

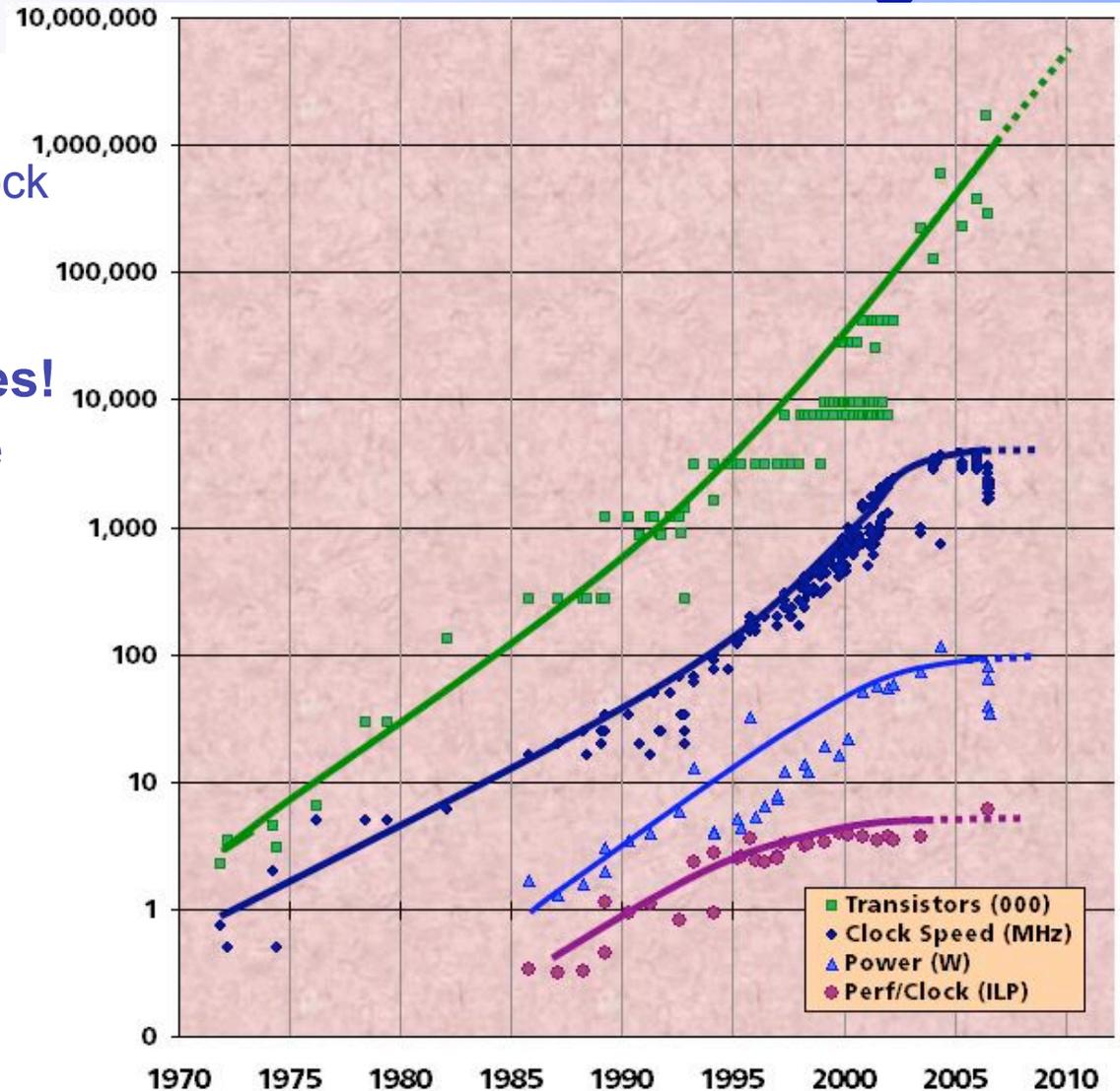
Technology Trends

John Shalf
July 2008



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!



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



What is Happening Now?

- **Moore's Law**

- Silicon lithography will improve by 2x every 18 months
- Double the number of transistors per chip every 18mo.

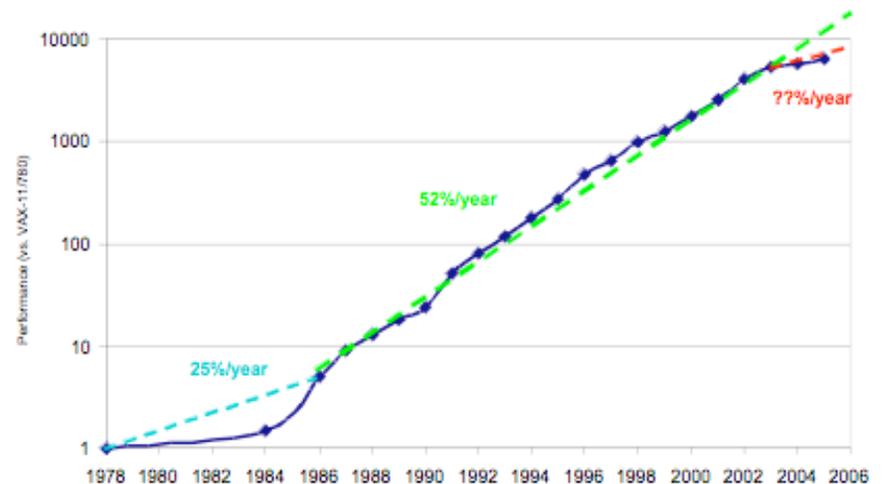
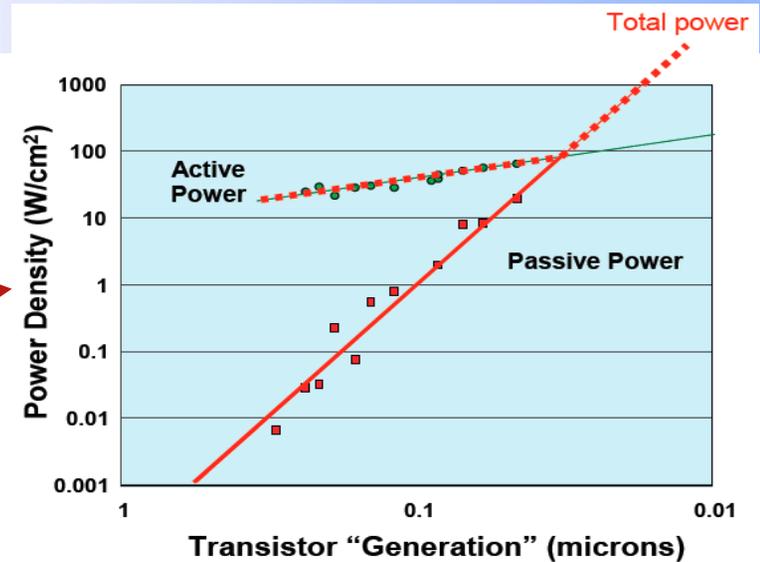
- **CMOS Power**

$$\text{Total Power} = \underbrace{V^2 * f * C}_{\text{active power}} + \underbrace{V * I_{\text{leakage}}}_{\text{passive power}}$$

- As we reduce feature size Capacitance (C) decreases proportionally to transistor size
- Enables increase of clock frequency (f) proportionally to Moore's law lithography improvements, with same power use
- This is called "Fixed Voltage Clock Frequency Scaling" (Borkar '99)

- **Since ~90nm**

- $V^2 * f * C \approx V * I_{\text{leakage}}$
- Can no longer take advantage of frequency scaling because passive power ($V * I_{\text{leakage}}$) dominates
- Result is recent clock-frequency stall reflected in Patterson Graph at right



What is Happening Now?

- **Moore's Law**

- Silicon lithography will improve by 2x every 18 months
- Double the number of transistors per chip every 18mo.

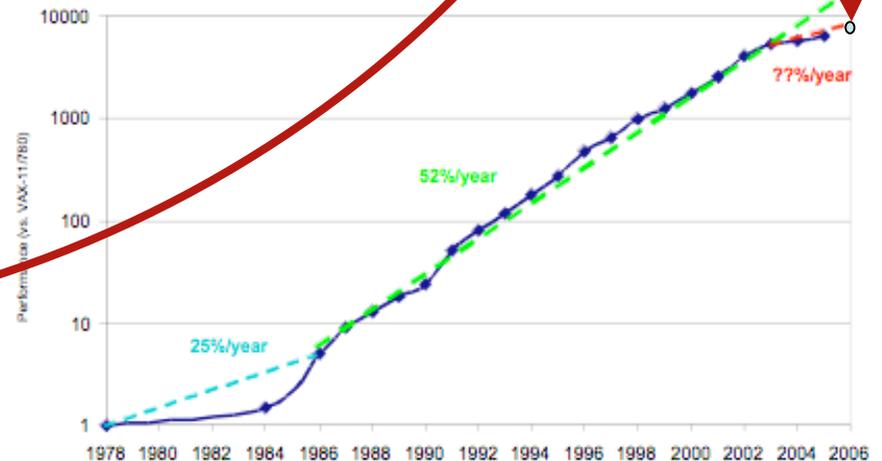
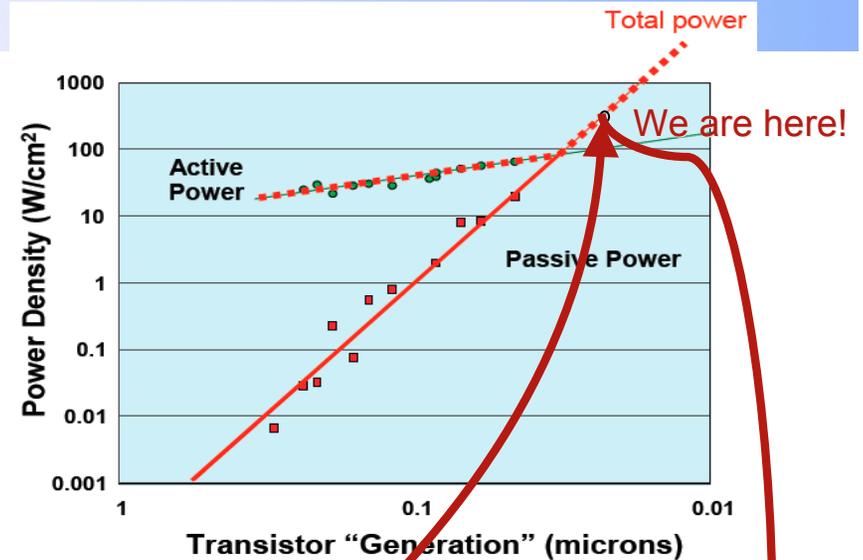
- **CMOS Power**

$$\text{Total Power} = \underbrace{V^2 * f * C}_{\text{active power}} + \underbrace{V * I_{\text{leakage}}}_{\text{passive power}}$$

- As we reduce feature size Capacitance (C) decreases proportionally to transistor size
- Enables increase of clock frequency (f) proportionally to Moore's law lithography improvements, with same power use
- This is called "Fixed Voltage Clock Frequency Scaling" (Borkar `99)

- **Since ~90nm**

- $V^2 * f * C \approx V * I_{\text{leakage}}$
- Can no longer take advantage of frequency scaling because passive power ($V * I_{\text{leakage}}$) dominates
- Result is recent clock-frequency stall reflected in Patterson Graph at right



SPEC_Int benchmark performance since 1978 from Patterson & Hennessy Vol 4.

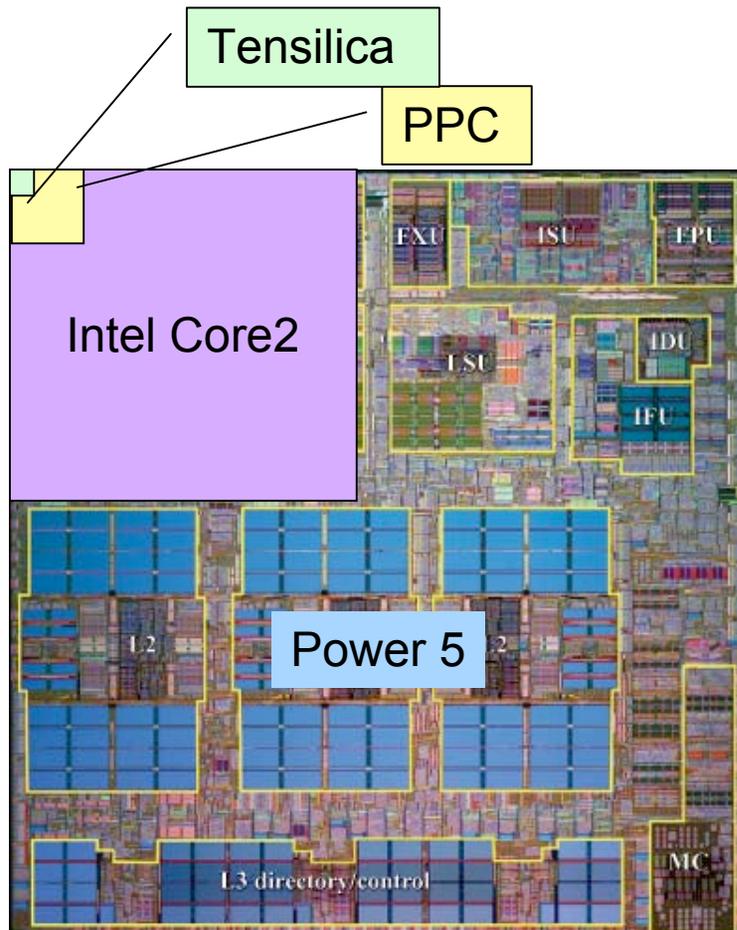




Hardware: What are the problems?

- **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)**
 - 389mm²
 - 120W@1900MHz
- **Intel Core2 sc (laptop)**
 - 130mm²
 - 15W@1000MHz
- **PowerPC 450 (autos and BG/P)**
 - 5mm²
 - 1W@800MHz
- **Tensilica DP (cell phones / printers)**
 - 0.8mm²
 - 0.09W@600MHz

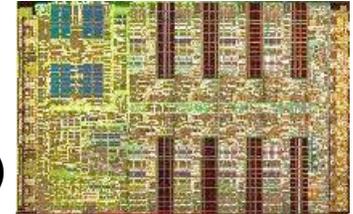
Each core operates at 1/3 to 1/10th computational efficiency of largest chip, but you can pack 100x more cores onto a chip and consume 1/20 the power⁵

Multicore vs. Manycore

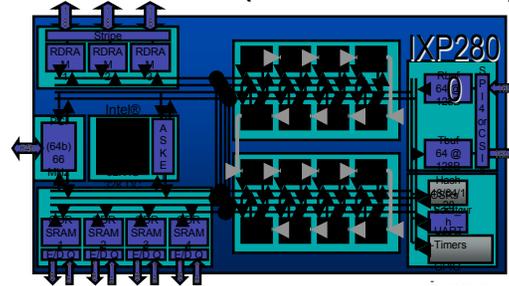
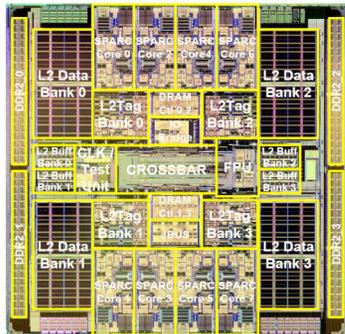
- **Multicore: current trajectory**
 - Stay with current fastest core design
 - Replicate every 18 months (2, 4, 8 . . . Etc...)
 - Advantage: Do not alienate serial workload
 - Example: AMD X2 (2 core), Intel Core2 Duo (2 cores), Madison (2 cores), AMD Barcelona (4 cores), Intel Tigerton (4 cores)
- **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

Convergence of Platforms

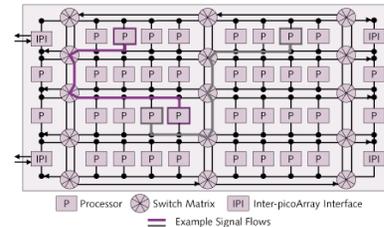
- Multiple parallel general-purpose processors (GPPs)
- Multiple application-specific processors (ASPs)



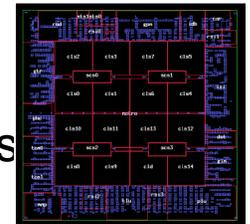
Intel Network Processor
1 GPP Core
16 ASPs (128 threads)



IBM Cell
1 GPP (2 threads)
8 ASPs



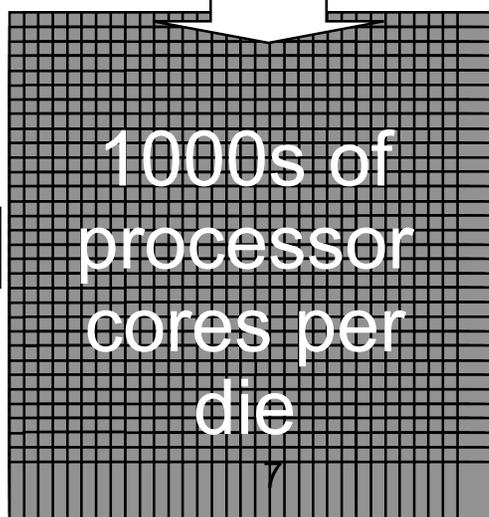
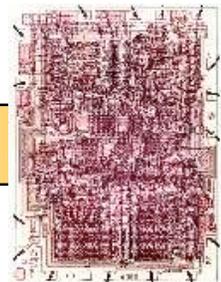
Picochip DSP
1 GPP core
248 ASPs



Cisco CRS-1
188 Tensilica GPPs

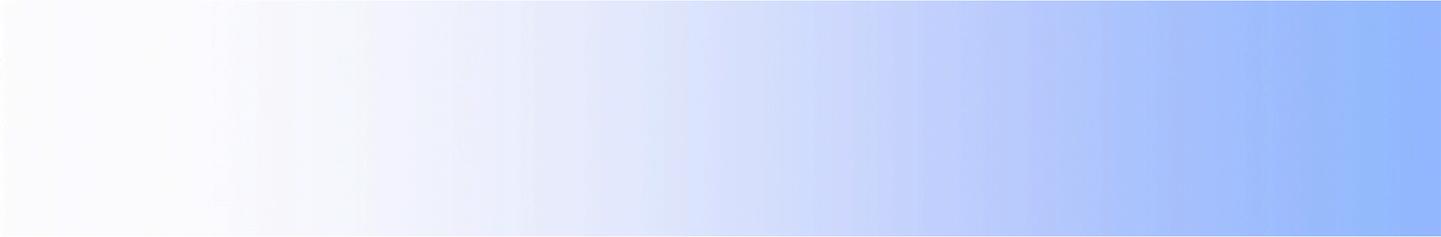
Sun Niagara
8 GPP cores (32 threads)

Intel 4004 (1971):
4-bit processor,
2312 transistors,
~100 KIPS,
10 micron PMOS,
11 mm² chip



***“The Processor is
the new Transistor”***

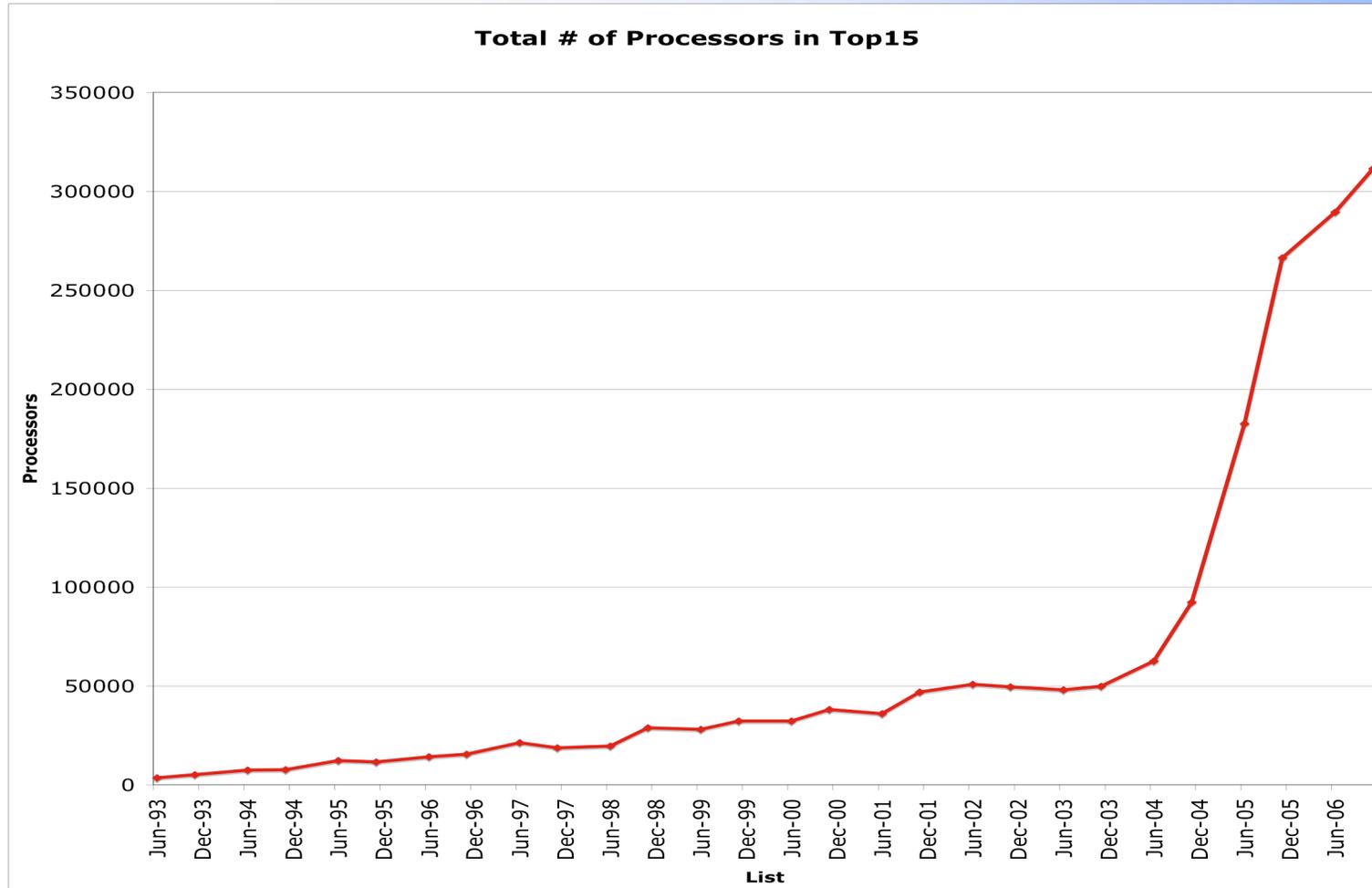
[Chris Rowen]



Ramifications of Massive Parallelism



The Future of HPC System Concurrency



Must ride exponential wave of increasing concurrency for foreseeable future!



Concerns about Multicore

(in the context of HPC)

- **Programmability:** How can I possibly program 1M+ cores in an effective manner?
- **Reliability:** More “moving parts” means more opportunity for failures
- **System Balance:** Concern that memory and interconnect performance will ultimately cap multicore performance



Programmability



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)*
- **Entering transition period for programming models**



NERSC-6 Response to Uncertainty in Programming Model

- **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

- **NERSC-6 new “Full Fury” benchmark rules**

- Supports innovation as a response to uncertainty regarding future programming environments
- Allows, SMP, CMP, accelerators, and novel programming models
- Evaluated both in terms of concrete performance improvement over baseline AND ease of use to achieve that performance



Application Community's Response to Technology Trends

- **Parallel computing has thrived on weak-scaling for past 15 years**
- **Flat CPU performance increases emphasis on strong-scaling**
- **Workload Requirements will change accordingly**
 - Concurrency will increase proportional to system scale (3-5x increase over NERSC-5)
 - Timestepping algorithms will be increasingly driven towards implicit or semi-implicit stepping schemes
 - Multiphysics/multiscale problems increasingly rely on spatially adaptive approaches such as Berger-Oliger AMR
 - Strong scaling will push applications towards smaller messages sizes – requiring lighter-weight messaging



NERSC-6 Response To Trends

- **Parallel computing has thrived on weak-scaling for past 15 years**
- **Flat CPU performance increases emphasis on strong-scaling**
- **NERSC-6 Benchmarks changed accordingly**
 - Increased concurrency 4x over NERSC-5 benchmarks
 - Input decks emphasize strong-scaled problems
 - Emphasis on implicit methods
 - New AMR benchmark
 - New UPC benchmark



Compiler Technology

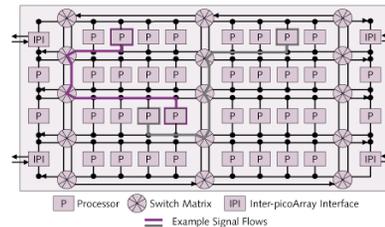
Faced with increased architectural diversity



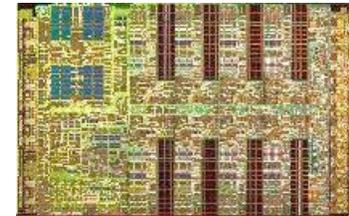
Performance Portability

- **Diverse set of architectural options == Daunting tuning requirements**
- **Performance portability was bad enough**
 - Diversity makes performance portability tough
 - In many cases, basic portability is lost
 - Need new approaches such as multi-target languages, auto-tuning and/or code generators

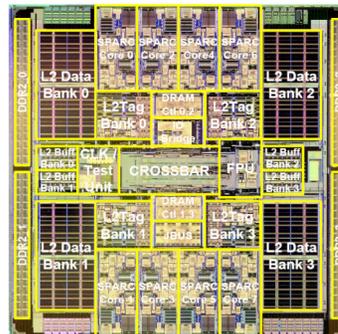
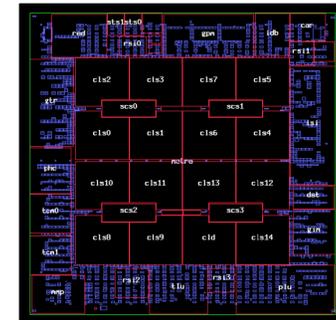
Picochip DSP
 1 GPP core
 248 ASPs



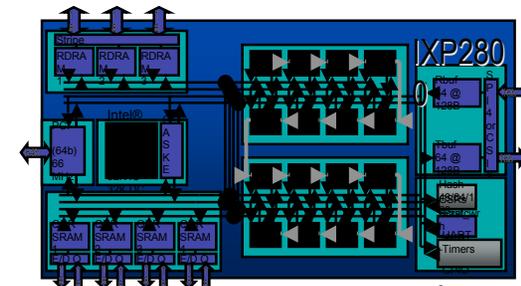
STI Cell
 8 ASPs, 1GPP



Cisco CRS-1
 188 Tensilica GPPs



Sun Niagara
 8 GPP cores (32 threads)

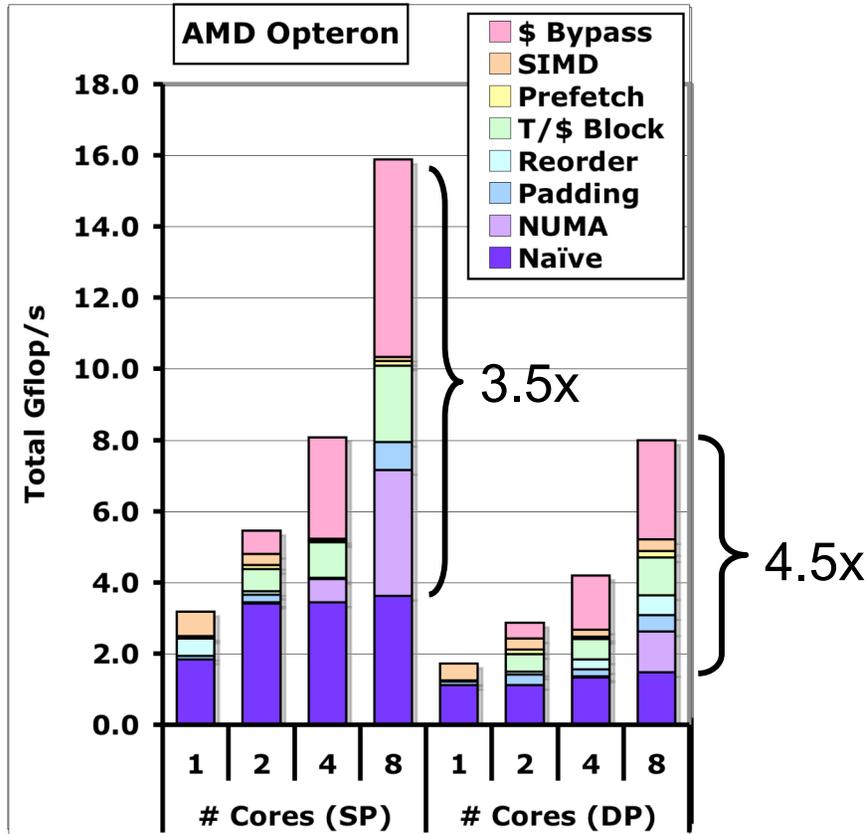


Intel Network Processor
 1 GPP Core
 16 ASPs (128 threads)

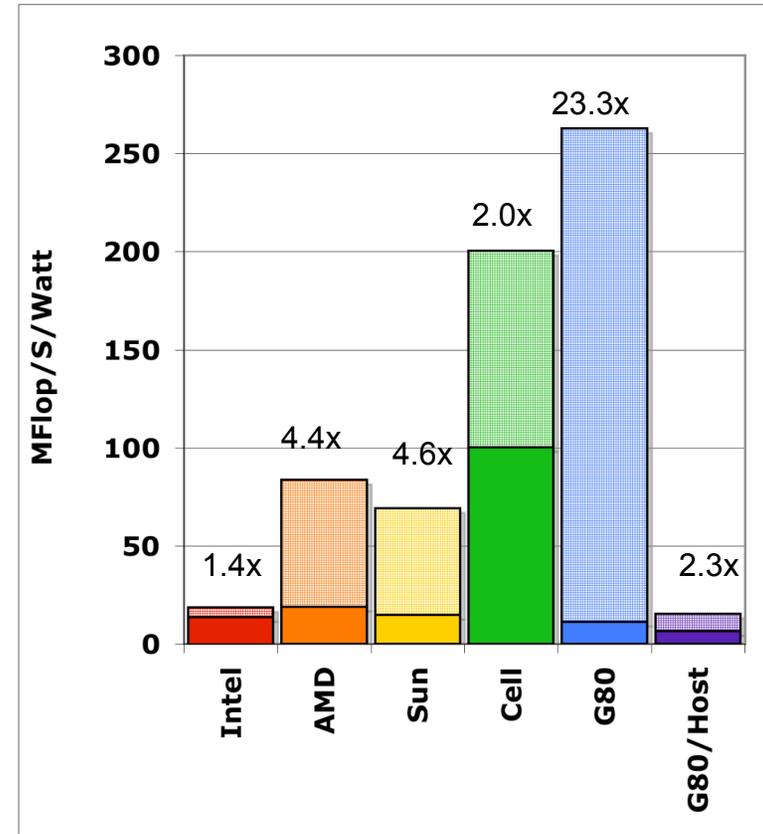
Multiprocessor Performance

(auto-tuned stencil kernel)

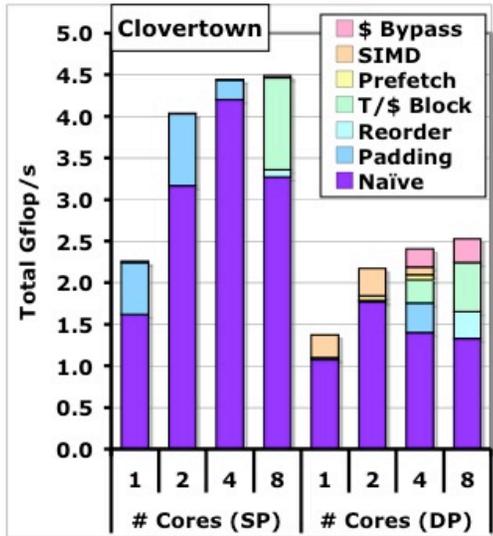
Performance Scaling



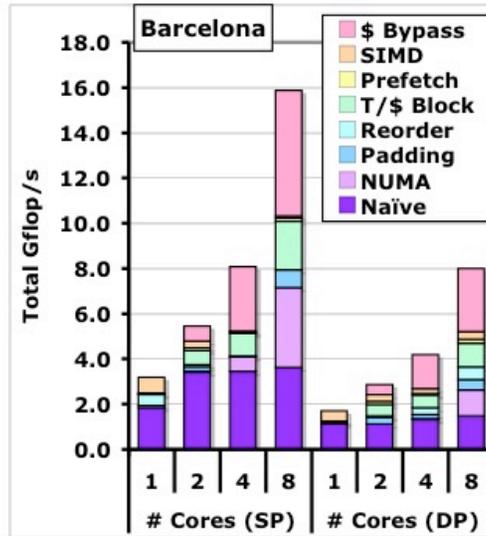
Power Efficiency



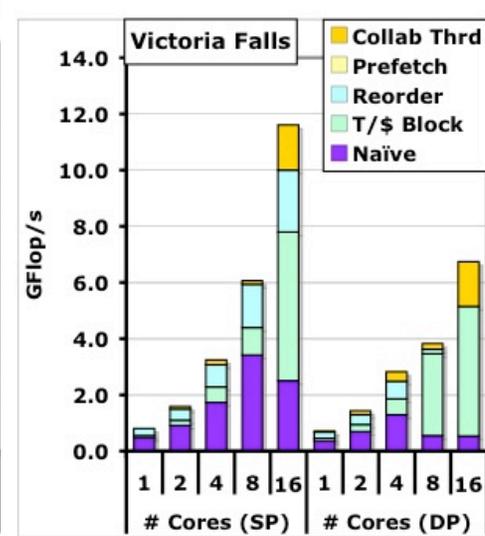
Performance Portability



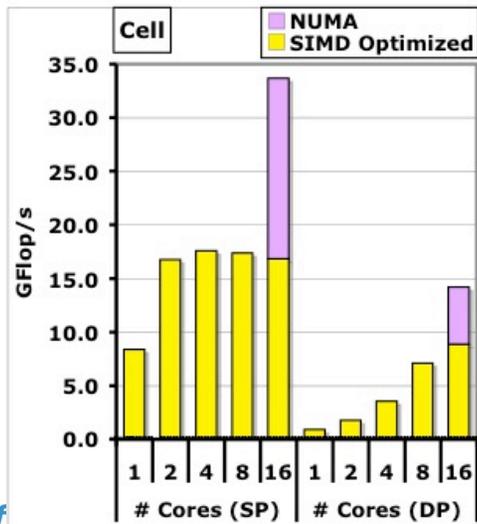
(a)



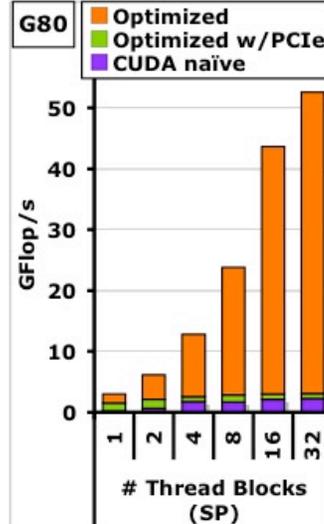
(b)



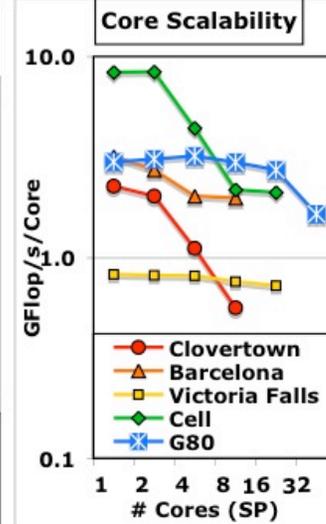
(c)



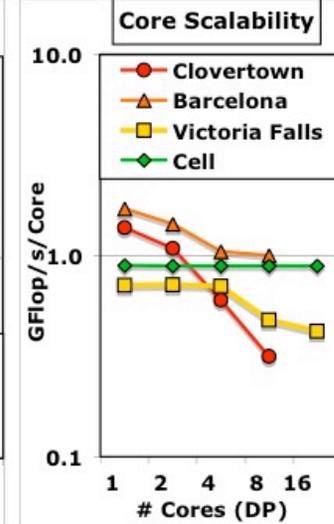
(d)



(e)



(f)



(g)



Reliability





Reliability

- **Hardware FIT rates**
 - Yes: as transistors get smaller, reliability is worse
 - Circuit designers have headroom to mitigate reliability issues (just more cost and less performance)
 - x86 chip manufacturers are not motivated to make their chip more reliable than MS Windows. (but not related to device scale)
- **Strategy**
 - Adopt rigorous metrics to reliability and availability as part of contract requirements
 - Do not push more reliability management (aside from checkpointing) onto application programmers.
- **Concerns**
 - Increased rate of silent errors that are not flagged by hardware protections (CERN and NCSA studies)
 - CRC checks on your data files (CERN study on SW data corruption)

Chips are designed to not exceed reliability of OS!

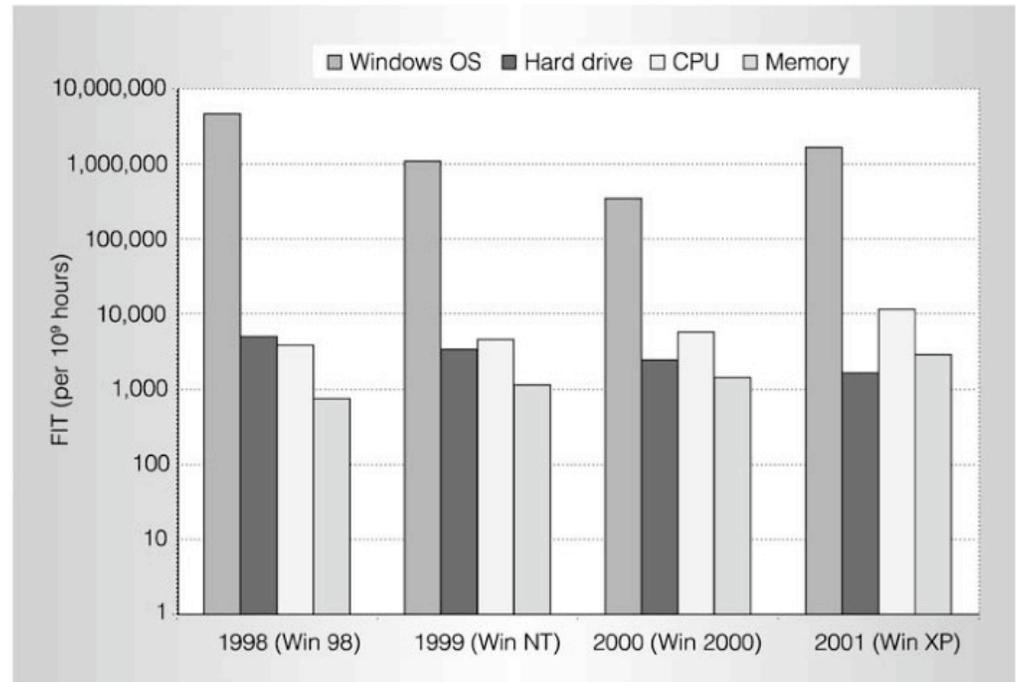
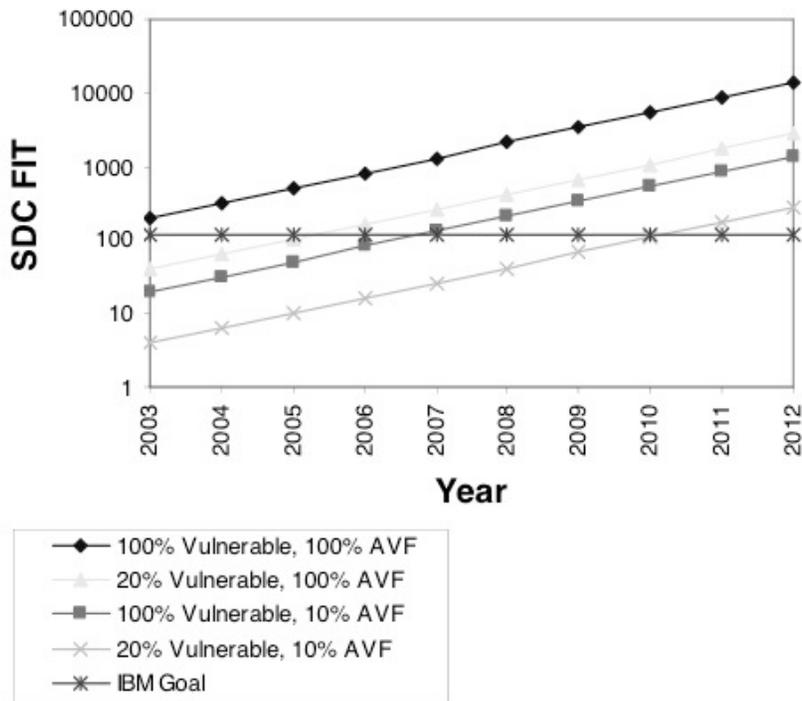


Figure 2. Failures in billions of hours of operation.²⁻⁵

- Majority of tracked system-wide outages at NERSC related to software failures
- RFP requires vendors explicitly track root cause (software vs. hardware failures)

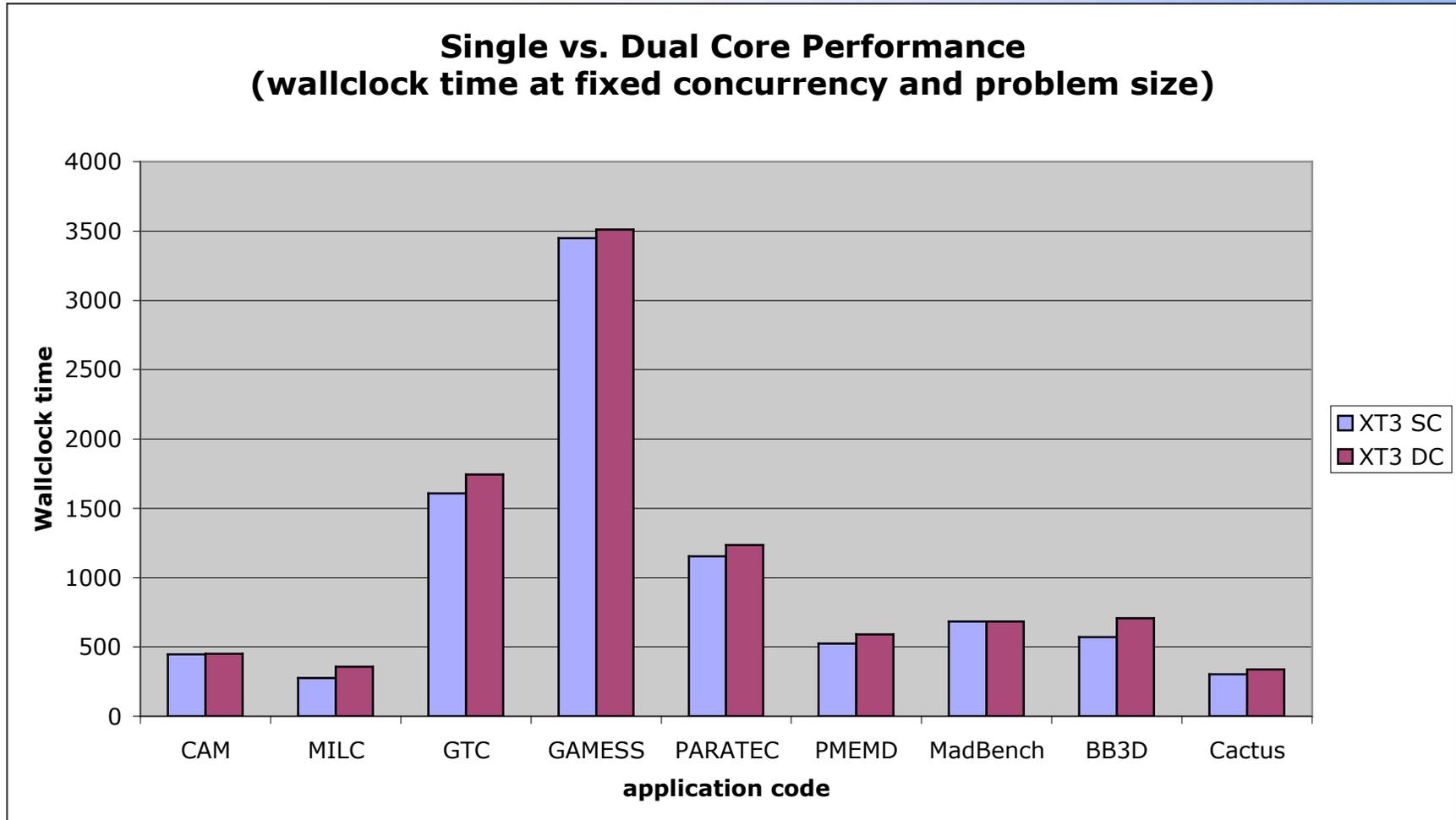


Memory Bandwidth



Memory Bandwidth

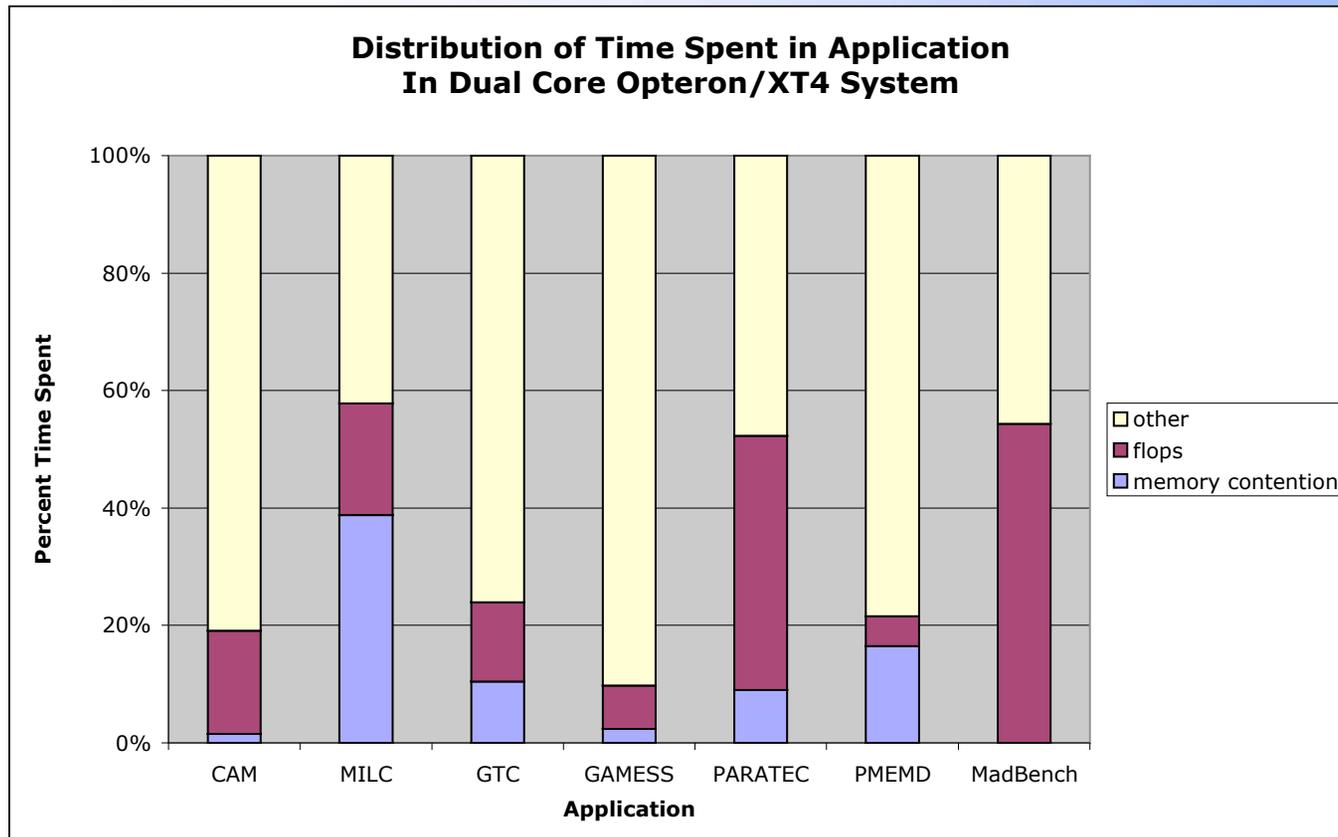
(not stressed by untuned apps)



Many people believe applications are memory bandwidth bound, but they usually aren't. Compilers just can't do the job (need autotuning for performance portability)

Memory Bandwidth

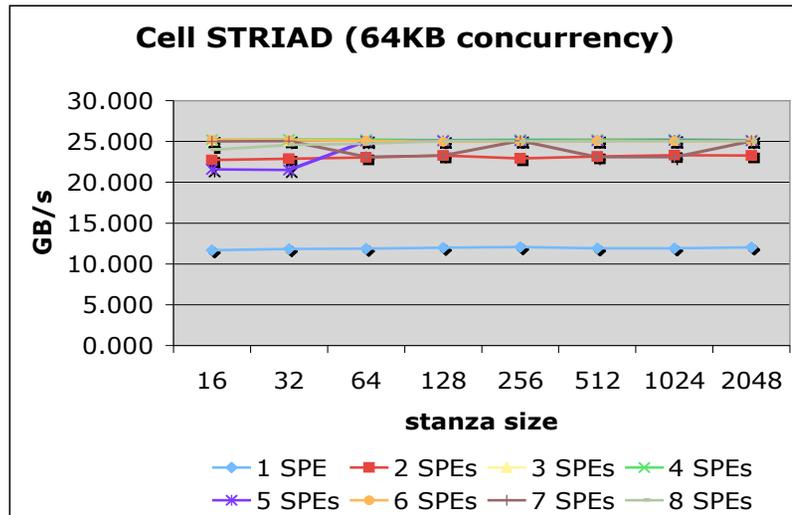
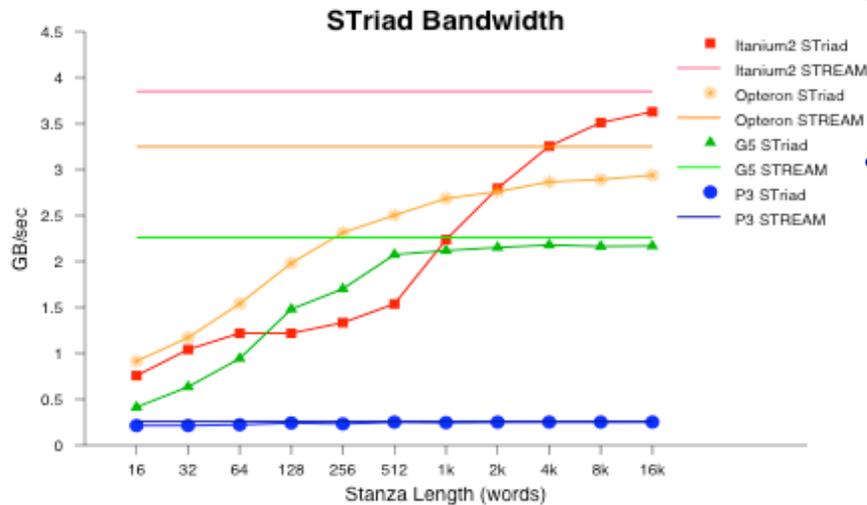
(maintaining system balance)



- 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)

About Latency

(goes hand-in-hand-with bandwidth)



- **Little's Law: $BW * latency = concurrency$**
 - $bandwidth * latency = \#memory_fetches_in_flight$
- **For Power5+ single-core: $120ns * 25$ Gigabytes/sec**
 - 3000 bytes of data must be in flight to balance Little's Law!
 - 375 DP operands (> number of registers and OOO depth!)
 - 23.4 cache lines (cache line size cannot hide the latency)
- **Various ways to manipulate memory fetch concurrency**
 - 2x memory bandwidth: Need 6000 bytes/flight
 - 2x cores: Each only needs 1500 bytes/flight
 - 2 threads/core: Each needs 750 bytes/flight
 - 128 slower cores/threads?: 24 bytes in flight (3 DP words)
 - Vectors (*not SIMD!*): 64-128 words per vec load (1024 bytes)
 - Software Controlled Memory: multi-kilobytes/DMA (eg. Cell, ViVA)



Concerns About Memory And Interconnects

- **Growing Memory Power Consumption**
 - Memory poised to consume more power than CPU (GDDR3 on GPU consumes 50W for 256Megs!!!)
 - Its not the memory cells (it's sense amps and device interface burning the power)
 - *Motivates changes in memory packaging technology (memory stacking and photonics)*
- **Balancing Little's Law**
 - Concurrency = Bandwidth * Latency
 - Latency is fixed and Concurrency is growing
 - Copper: Power consumption is proportional to length*signal_rate *demands spatially localized communication*
 - Optical: Reduces power as a function of length *Flat Bandwidth model* Easier to balance $C=B*L$ where constraint is fixed latency and growing concurrency



Power Trends



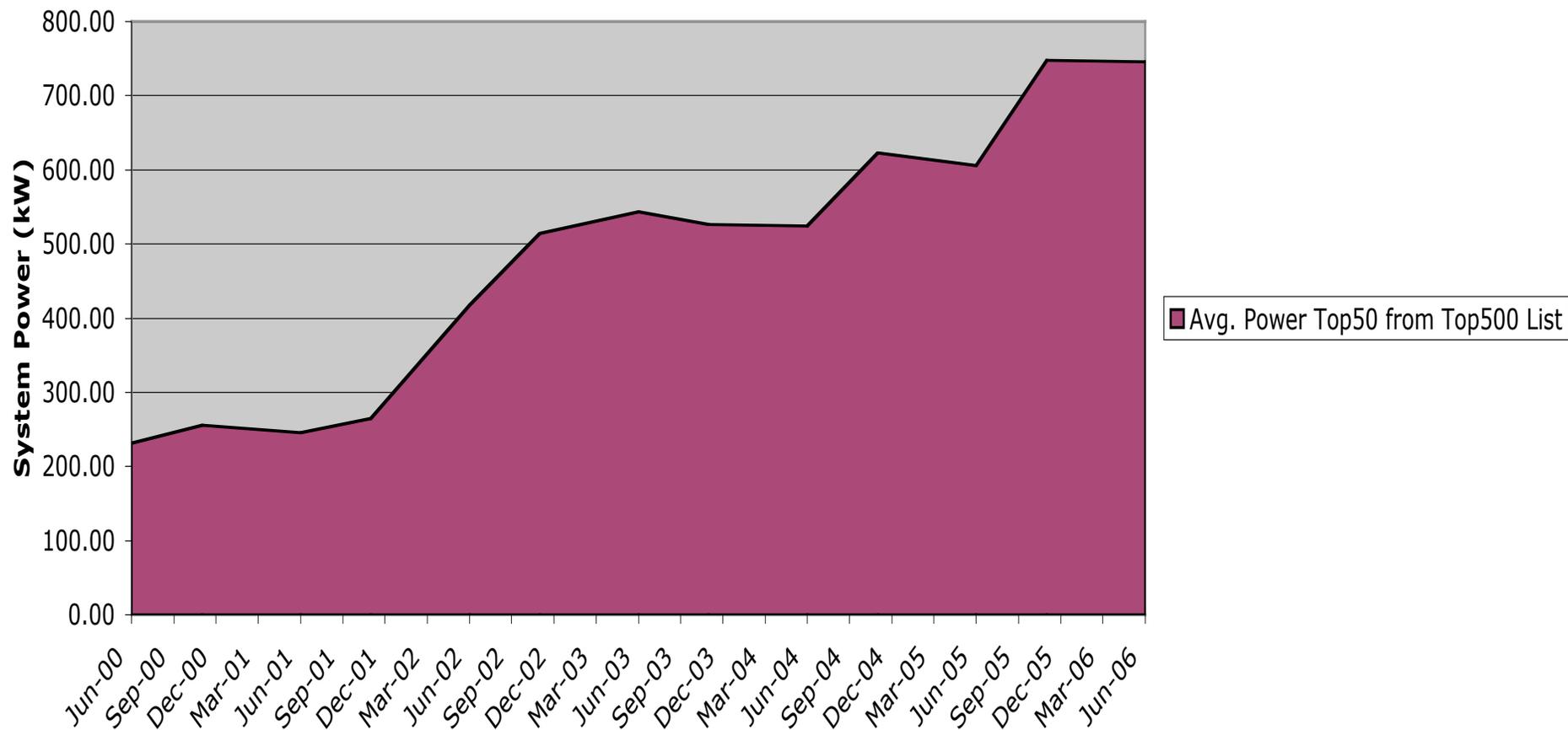


New Design Constraint: *POWER*

- **Transistors still getting smaller**
 - Moore's Law is alive and well
- **But Dennard scaling is dead**
 - No power efficiency improvements with smaller transistors
 - No clock frequency scaling with smaller transistors
 - All “magical improvement of silicon goodness” has ended

Power Consumption by Top500 Systems

**Growth in Power Consumption (Top50)
Excluding Cooling**





Other Power Growth Estimates

- **Baltimore Sun (Jan 23, 2007): NSA drawing 65-75 MW in Maryland**
 - Crisis: Baltimore Gas & Electric doesn't have power for city of Baltimore!
 - Expected to increase by 10-15 MW next year
- **LBL IJHPCA Study for ~1/5 Exaflop for Climate Science in 2008**
 - 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 to exascale in 2016
 - Estimate: 130 MW
- **DARPA Study**
 - More detailed assessment of component technologies
 - Estimate: 20 MW just for memory alone, 60 MW aggregate extrapolated from current design trend



NERSC-6 Response To Power Trends

- **New emphasis on power efficiency**
 - 3.5 MW power limit for Oakland Facility (OSF)
 - Require 480VAC 3-phase power distribution for efficiency
 - Increased cooling efficiency if systems operate at high-end of ASHRAE allowable temperature range
- **Memory is also increasing source of power consumption**
 - Expect bids with constrained memory
 - Modified benchmark rules to allow strong-scaling to accommodate constrained memory
- **Manycore offers good performance/joule**
 - Anticipate some bids with “accelerators” for NERSC-6
 - Have modified benchmark rules to accommodate (“Full Fury”)



Evolving OS Requirements





Challenging Old OS Assumptions

- **Assumes limited number of CPUs that must be shared**
 - *Old OS: time-multiplexing (context switching and cache pollution)*
 - *New OS: spatial partitioning*
- **Greedy allocation of finite I/O device interfaces (eg. 100 cores go after the network interface simultaneously)**
 - *Old OS: First process to acquire lock gets device (resource/lock contention Nondeterm delay)*
 - *New OS: QoS management for symmetric device access*
- **Background task handling via threads and signals**
 - *Old OS: Interrupts and threads (time-multiplexing) (inefficient!)*
 - *New OS: side-cores dedicated to DMA and async I/O*
- **Fault Isolation**
 - *Old OS: CPU failure --> Kernel Panic (will happen with increasing frequency in future silicon)*
 - *New OS: CPU failure --> Partition Restart (partitioned device drivers)*
- **Inter-Processor Communication**
 - *Old OS: invoked for ANY interprocessor communication or scheduling*
 - *New OS: direct HW access mediated by hypervisor*



More Info

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