Lecture 8
Thread Level Parallelism (2)

EEC 171 Parallel Architectures

John Owens
UC Davis

Credits

- © John Owens / UC Davis 2007–17.
- Thanks to many sources for slide material: Computer Organization and Design (Patterson & Hennessy) © 2005, Computer Architecture (Hennessy & Patterson) © 2007, Inside the Machine (Jon Stokes) © 2007, © Dan Connors / University of Colorado 2007, © Kathy Yelick / UCB 2007, © Wen-Mei Hwu/David Kirk, University of Illinois 2007, © David Patterson / UCB 2003–7, © John Lazzaro / UCB 2006, © Mary Jane Irwin / Penn State 2005, © John Kubiatowicz / UCB 2002, © Krste Asinovic/Arvind / MIT 2002, © Morgan Kaufmann Publishers 1998.

Cost of a Modern Datacenter

insideHPC > HPC > NSA to Build $900 Million HPC Datacenter

NSA to Build $900 Million HPC Datacenter

04.21.2011

Like

J. Nicholas Hoover writes in Information week that the NSA plans to build powerful, energy efficient, secure HPC datacenter near agency headquarters in Maryland by December 2015. The price tag will be nearly $900 Million USD.

> *"The specs for the new supercomputing center read much like the NSA is building a massive data center, with typical requirements for raised flooring, chilled water systems, fire suppression, and alarms. Power requirements are 60 megawatts, equivalent to the power requirements of Microsoft's recently completed 700,000 square foot data center in Chicago, which itself is one of the largest ever constructed."*

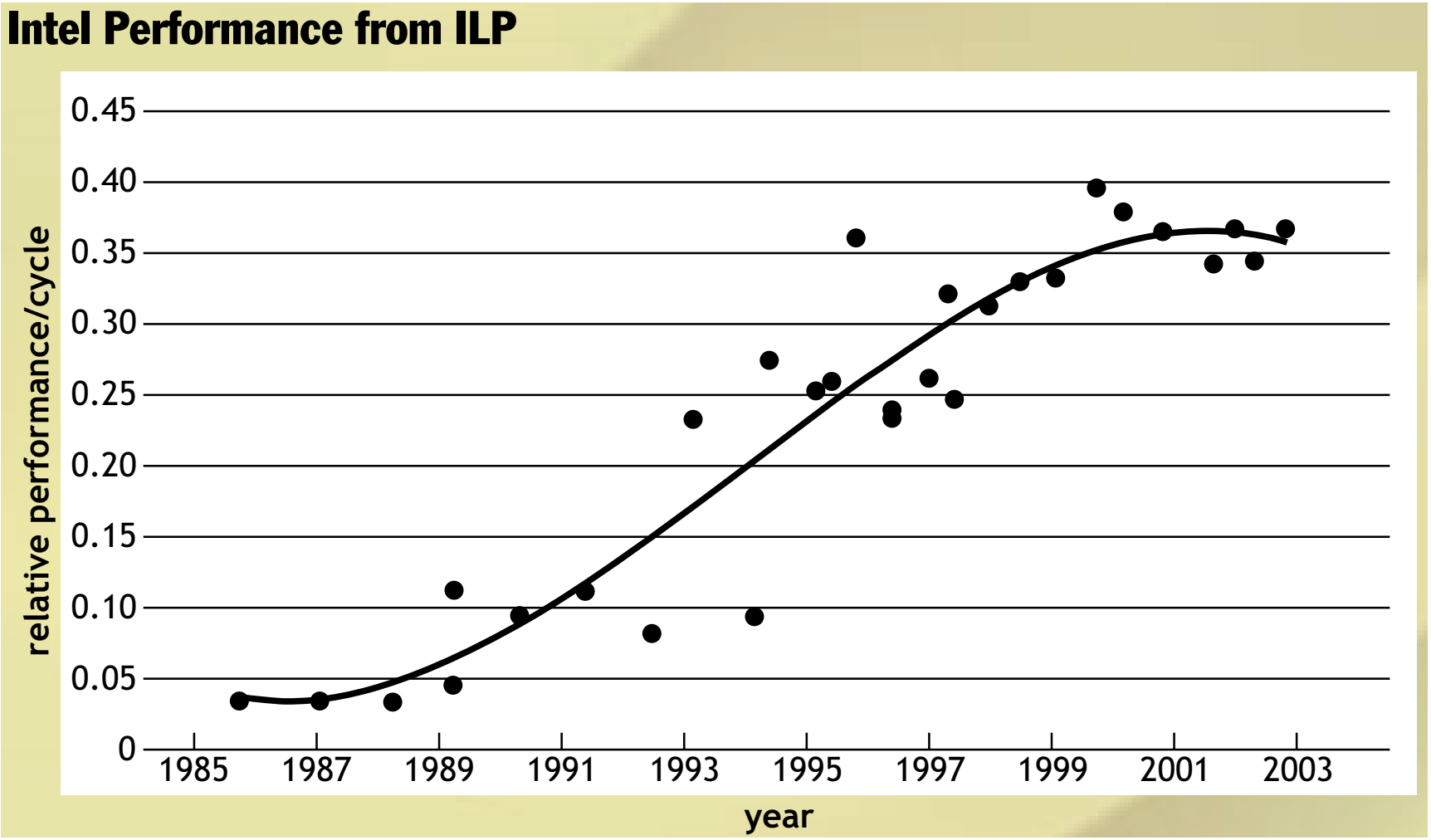
$100 $/MW.h * 60 MW * 8766 h/yr = $52.6M/yr

Tianhe-2 is 18 MW

Outline

- Google’s Architecture
- Sun T1 (Niagara) & Cray Eldorado
- Basics of parallel programming

ILP reaching limits



- Olukotun and Hammond, “The Future of Microprocessors”, *ACM Queue*, Sept. 2005

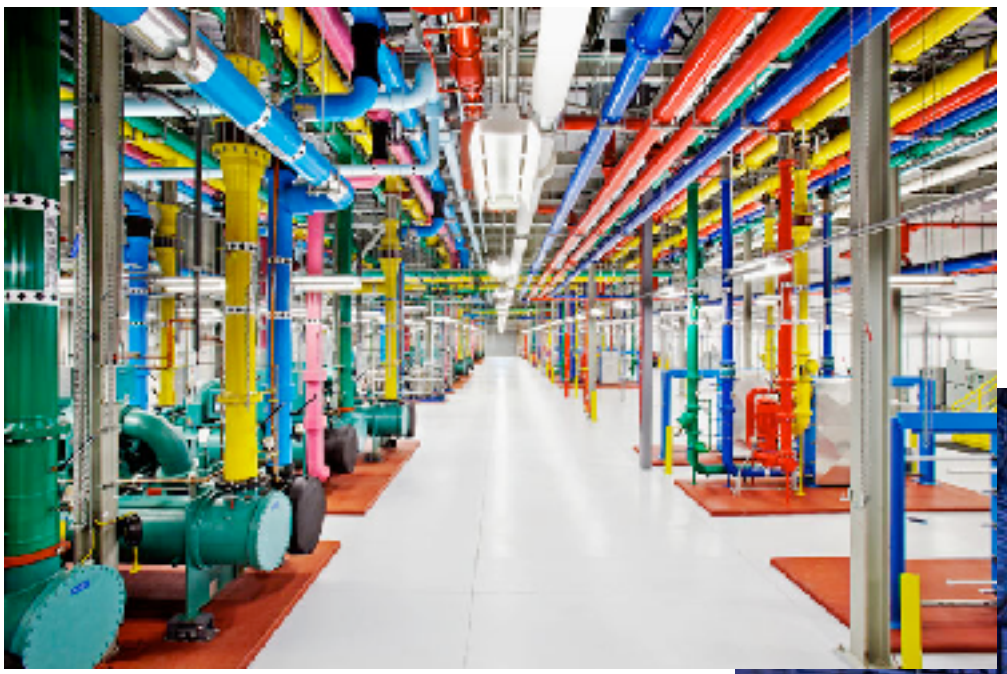
Olukotun's view

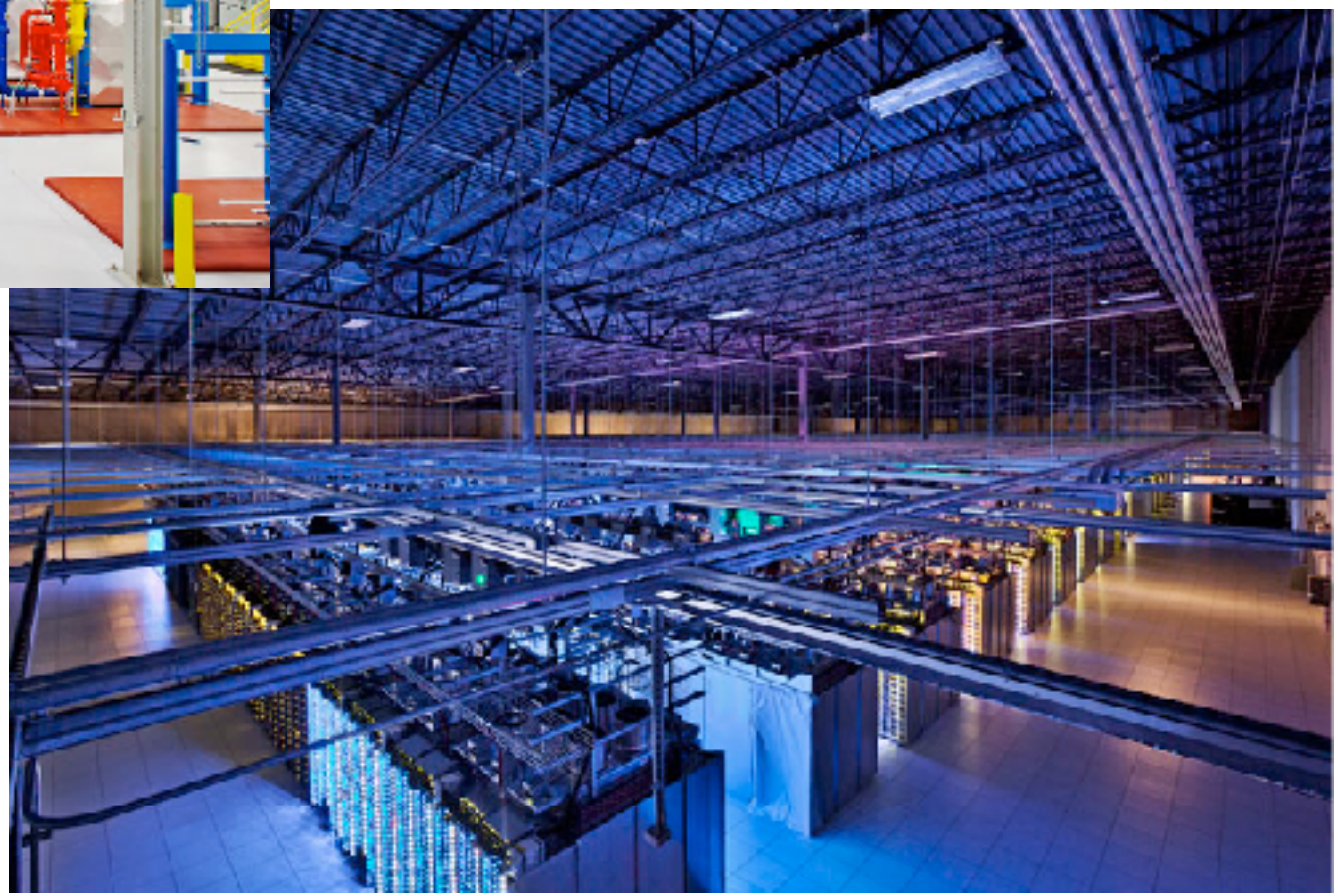
- "With the exhaustion of essentially all performance gains that can be achieved for 'free' with technologies such as superscalar dispatch and pipelining, we are now entering an era where programmers must switch to more parallel programming models in order to exploit multi-processors effectively, if they desire improved single-program performance."

Olukotun (pt. 2)

- “This is because there are only three real ‘dimensions’ to processor performance increases beyond Moore’s law: clock frequency, superscalar instruction issue, and multiprocessing. We have pushed the first two to their logical limits and must now embrace multiprocessing, even if it means that programmers will be forced to change to a parallel programming model to achieve the highest possible performance.”

Google data centers





http://www.wired.com/wiredenterprise/2012/10/ff-inside-google-data-center/all/

Google's Architecture

- "Web Search for a Planet: The Google Cluster Architecture"
  - Luiz André Barroso, Jeffrey Dean, Urs Hölzle, Google
- Reliability in software not in hardware
- 2003: 15k commodity PCs
- July 2006 (estimate): 450k commodity PCs
  - $2M/month for electricity
- Most interesting slides: http://www.slideshare.net/hasanveldstra/the-anatomy-of-the-google-architecture-fina-lv11 (2009) (estimate: 800k PCs)

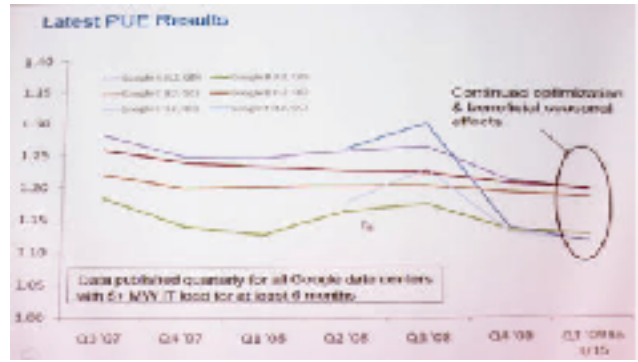
Goal: Price/performance

- “We purchase the CPU generation that currently gives the best performance per unit price, not the CPUs that give the best absolute performance.”
  - Google rack: 40–80 x86 servers
    - “Our focus on price/performance favors servers that resemble mid-range desktop PCs in terms of their components, except for the choice of large disk drives.”
    - 4-processor motherboards: better perf, but not better price/perf
    - SCSI disks: better perf and reliability, but not better price/perf
  - Depreciation costs: $7700/month; power costs: $1500/month
    - Low-power systems must have equivalent performance

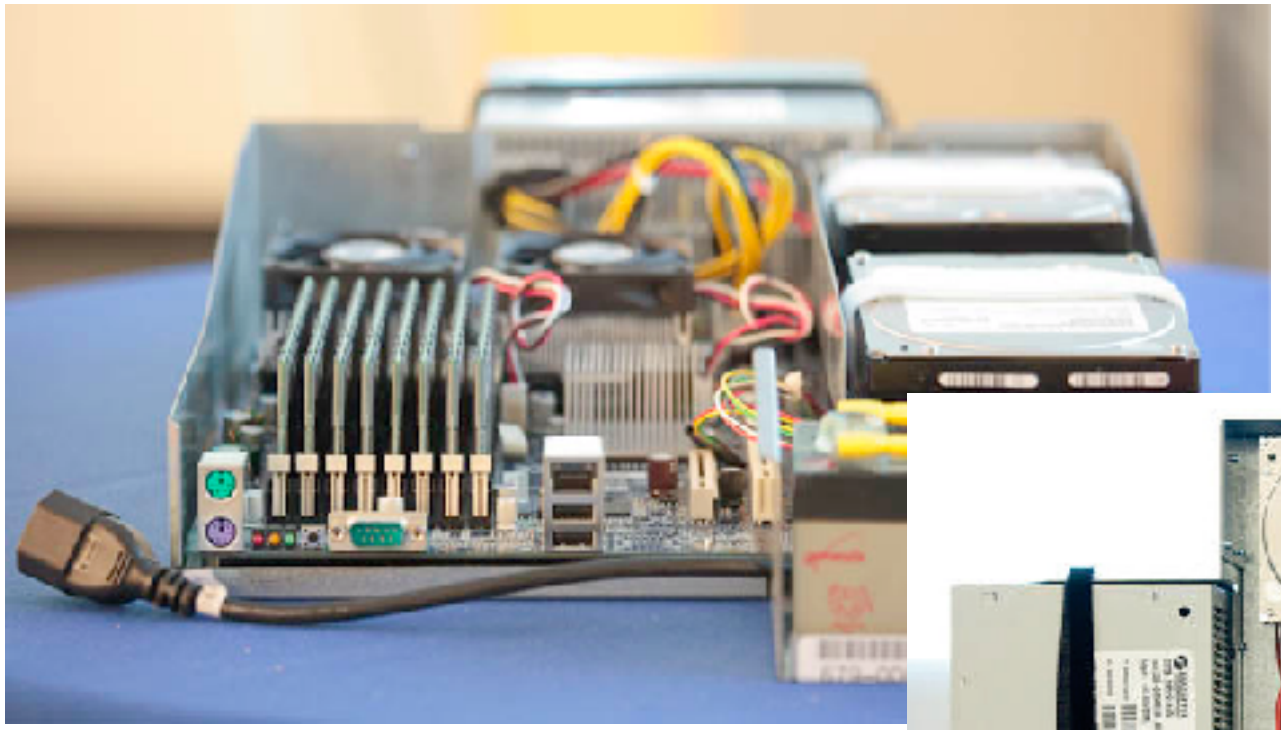
Google power density

- Mid-range server, dual 1.4 GHz Pentium III: 90 watts
  - 55 W for 2 CPUs
  - 10 W for disk drive
  - 25 W for DRAM/motherboard
    - so 120 W of AC power (75% efficient)
- Rack fits in 25 ft²
  - 400 W/ft²; high end processors 700 W/ft²
  - Typical data center: 70–150 W/ft²
  - Cooling is a big issue

"The standard measurement of data center efficiency is called power usage effectiveness, or PUE. A perfect number is 1.0, meaning all the power drawn by the facility is put to use. Experts considered 2.0—indicating half the power is wasted—to be a reasonable number for a data center. Google was getting an unprecedented 1.2."



Google Server



"By going homebrew and eliminating unneeded components, Google built a batch of servers for about $1,500 apiece, instead of the then-standard $5,000."

- 3.5 inches thick (2u)
- 2 CPUs (Intel or AMD), 2 hard drives, 8 memory slots
- 1 12-volt battery (!) [UPS: 15% of total energy]

http://news.cnet.com/8301-1001_3-10209580-92.html

Google Data Center

- 1AAA shipping container (1160 servers), 30 racks
- 250 KW

http://news.cnet.com/8301-1001_3-10209580-92.html

Google Workload (1 GHz P3) (2003)

Table 1. Instruction-level measurements on the index server.

Characteristic	Value
Cycles per instruction	1.1
Ratios (percentage)	
Branch mispredict	5.0
Level 1 instruction miss*	0.4
Level 1 data miss*	0.7
Level 2 miss*	0.3
Instruction TLB miss*	0.04
Data TLB miss*	0.7

* Cache and TLB ratios are per instructions retired.

Details of workload

- “Moderately high CPI” (P3 can issue 3 instrs/cycle)
  - “Significant number of difficult-to-predict branches”
  - Same workload on P4 has “nearly twice the CPI and approximately the same branch prediction performance”
- “In essence, there isn’t that much exploitable instruction-level parallelism in the workload.”
- “Our measurements suggest that the level of aggressive out-of-order, speculative execution present in modern processors is already beyond the point of diminishing performance returns for such programs.”

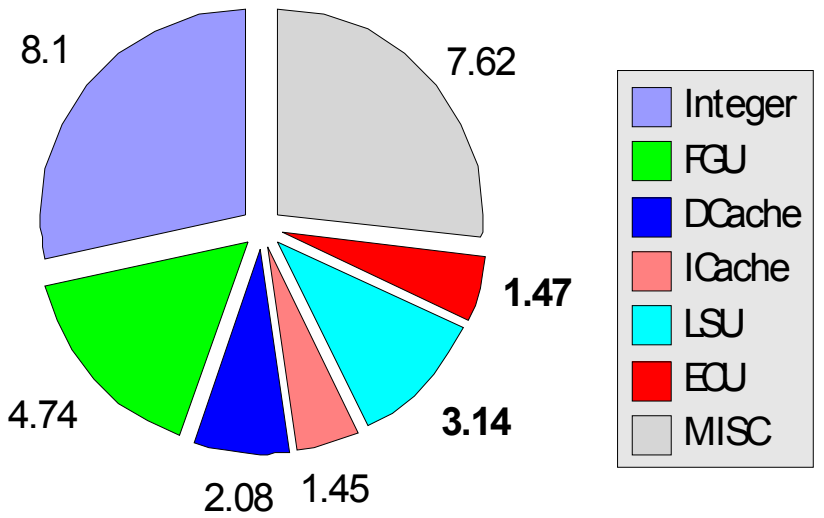
Google and SMT

- “A more profitable way to exploit parallelism for applications such as the index server is to leverage the trivially parallelizable computation.”
- “Exploiting such abundant thread-level parallelism at the microarchitecture level appears equally promising. Both simultaneous multithreading (SMT) and chip multiprocessor (CMP) architectures target thread-level parallelism and should improve the performance of many of our servers.”
- “Some early experiments with a dual-context (SMT) Intel Xeon processor show more than a 30 percent performance improvement over a single-context setup.”

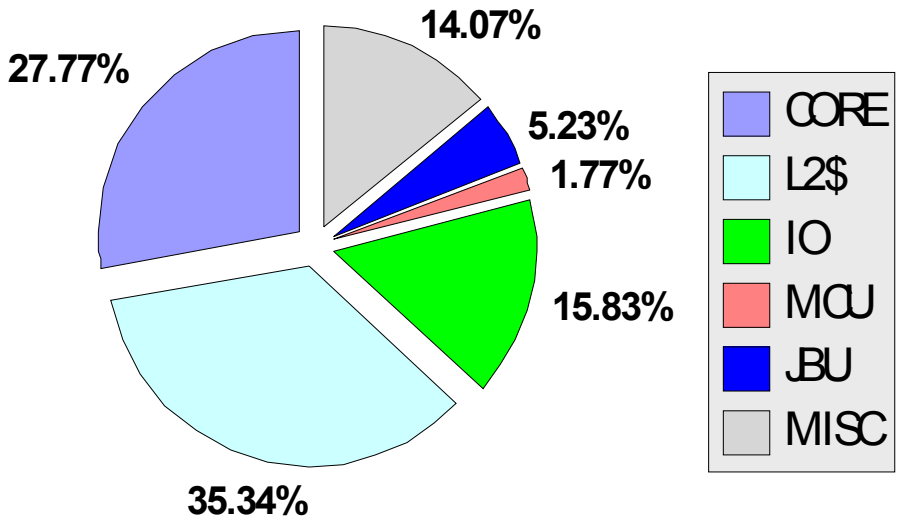
CMP: Chip Multiprocessing

- First CMPs: Two or more conventional superscalar processors on the same die
  - UltraSPARC Gemini, SPARC64 VI, Itanium Montecito, IBM POWER4
- One of the most important questions: What do cores share and what is not shared between cores?

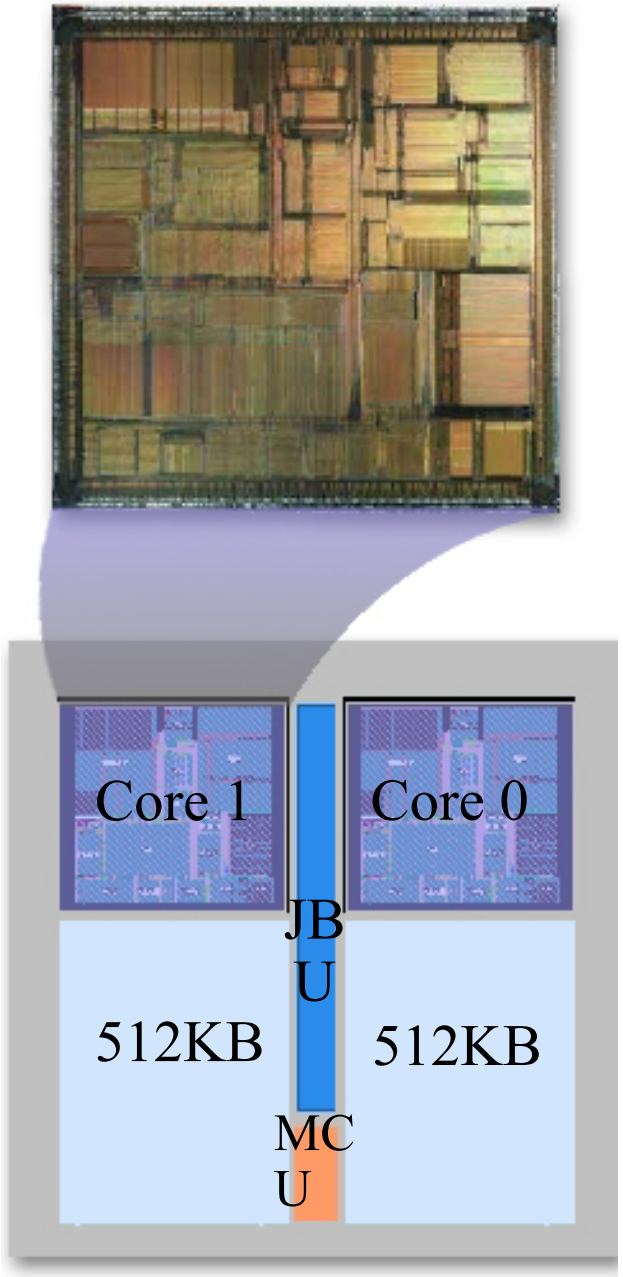
UltraSPARC Gemini



Core Area = 28.6 mm^2

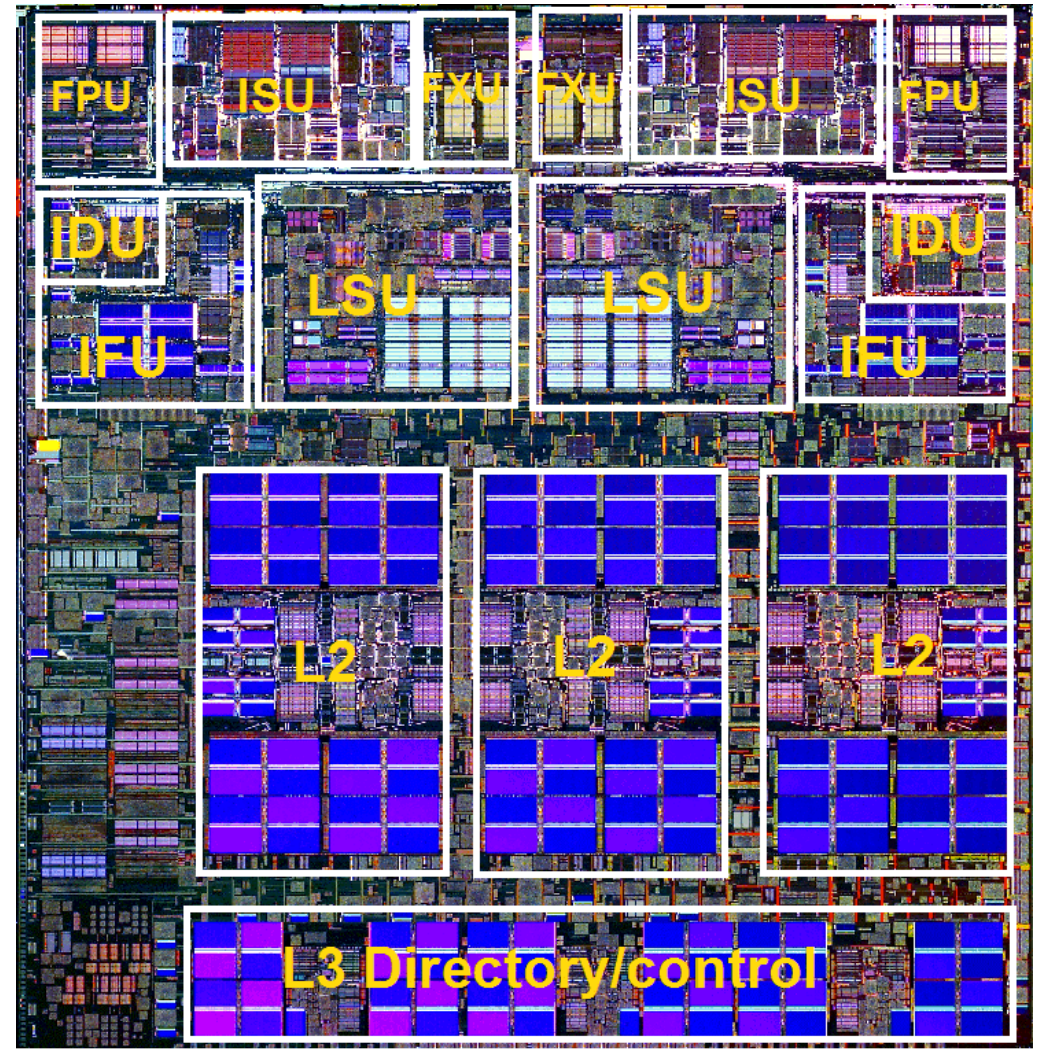


CPU Area = 206 mm^2



POWER5

- Technology: 130nm lithography, Cu, SOI
- Dual processor core
- 8-way superscalar
- Simultaneous multithreaded (SMT) core
  - Up to 2 virtual processors per real processor
  - 24% area growth per core for SMT
  - Natural extension to POWER4 design



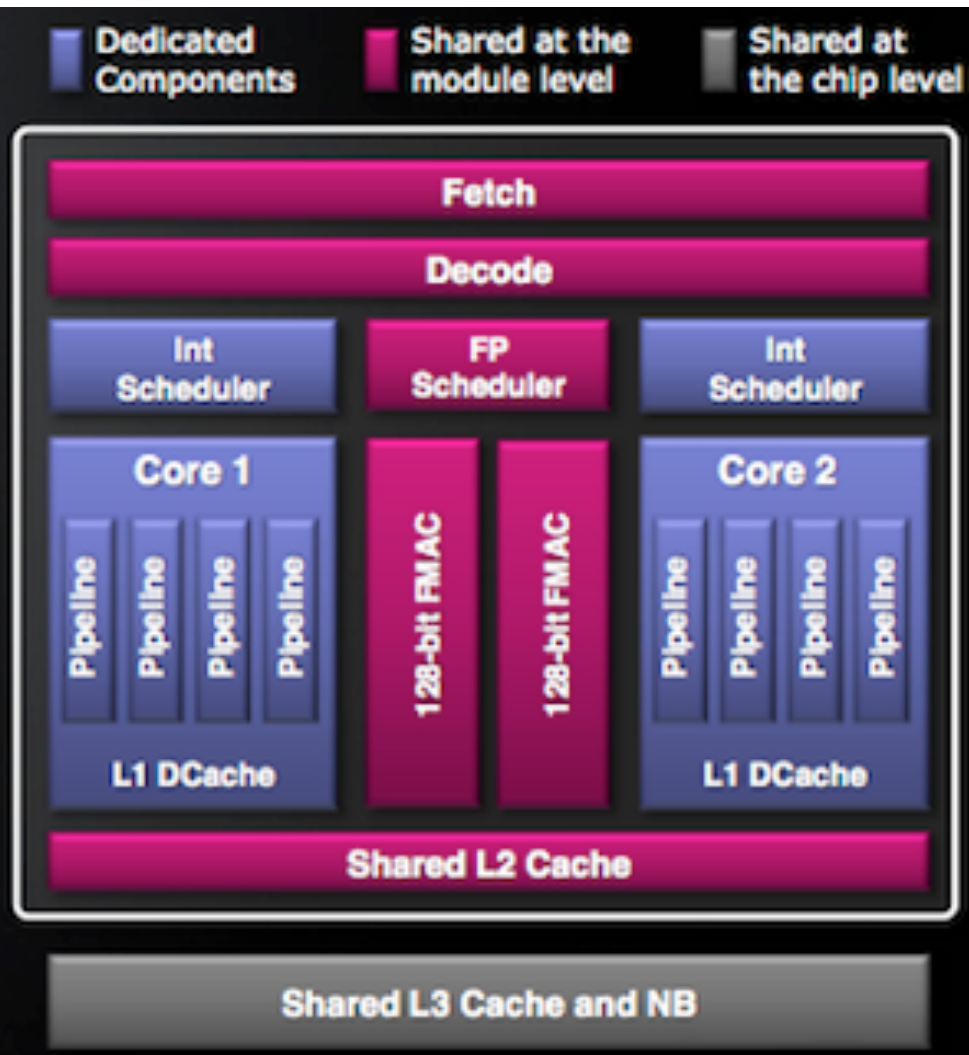
AMD Bulldozer

- 32 nm 6- or 8-core x86 CPUs, “3.5 GHz+”
- 16 kB, 4-way L1 per-core, 64 kB, 2-way L2 shared across 2-core “modules”, 8 MB L3 shared across all cores
- 213M transistors per 2-core module
  - 2 ALUs, 2 AGU [address generation unit] per core
  - Second ALU costs 12% extra per module, 5% extra per die
- “In general, AMD has made tradeoffs in favor of leaving data where it is and using the out-of-order window to hold and juggle pointers to that data ... Bulldozer's approach spreads out to take up more die space, but the upside from spreading the work around to more structures is that you get finer granularity of control in managing power by turning blocks on and off.”

AMD Bulldozer

Sharing Resources

- The Bulldozer architecture has shared and dedicated components
- The shared components:
  - Help reduce power consumption
  - Help reduce die space (cost)
- The dedicated components:
  - Help increase performance and scalability
- Bulldozer dynamically switches between shared and dedicated components to maximize performance per watt



http://arstechnica.com/business/news/2011/02/amd-to-break-new-ground-with-32nm-bulldozer-design.ars

AMD Bulldozer

- “SMT works great for multithreaded workloads where none of the threads that share the processor need to run full-bore.”
- “AMD claims that in the real world, a single two-way SMT core works as well as 1.3 regular cores, because the threads don't wait on main memory but on execution resources to free up.”

http://arstechnica.com/business/news/2010/08/evolution-not-revolution-a-look-at-amds-bulldozer.ars

CMP Benefits

- Volume: 2 processors where 1 was before
- Power: All processors on one die share a single connection to rest of system

CMP Power

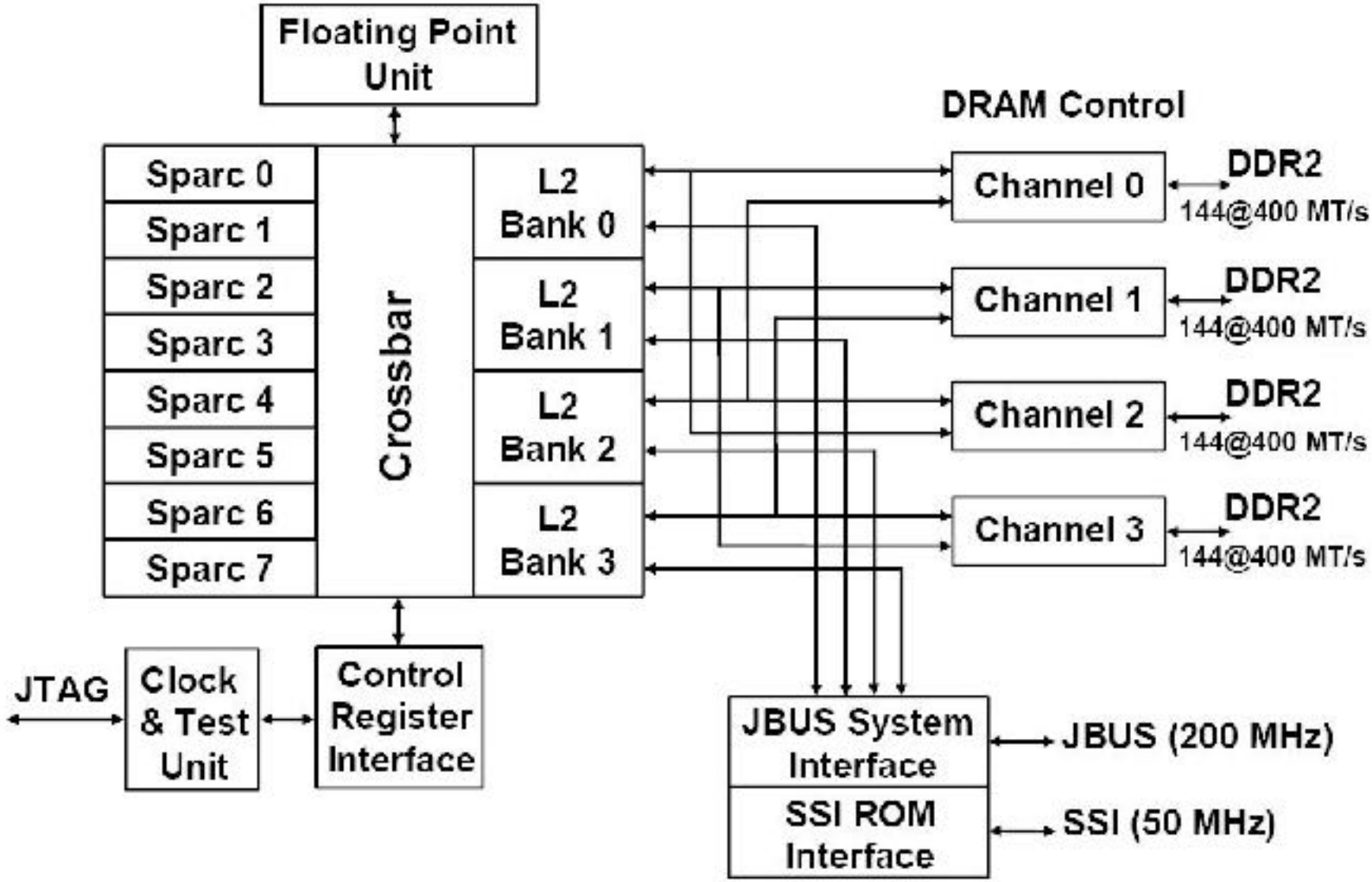
- Consider a 2-way CMP replacing a uniprocessor
- Run the CMP at half the uniprocessor's clock speed
  - Each request takes twice as long to process …
  - … but slowdown is less because request processing is likely limited by memory or disk
  - If there's not much contention, overall throughput is the same
- Half clock rate -› half voltage -› quarter power per processor, so 2x savings overall

Sun T1 (“Niagara”) (2005)

- Target: Commercial server applications
  - High thread level parallelism (TLP)
    - Large numbers of parallel client requests
  - Low instruction level parallelism (ILP)
    - High cache miss rates
    - Many unpredictable branches
    - Frequent load-load dependencies

- Power, cooling, and space are major concerns for data centers
- Metric: Performance/Watt/Sq. Ft.
- Approach: Multicore, Fine-grain multithreading, Simple pipeline, Small L1 caches, Shared L2

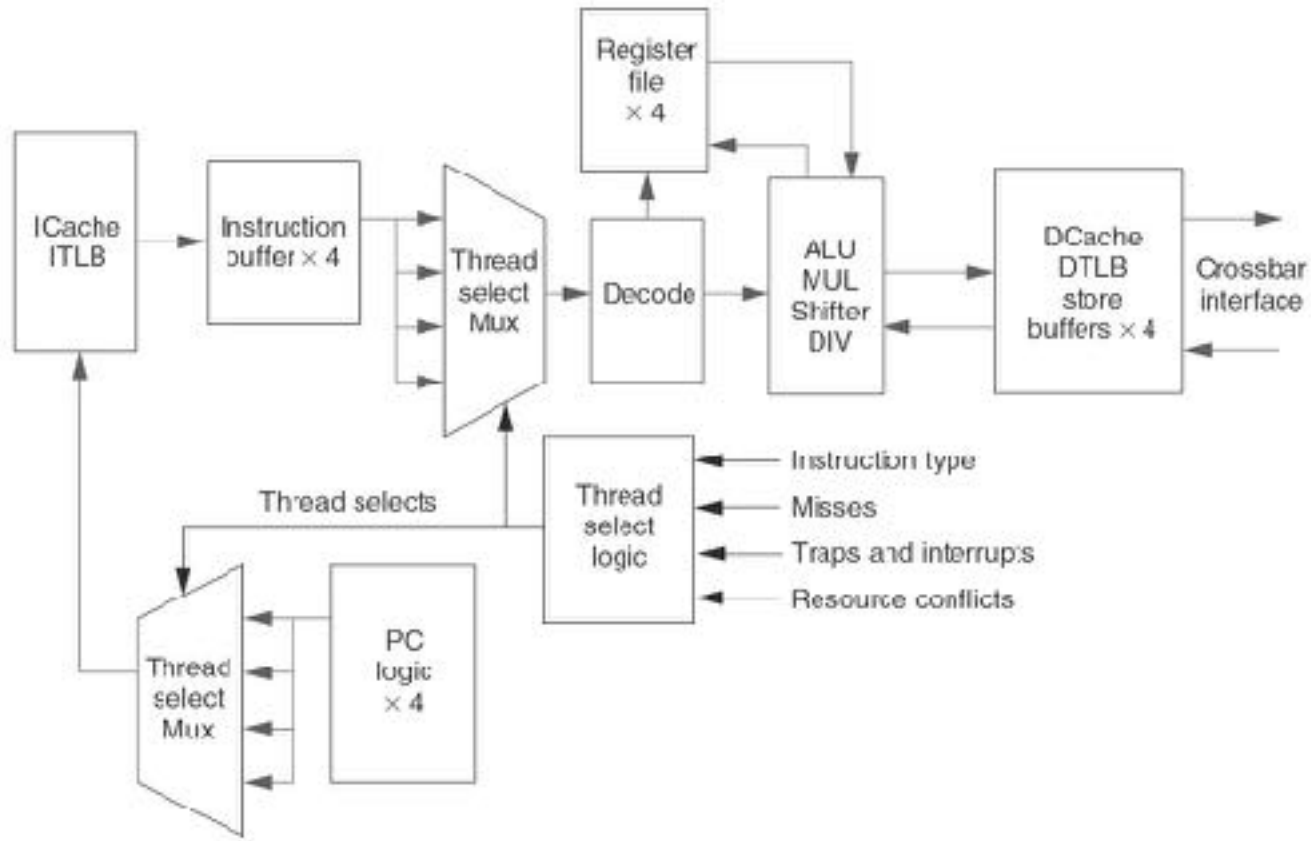
T1 Architecture



- Also ships with 6 or 4 processors

T1 pipeline

- Single issue, in-order, 6-deep pipeline: F, S, D, E, M, W
  - “Only single-issue desktop or server microprocessor introduced in more than 5 years.”
- 3 clock delays for loads & branches
- Shared units:
  - L1 $, L2 $
  - TLB
  - X units
  - pipe registers
- 1.2 GHz at ≈72W typical, 79W peak



T1 Fine-Grained Multithreading

- Each core supports four threads and has its own level one caches (16 KB for instructions and 8 KB for data)
- Switching to a new thread on each clock cycle
- Idle threads are bypassed in the scheduling
  - Waiting due to a pipeline delay or cache miss
  - Processor is idle only when all 4 threads are idle or stalled
- Both loads and branches incur a 3 cycle delay that can only be hidden by other threads
- A single set of floating point functional units is shared by all 8 cores
  - Floating point performance was not a focus for T1

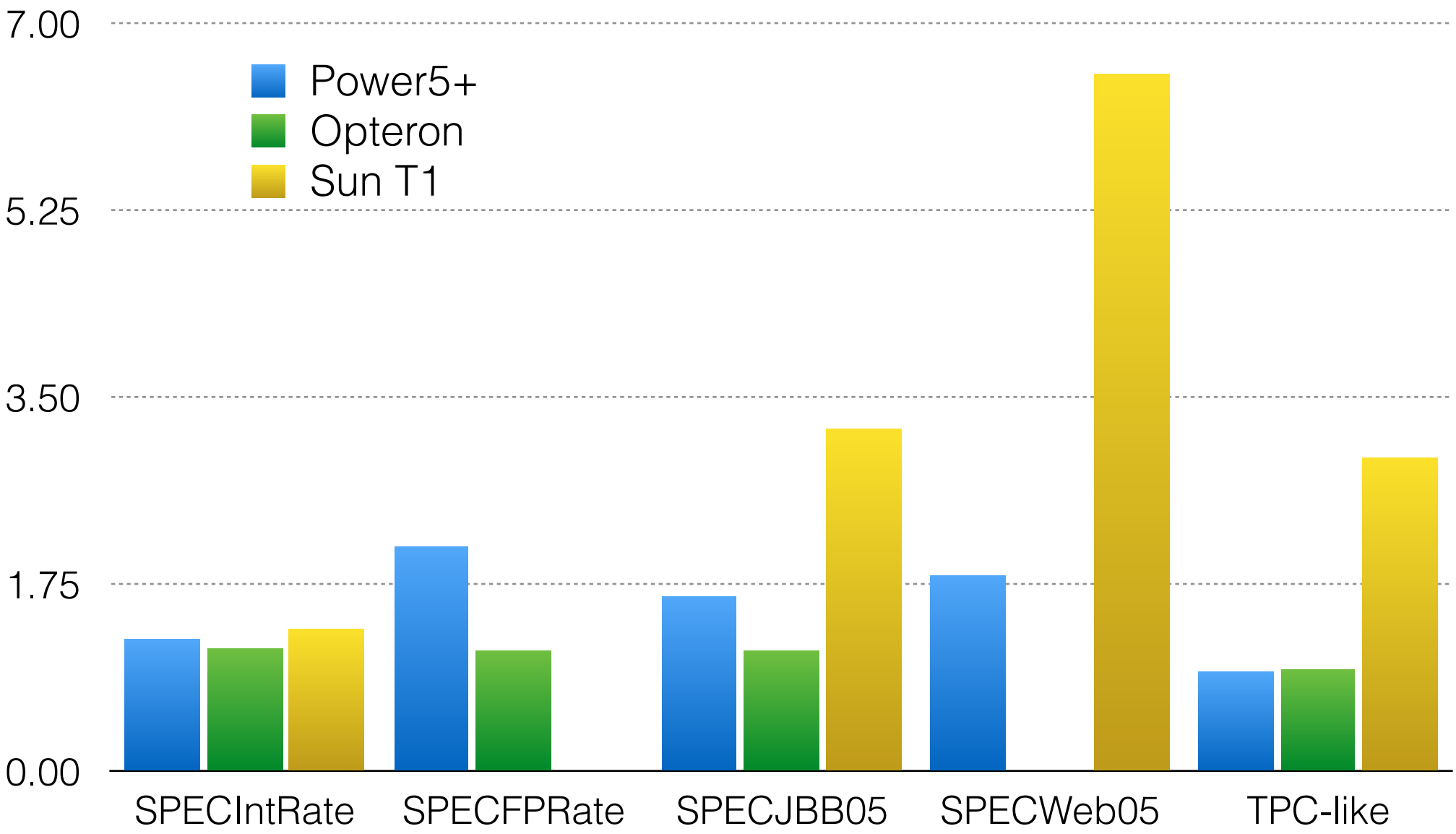
T1 Memory System

- 16 KB 4 way set assoc. I$/ core
- 8 KB 4 way set assoc. D$/ core
- 3MB 12 way set assoc. L2 $ shared
  - 4 x 750KB independent banks
  - Write-through, allocate for loads, no-allocate for stores
  - crossbar switch to connect
  - 2 cycle throughput, 8 cycle latency
  - Direct link to DRAM & Jbus
  - Manages cache coherence for the 8 cores
  - CAM based directory

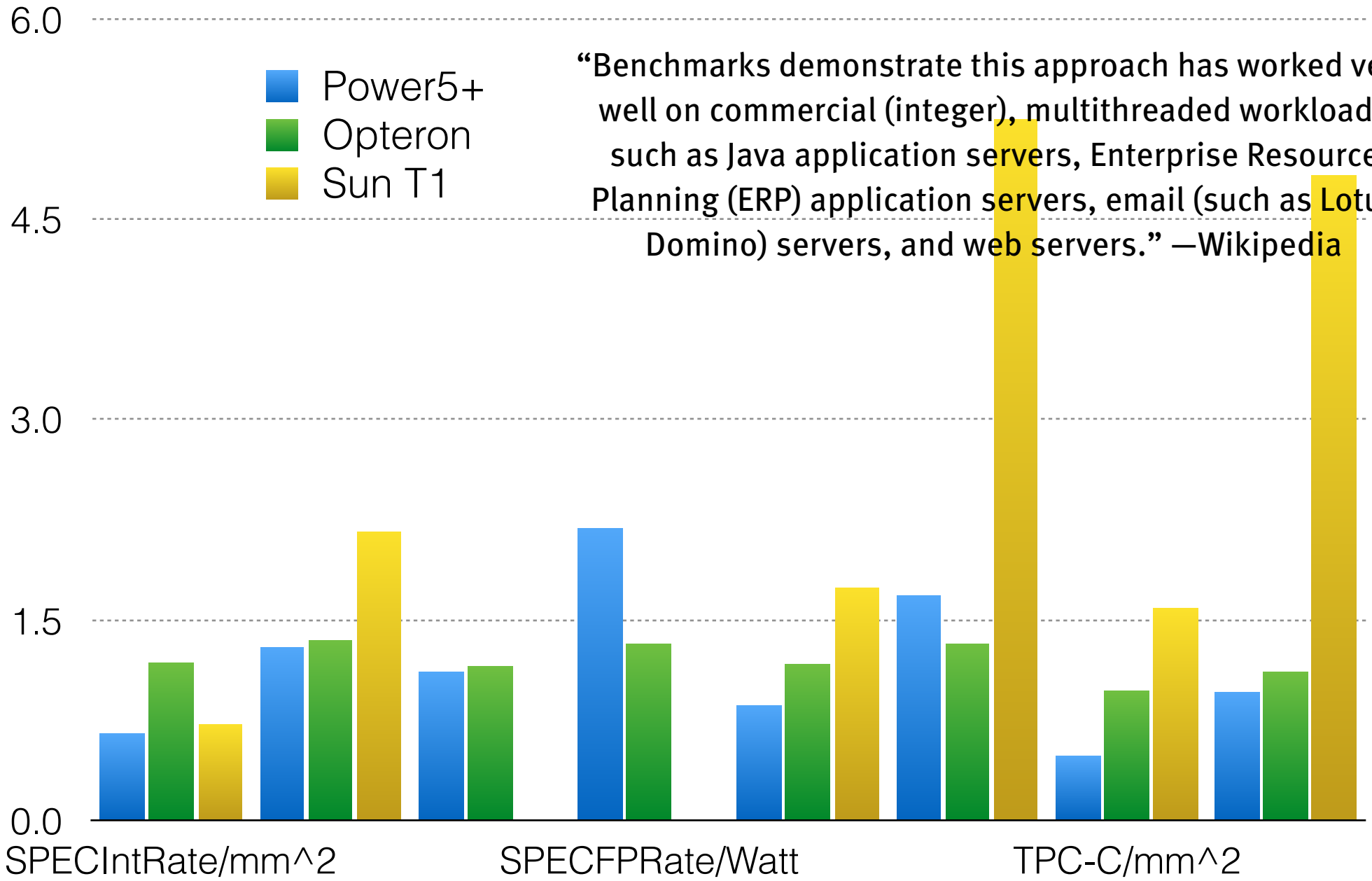
CPI Breakdown of Performance

Benchmark	Per Thread CPI	Per core CPI	Effective CPI for 8 cores	Effective IPC for 8 cores
TPC-C	7.2	1.8	0.23	4.4
SPECJBB	5.6	1.4	0.18	5.7
SPECWeb99	6.6	1.65	0.21	4.8

Performance Relative to Pentium D



Performance/mm², Performance/Watt



"Benchmarks demonstrate this approach has worked very well on commercial (integer), multithreaded workloads such as Java application servers, Enterprise Resource Planning (ERP) application servers, email (such as Lotus Domino) servers, and web servers." —Wikipedia

Microprocessor Comparison

Processor	SUN T1	Opteron	Pentium D	IBM Power 5
Cores	8	2	2	2
Instruction issues / clk / core	1	3	3	4
Peak instr. issues / chip	8	6	6	8
Multithreading	Fine-grained	No	SMT	SMT
L1 I/D in KB per core	16/8	64/64	12K uops/16	64/32
L2 per core/shared	3 MB shared	1 MB / core	1 MB/ core	1.9 MB shared
Clock rate (GHz)	1.2	2.4	3.2	1.9
Transistor count (M)	300	233	230	276
Die size (mm^2)	379	199	206	389
Power (W)	79	110	130	125

Niagara 2 (October 2007)

- Improved performance by increasing # of threads supported per chip from 32 to 64
  - 8 cores * 8 threads per core [now has 2 ALUs/core, 4 threads/ALU]
- Floating-point unit for each core, not for each chip
- Hardware support for encryption standards EAS, 3DES, and elliptical-curve cryptography
- Added 1 8x PCI Express interface directly into the chip in addition to integrated 10 Gb Ethernet XAU interfaces and Gigabit Ethernet ports.
- Integrated memory controllers will shift support from DDR2 to FB-DIMMs and double the maximum amount of system memory.

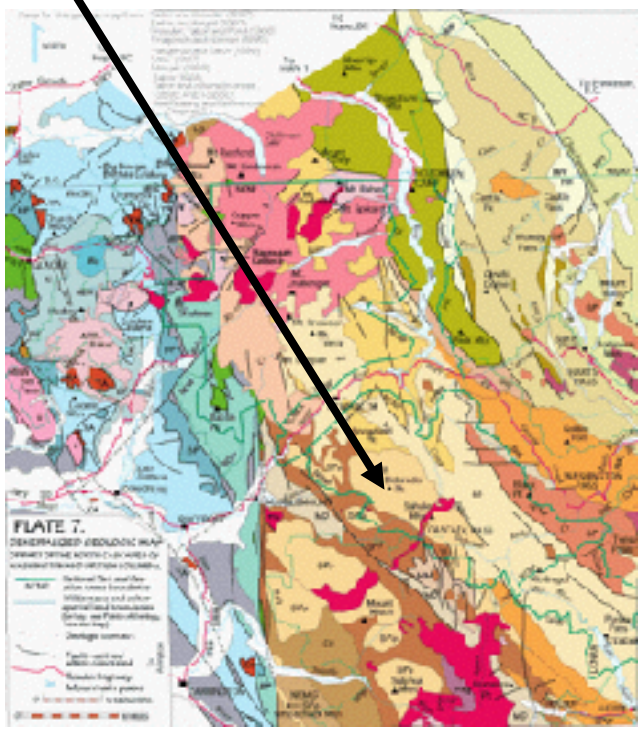
Kevin Krewell, “Sun's Niagara Begins CMT Flood—The Sun UltraSPARC T1 Processor Released”. Microprocessor Report, January 3, 2006

SPARC T5 (March 2013)

- 16 cores, 8 threads/core, 8 MB shared L3 cache
- Design scales to 8 sockets with no additional hardware
- 28 nm process, 3.6 GHz
- "The S3 core is a dual-issue core that uses dynamic threading and out-of-order execution,[6] incorporates one floating point unit, one dedicated cryptographic unit per core."

Overview

- Eldorado is a peak in the North Cascades.
- Internal Cray project name for the next MTA system.
  - Make the MTA-2 cost-effective and supportable.
  - Retain proven performance and programmability advantages of the MTA-2.
- Eldorado
  - Hardware based on Red Storm.
  - Programming model and software based on MTA-2.



Slides from John Feo, Cray Inc.

… but balance is short-lived

- Processors have gotten much faster (~ 1000x)
- Memories have gotten a little faster (~ 10x), and much larger (~ 1,000,000x)
- System diameters have gotten much larger (~ 1000x)
- Flops are for free, but bandwidth is very expensive
- ***Systems are no longer balanced, processors are starved for data***

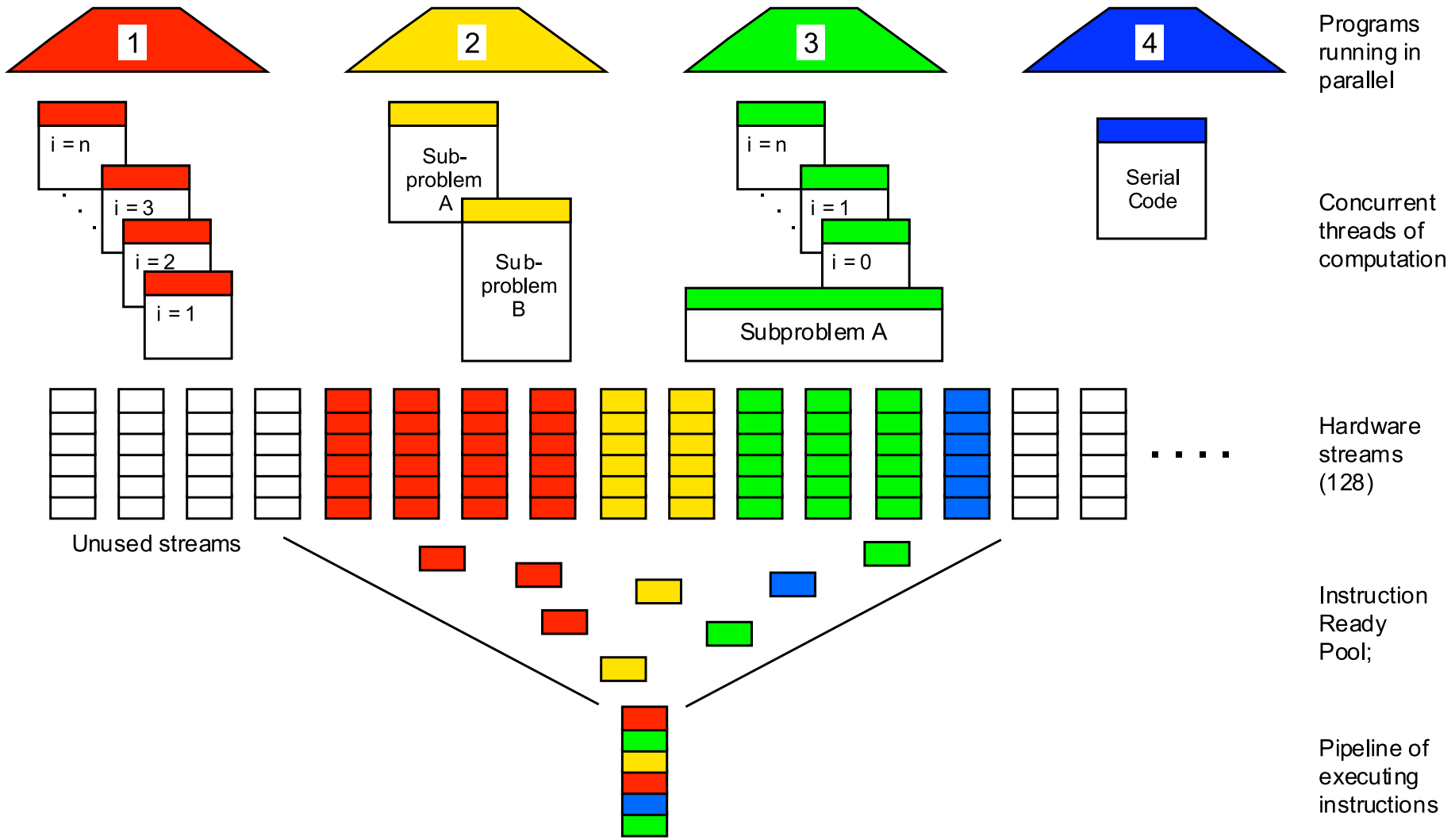
Constraints on parallel programs

- Place data near computation
- Avoid modifying shared data
- Access data in order and reuse
- Avoid indirection and linked data-structures
- Partition program into independent, balanced computations
- Avoid adaptive and dynamic computations
- Avoid synchronization and minimize inter-process communications

Multithreading

- Hide latencies via parallelism
- Maintain multiple active threads per processor, so that ***gaps introduced by long latency operations in one thread are filled by instructions in other threads***
- Requires special hardware support
  - Multiple threads
  - Single-cycle context switch
  - Multiple outstanding memory requests per thread
  - Fine-grain synchronization

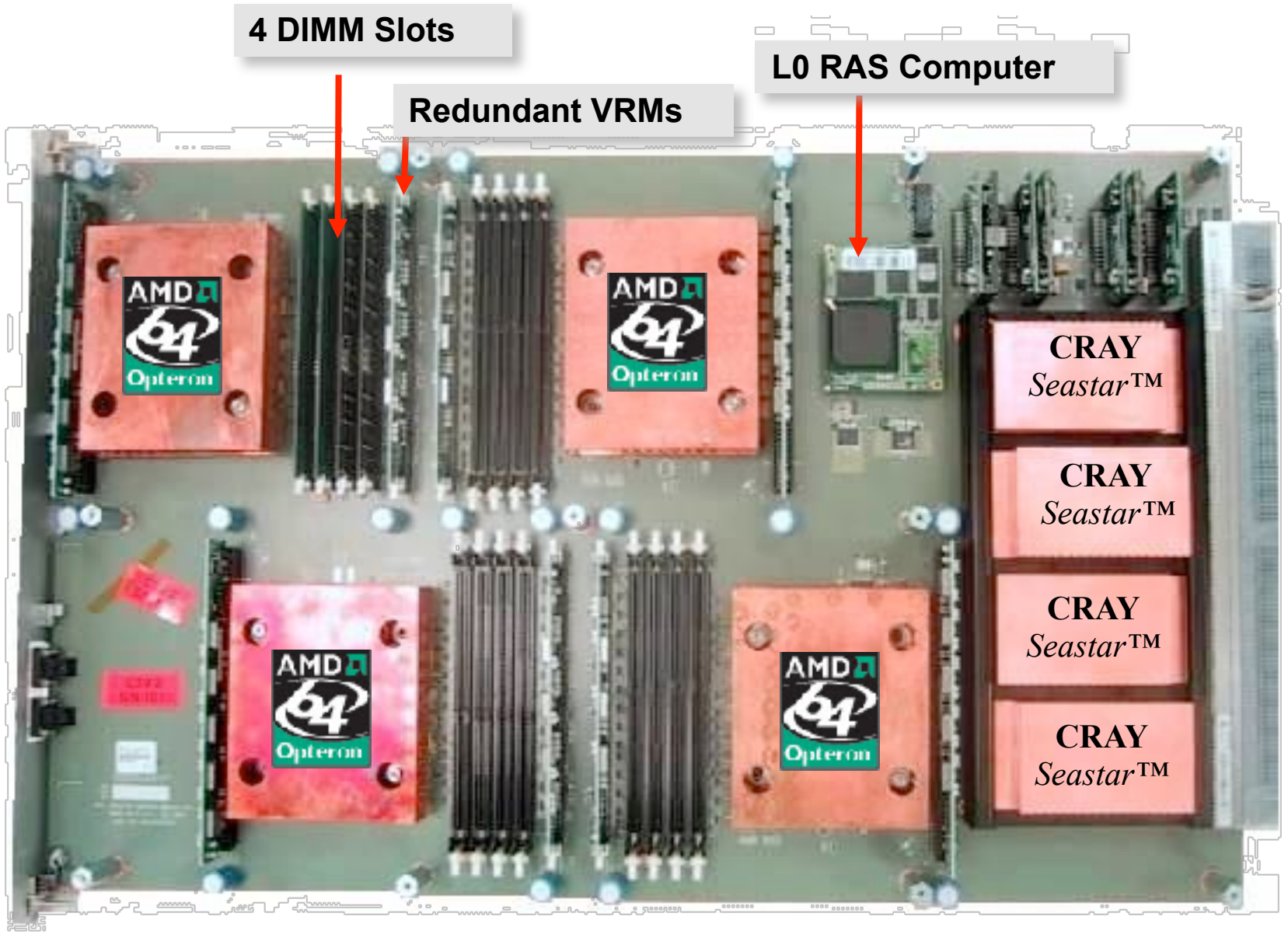
Eldorado Processor (logical view)



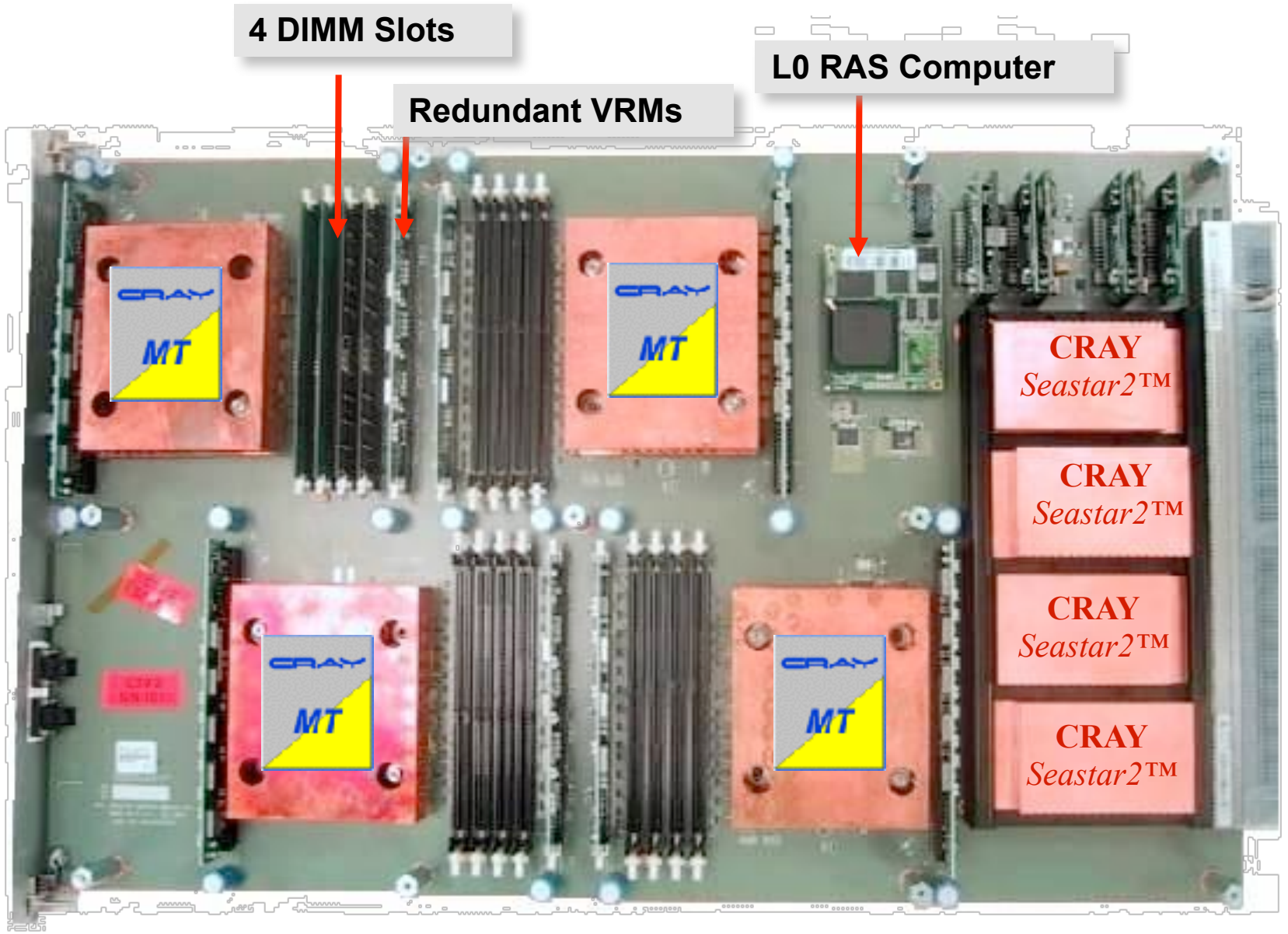
Red Storm

- Red Storm consists of over 10,000 AMD Opteron™ processors connected by an innovative high speed, high bandwidth 3D mesh interconnect designed by Cray (Seastar)
- Cray is responsible for the design, development, and delivery of the Red Storm system to support the Department of Energy's Nuclear stockpile stewardship program for advanced 3D modeling and simulation
- Red Storm uses a distributed memory programming model (MPI)

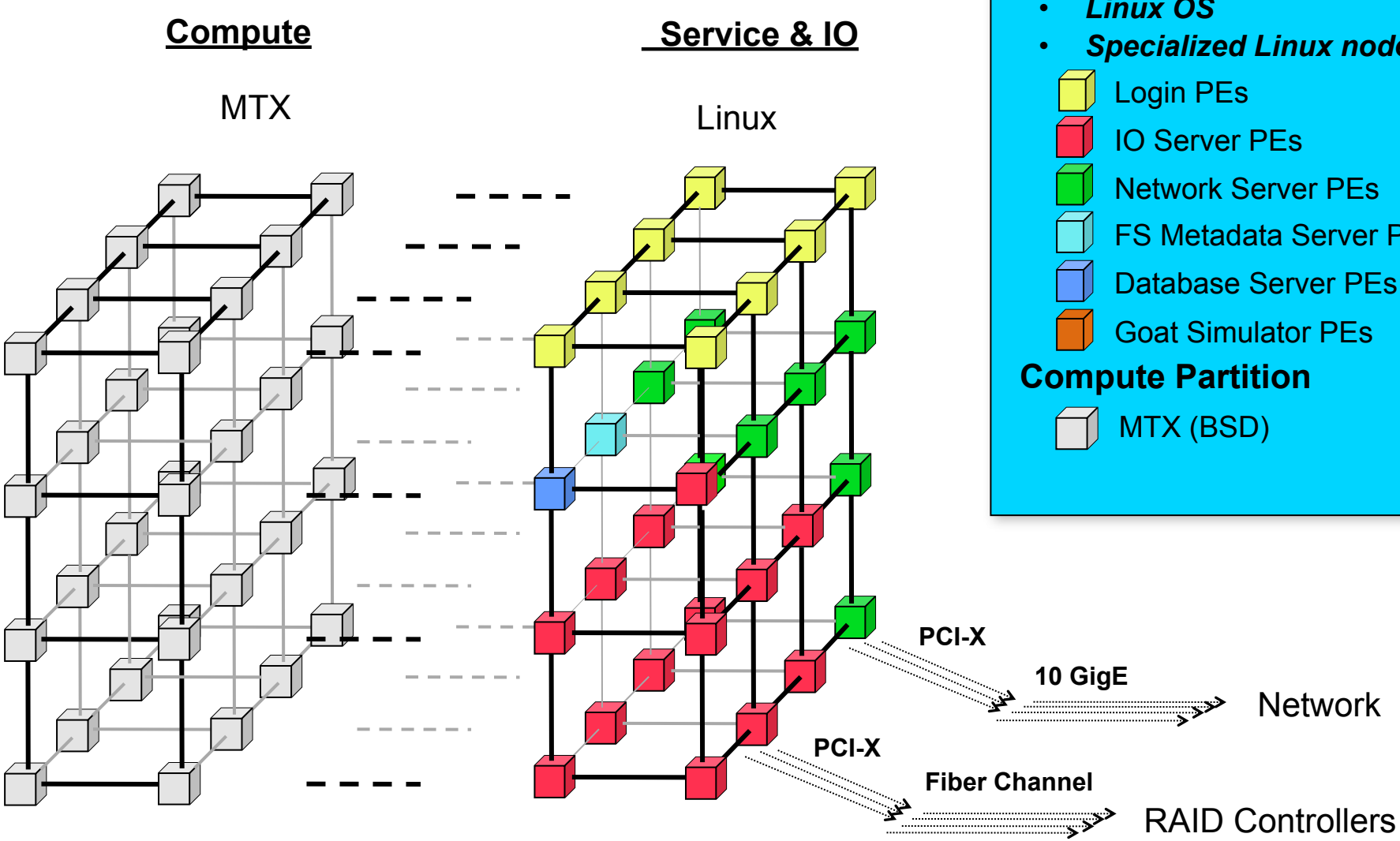
Red Storm Compute Board



Eldorado Compute Board



Eldorado system architecture



Service Partition

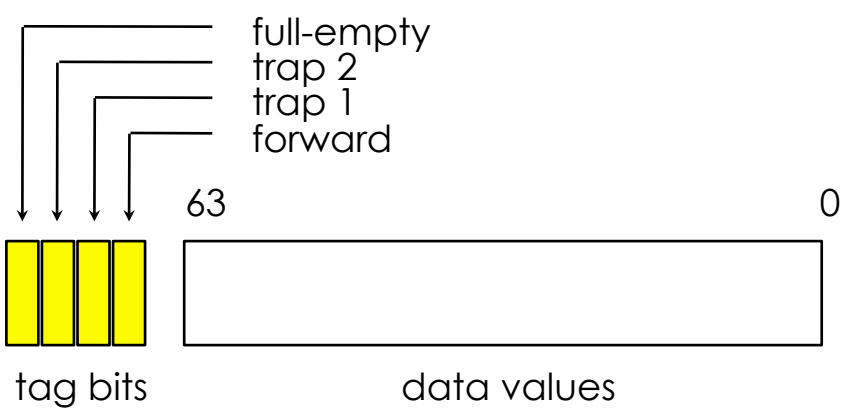
- ***Linux OS***
- ***Specialized Linux nodes***
  - Login PEs
  - IO Server PEs
  - Network Server PEs
  - FS Metadata Server PEs
  - Database Server PEs
  - Goat Simulator PEs

Compute Partition

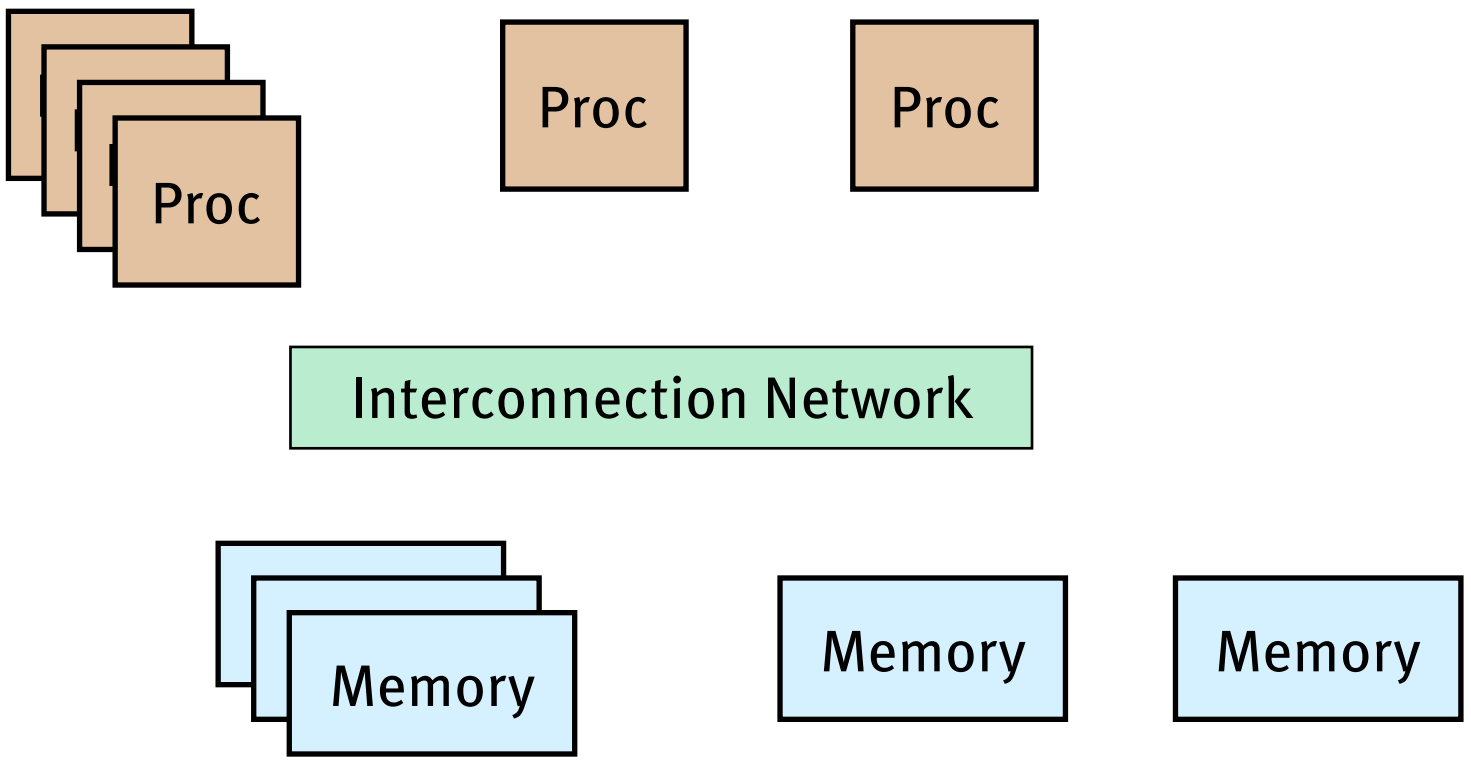
- MTX (BSD)

Eldorado memory

- Shared memory
  - Some memory can be reserved as local memory at boot time
  - Only compiler and runtime system have access to local memory
- Memory module cache
  - Decreases latency and increases bandwidth
  - No coherency issues
- 8 word data segments ***randomly distributed*** across the memory system
  - Eliminates stride sensitivity and hotspots
  - Makes programming for data locality impossible
  - Segment moves to cache, but only word moves to processor
- Full/empty bits on all data words



A generic parallel architecture

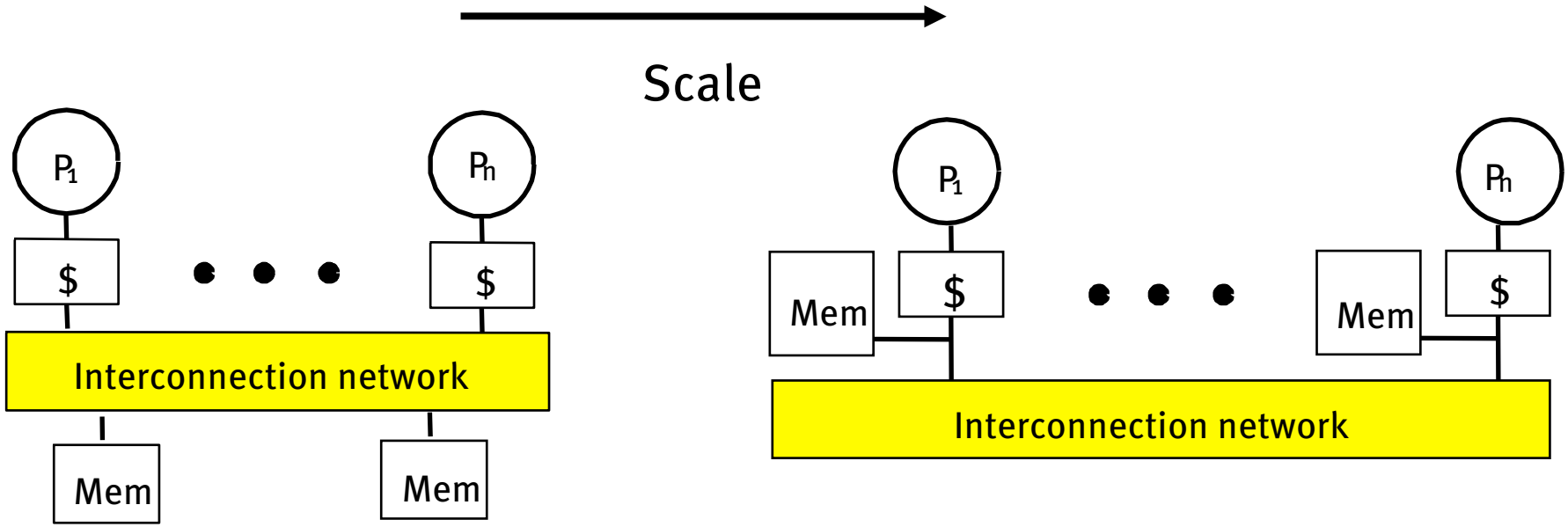


Where is the memory physically located?

Is it connected directly to processors?

What is the connectivity of the network?

Centralized vs. Distributed Memory



Centralized Memory

Distributed Memory

What is a programming model?

Specification model (in domain of the application)
Programming model
Computational model (representation of computation)
Cost model (how computation maps to hardware)

- Is a programming model a language?
  - Programming models allow you to express ideas in particular ways
  - Languages allow you to put those ideas into practice

Writing Parallel Programs

- ***Identify*** **concurrency in task**
  - Do this in your head
- ***Expose*** **the concurrency when writing the task**
  - Choose a programming model and language that allow you to express this concurrency
- ***Exploit*** **the concurrency**
  - Choose a language and hardware that together allow you to take advantage of the concurrency

Parallel Programming Models

- Programming model is made up of the languages and libraries that create an abstract view of the machine
- Control
  - How is parallelism created?
  - What orderings exist between operations?
  - How do different threads of control synchronize?

Parallel Programming Models

- Programming model is made up of the languages and libraries that create an abstract view of the machine
- Data
  - What data is private vs. shared?
  - How is logically shared data accessed or communicated?

Parallel Programming Models

- Programming model is made up of the languages and libraries that create an abstract view of the machine
- Synchronization
  - What operations can be used to coordinate parallelism?
  - What are the atomic (indivisible) operations?
    - Next slides

Segue: Atomicity

- Swaps between threads can happen any time
- Communication from other threads can happen any time
- Other threads can access shared memory any time
- Think about how to grab a shared resource (lock):
  - Wait until lock is free
  - When lock is free, grab it
  - ```
    while (*ptrLock == 1) ;     // 1 means "locked"
    *ptrLock = 1;
    ```
    ```

Segue: Atomicity

- Think about how to grab a shared resource (lock):
  - Wait until lock is free
  - When lock is free, grab it
  - ```
    while (*ptrLock == 1) ;     // 1 means "locked"
    *ptrLock = 1;
    ```
- Why do you want to be able to do this?
- What could go wrong with the code above?
- How do we fix it?
    ```

Parallel Programming Models

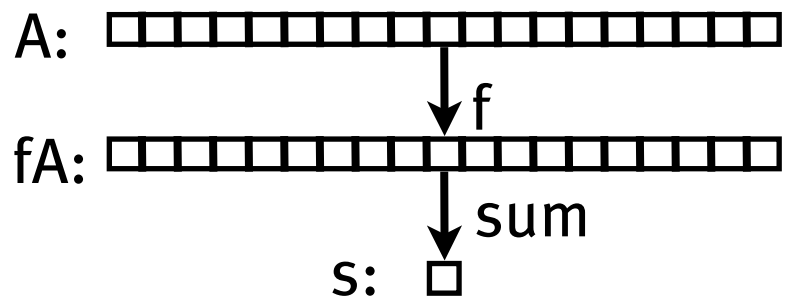
- Programming model is made up of the languages and libraries that create an abstract view of the machine
- Cost
  - How do we account for the cost of each of the above?

Simple Example

- Consider applying a function f to the elements of an array A and then computing its sum:

$$\sum_{i=0}^{n-1} f(A[i])$$

A = array of all data
fA = f(A)
s = sum(fA)



- Questions:
  - Where does A live? All in single memory? Partitioned?
  - How do we divide the work among processors?
  - How do processors cooperate to produce a single result?