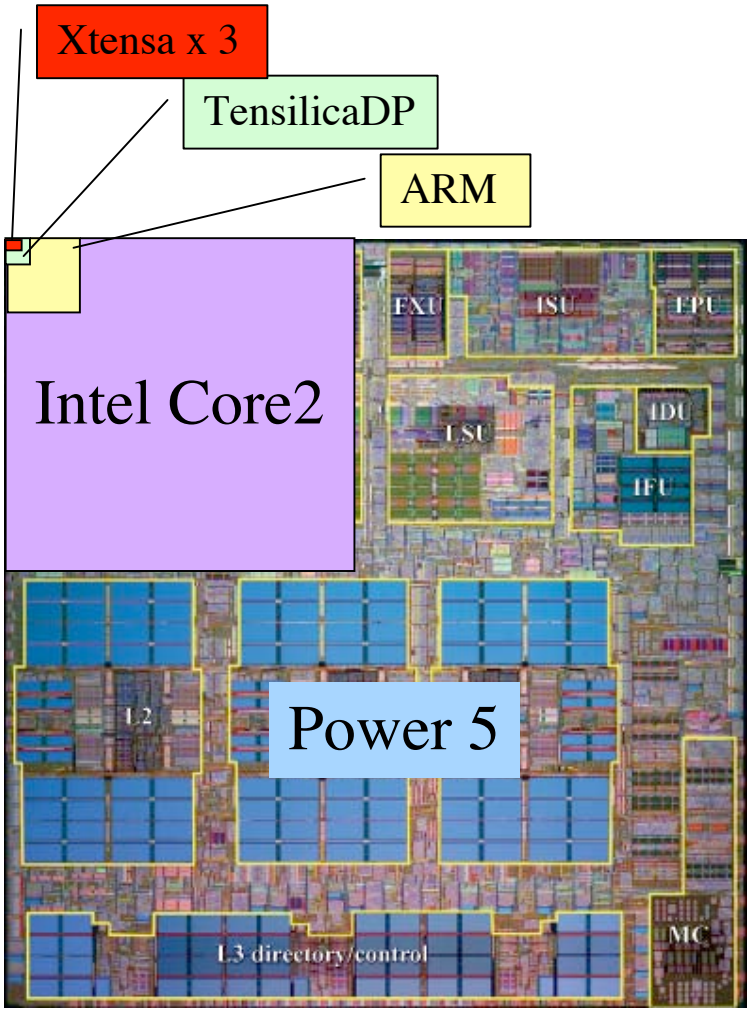
(Inspired by a view of the Golden Gate Bridge from Berkeley)

Office of Science
U.S. DEPARTMENT OF ENERGY

Hardware: What are the problems?

- **Current Hardware/Lithography Constraints**
  - Power limits leading edge chip designs
    - Intel Tejas Pentium 4 cancelled due to power issues
  - Yield on leading edge processes dropping dramatically
    - IBM quotes yields of 10 – 20% on 8-processor Cell
  - Design/validation leading edge chip is becoming unmanageable
    - Verification teams > design teams on leading edge processors
- **Solution: Small Is Beautiful**
  - Expect modestly pipelined (5- to 9-stage) CPUs, FPUs, vector, SIMD PEs
    - Small cores not much slower than large cores
  - Parallel is energy efficient path to performance:CV^2F
    - Lower threshold and supply voltages lowers energy per op
  - Redundant processors can improve chip yield
    - Cisco Metro 188 CPUs + 4 spares; Sun Niagara sells 6 or 8 CPUs
  - Small, regular processing elements easier to verify

How Small is "Small"



- Power5 (Server)
  - 389mm^2
  - 120W@1900MHz
- Intel Core2 sc (laptop)
  - 130mm^2
  - 15W@1000MHz
- ARM Cortex A8 (automobiles)
  - 5mm^2
  - 0.8W@800MHz
- Tensilica DP (cell phones / printers)
  - 0.8mm^2
  - 0.09W@600MHz
- Tensilica Xtensa (Cisco router)
  - 0.32mm^2 for 3!
  - 0.05W@600MHz

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

Consider the comparison

Intel

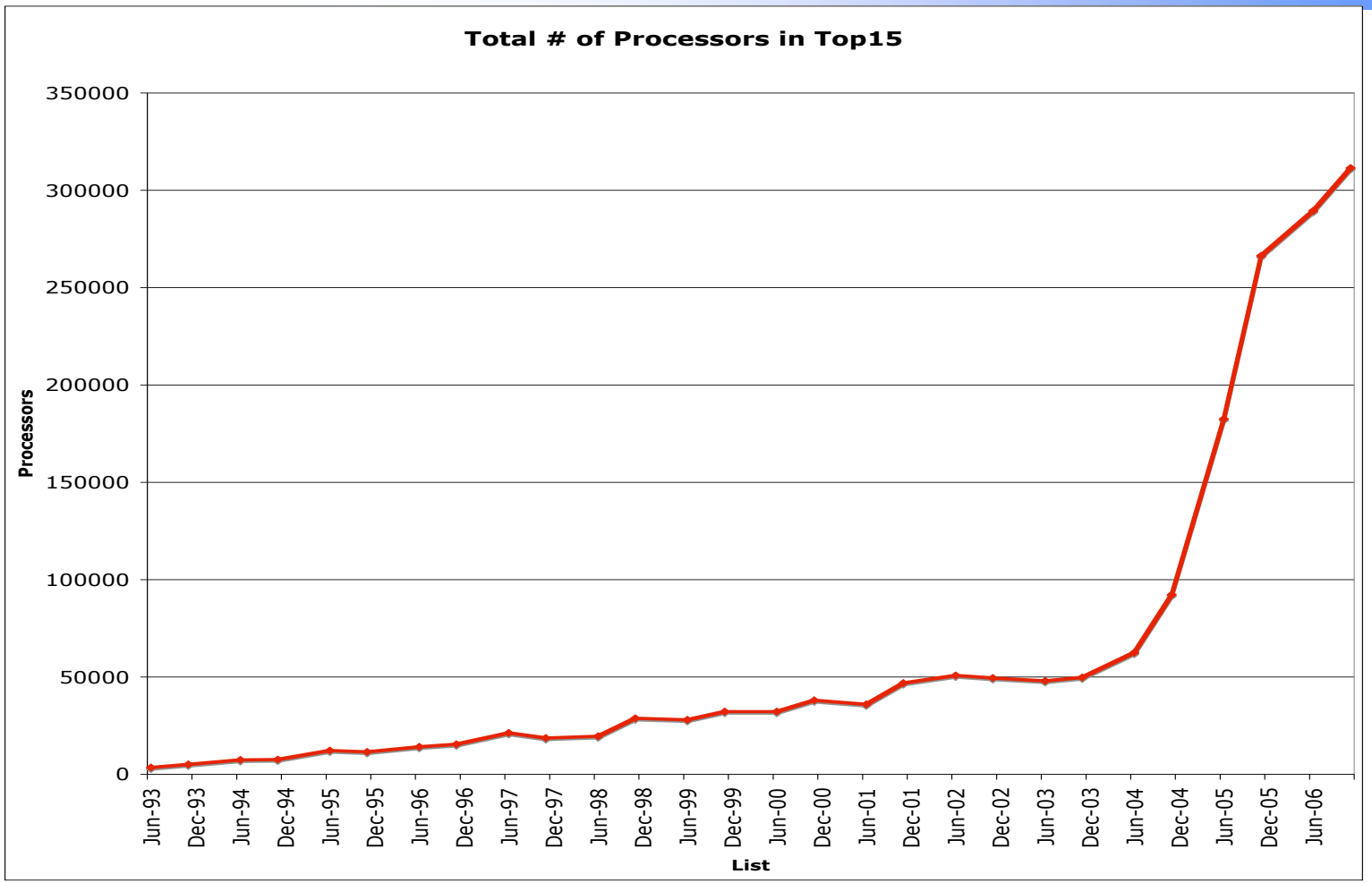
	Traditional Core	Throughput Core	
uArch	Out of Order	In Order	
Size	50	10	mm^2
Power	37.5	6.25	W
Freq	4	4	GHz
Threads	2	4	
Single Thread	1	0.3	Relative Performance
Vector	4 (128-bit)	16 (512-bit)	
Peak Throughput	32	128	GFLOPS
Area Capacity	0.6	13	GFLOPS/mm
Power Capacity	0.9	20	GFLOPS/W

Potential for 20x power efficiency improvement

Multicore vs. Manycore

- **Multicore: current trajectory**
  - Stay with current fastest core design
  - Replicate every 18 months (2, 4, 8 . . . Etc…)
  - Advantage: Do not alienate serial workload
  - Example: AMD X2 (2 core), Intel Core2 Duo (2 cores), Madison (2 cores), AMD Barcelona (4 cores)
- **Manycore: converging in this direction**
  - Simplify cores (shorter pipelines, lower clock frequencies, in-order processing)
  - Start at 100s of cores and replicate every 18 months
  - Advantage: easier verification, defect tolerance, highest compute/surface-area, best power efficiency
  - Examples: Cell SPE (8 cores), Nvidia G80 (128 cores), Intel Polaris (80 cores), Cisco/Tensilica Metro (188 cores)
- **Convergence: Ultimately toward Manycore**
  - Manycore *if we can figure out how to program it!*
  - Hedge: Heterogenous Multicore

The Future of HPC System Concurrency



Must ride exponential wave of increasing concurrency for forseeable future!

You will hit 1M cores sooner than you think!

Office of Science
U.S. DEPARTMENT OF ENERGY

Power Efficiency Motivates Manycore

(why should HPC "users" care?)

- **Power Efficiency concerns drive industry to Manycore**
- **Power is leading factor limiting future system growth**
  - Cost of power > cost of hardware
  - $33M/year projected power+cooling costs at ORNL 2010
  - 130MW projected power for Exascale based on todays technology (>$130M/year for cheap power!)
  - Increasing fraction of OASCR budget will go to power (which means less capability for YOU)
- **Misplaced Concerns**
  - Computational Efficiency is NOT "sustained-to-peak"
  - Computational Efficiency is performance/watt (MPG)
  - *Optimizing performance-per-watt necessarily includes consideration of programmability!*
  - *This means hardware architects MUST understand application requirements to move forward with next-generation architectures! (a considerable departure from status quo)*

The *Entire* Computing Industry is Betting Its future on Parallelism

- **This transition is NOT just about HPC!**
  - Your Motorola Razor Cell Phone already has 8 Tensilica CPU cores in it (and will grow geometrically from there)
  - Cisco CRS-1 router has 188 tensilica CPU cores/socket (Metro) and scales to 400,000 cores! (more than HPC… runs an OS too!)
  - Your *toaster oven* is going be running parallel applications on manycore processors
- **Many key applications that motivate need for increased performance in consumer electronics are familiar scientific computing applications!**
- **Industry has already moved forward with parallelism without having a software solution in place (or even agreed upon)**

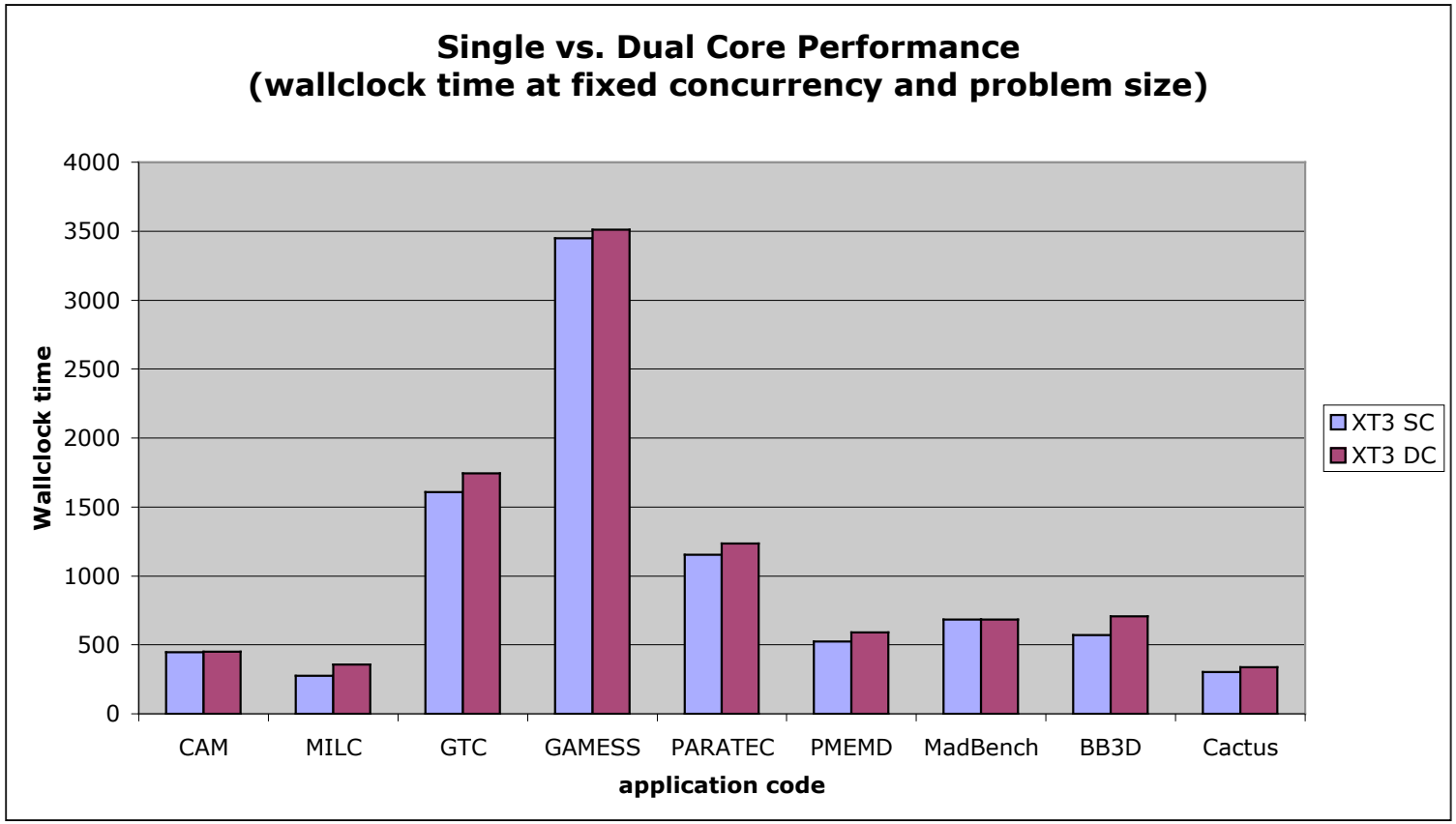
System Balance

- Will multicore slam against the memory wall?

Sensitivity to Memory Bandwidth

(NERSC SSP Application Mix)

NERSC SSP applications selected to represent typical DOE HPC center workload (used to assist with procurements)

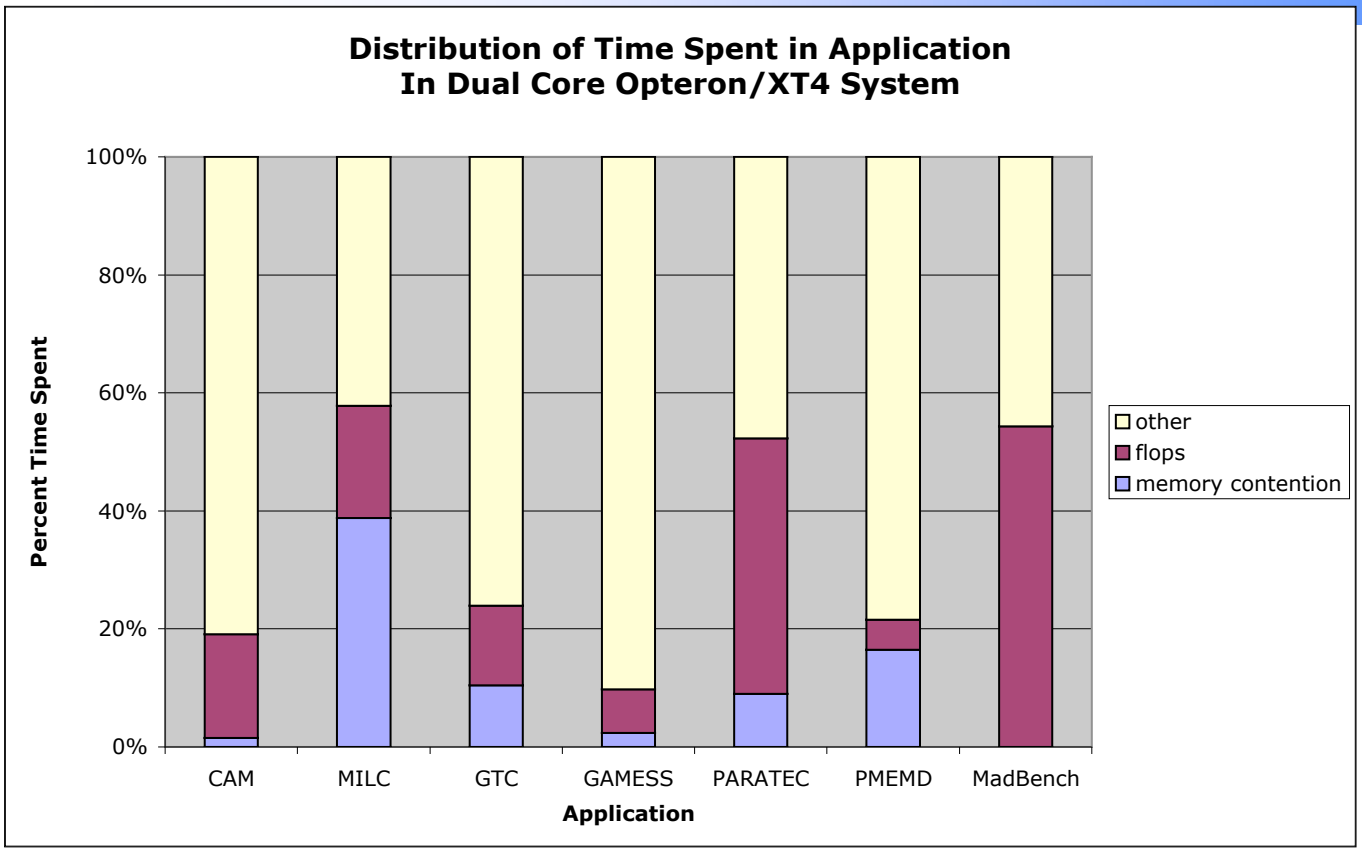


Halving memory bandwidth (moving from single-core to dual-core) has little effect on application runtime

Office of Science
U.S. DEPARTMENT OF ENERGY

Memory Bandwidth

(maintaining system balance)

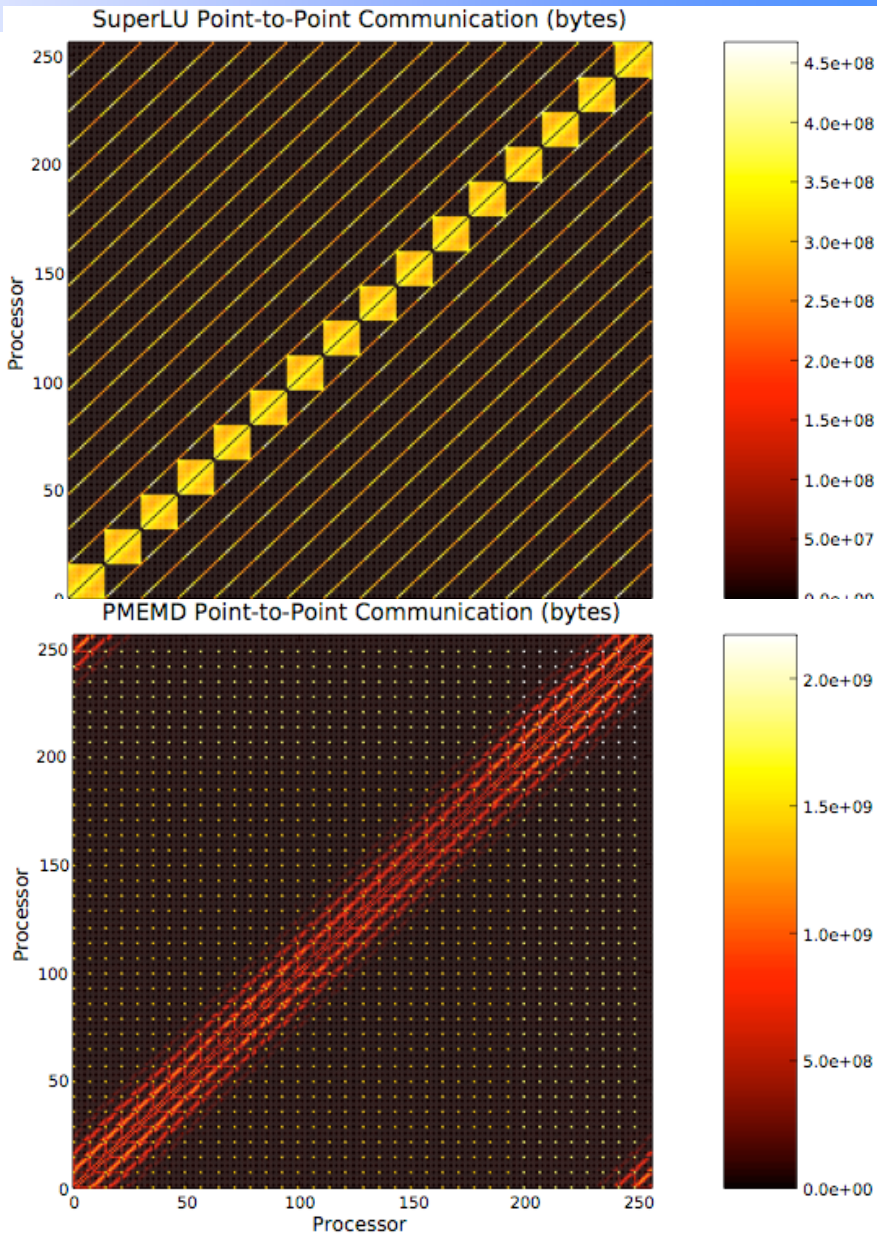


- **Neither memory bandwidth nor FLOPs dominate runtime**
- **The "other" category dominated by memory latency stalls**
- **Points to inadequacies in current CPU core design (inability to tolerate latency)**
- **Multicore relieves rather exacerbates this situation**

Office of Science
U.S. DEPARTMENT OF ENERGY

Interconnect Design Considerations for Massive Concurrency

- **Application studies provide insight to requirements for Interconnects (both on-chip and off-chip)**
  - On-chip interconnect is 2D planar (crossbar won't scale!)
  - Sparse connectivity for dwarfs; crossbar is overkill
  - No single best topology
- **A Bandwidth-oriented network for data**
  - Most point-to-point message exhibit sparse topology & BW bound
- **Separate Latency-oriented network for collectives**
  - E.g., Thinking Machines CM-5, Cray T3D, IBM BlueGene/L&P
- **Ultimately, need to be aware of the on-chip interconnect topology in addition to the off-chip topology**
  - Adaptive topology interconnects (fit-trees)
  - Intelligent task migration?



Office of Science
U.S. DEPARTMENT OF ENERGY

Concerns about Programmability

- Widespread panic regarding a programming model that can ride the "Tsunami of concurrency"
- "*Be afraid. . . Be Very Afraid.*" Ken Kennedy SC06

Programmability

- **Widespread panic over programming model that can ride the "Tsunami of concurrency"**
- **Inter-dependent requirements for programming environment**
  - Productivity
  - Performance
  - Correctness
- **Approaches**
  - Abstracting single-chip parallelism
    - Focus of the Broader Consumer Electronics/Computing Industry
    - Even in HPC, observe that # chips growing much slower than # cores
  - Hiding complexity of global parallelism
    - Frameworks, Advanced compilers and programming languages, Auto-tuning
  - Imagining Infinite Parallelism
    - Includes putting cores to other uses than SPMD application processing

Multicore is NOT a Familiar Programming Target

- **What about Message Passing on a chip?**
  - MPI buffers & datastructures growing O(N) or $O(N^2)$ a problem for constrained memory
  - Redundant use of memory for shared variables and program image
  - *Flat view of parallelism doesn't make sense given hierarchical nature of multicore sys.*
- **What about SMP on a chip?**
  - Hybrid Model (MPI+OpenMP) : *Long and mostly unsuccessful history*
  - But it is NOT an SMP on a chip
    - 10-100x higher bandwidth on chip
    - 10-100x lower latency on chip
  - SMP model ignores potential for much tighter coupling of cores
  - *Failure to exploit hierarchical machine architecture will drastically inhibit ability to efficiently exploit concurrency! (requires code structure changes)*
- **Looking beyond SMP**
  - Cache Coherency: *necessary but not sufficient (and not efficient for manycore!)*
  - Fine-grained language elements difficult to build on top of CC protocol
  - Hardware Support for Fine-grained hardware synchronization
  - Message Queues: direct hardware support for messages
  - Transactions: Protect against incorrect reasoning about concurrency

About Transactions

- **What are Transactions**
  - Speculatively execute, but don't commit result to memory (stays resident in cache for HW-assisted transactions)
  - If another thread updated the same (conflicting) memory locations, then DO NOT commit results and re-execute (Rollback)
  - If no conflict occurred, then commit results to memory
- **Why transactions are good**
  - Can assume parallelization of loop iterations where every iteration is a transaction (Lay-Z-boy parallelization!)
  - If you reason incorrectly about dataflow hazards (read-after-write), then suffer slower performance, but still get the correct answer (very GOOD property)
  - Auto-parallelizing compilers can be more aggressive
- **Why transactions are bad *(a subset of leading issues)***
  - What does it mean to have a nested transaction?
  - What does it mean to mix transactional regions with non-transaction regions? *(Kozyrakis)*
  - How large can a transaction be? *(finite hardware resources)*

Auto-Tuning

- **We expect a lot from compilers (perhaps TOO much)**
  - We underestimate the amount of information compiler optimizers and back-ends have to work with
  - Many ambiguities at compile-time that are only resolved at runtime
  - Conservative to ensure "correctness"
  - Don't hold your breath waiting for "autoparallelization"
- **New approach: "Auto-tuners" 1st run variations of program on computer to heuristically search for best combinations of optimizations (blocking, padding, …) and data structures, then produce C code to be compiled for *that* computer**
  - E.g., PHiPAC (BLAS), Atlas (BLAS), Spiral (DSP), FFT-W
  - Can achieve 10X over conventional compiler
  - Encode body of knowledge regarding tuning strategies
- **Example: Sparse Matrix (SPMv) for 3 multicores**
  - Fastest SPMv: 2X OSKI/PETSc Clovertown, 4X Opteron
  - Optimization space: register blocking, cache blocking, TLB blocking, prefetching/DMA options, NUMA,
    BCOO v. BCSR data structures, 16b v. 32b indices, …

Community Codes & Frameworks

(hiding complexity using good SW engineering)

- **Frameworks (eg. Chombo, Cactus, SIERRA, UPIC, etc…)**
  - Clearly separate roles and responsibilities of your expert programmers from that of the domain experts/scientist/users (productivity layer vs. performance layer)
  - Define a *social* contract between the expert programmers and the domain scientists
  - Enforces and facilitates SW engineering style/discipline to ensure correctness
  - Hides complex domain-specific parallel abstractions from scientist/users to enable performance (hence, most effective when applied to community codes)
  - Allow scientists/users to code nominally serial plug-ins that are invoked by a parallel "driver" (either as DAG or constraint-based scheduler) to enable productivity
- **Properties of the "plug-ins" for successful frameworks (CSE07)**
  - Relinquish control of main(): invoke user module when framework thinks it is best
  - Module must be stateless
  - Module only operates on the data it is handed (no side-effects)
- **Frameworks can be thought of as driver for coarse-grained dataflow**
  - Very much like classic static dataflow, except coarse-grained objects written in declarative language (dataflow without the functional languages)
  - Broad flexibility to schedule Directed Graph of dataflow constraints
  - See Jack Dongarra & Parry Husbands' poster on DAG-based scheduling

Multicore Opportunities
(thinking about large numbers of cores)

- **Operating Systems**
  - Spatially partition cores instead of time multiplexing
  - "Side-cores" for OS services and interrupts (D.K. Panda)
- **Offload engines for efficient one-sided communication**
- **Truly asynchronous/background I/O**
- **Load balancing calculation and data movement**
  - Load-imbalance is looming impediment to future scalability
  - Currently creates load-imbalance by attempting to compute balance
  - Run in tandem with computations (background balancing) on dedicated cores
- **Exploiting on-chip bandwidth (dataflow)**
  - Rather than decomposing for SPMD parallelism, decompose laterally (feed-forward pipelines) to reuse on-chip bandwidth
  - Good: More general than streaming. Better exploitation of on-chip bandwidth and data locality!
  - Bad: Requires strict control of side-effects
  - Would benefit greatly from rediscovering dataflow and functional programming languages

Conclusions

- **Enormous transition is underway that affects all sectors of computing industry**
  - Motivated by power limits
  - Proceeding before emergence of the parallel programming model
- **Will lead to new era of architectural exploration given uncertainties about programming and execution model (and we MUST explore!)**
- **Need to get involved now**
  - 3-5 years for new hardware designs to emerge
  - 3-5 years lead for new software ideas necessary to support new hardware to emerge
  - 5+ MORE years to general adoption of new software

More Info

- The Berkeley View
  - http://view.eecs.berkeley.edu
- NERSC Science Driven System Architecture Group
  - http://www.nersc.gov/projects/SDSA