EECS 252 Graduate Computer Architecture

Lec 15 – T1 ("Niagara") and Papers Discussion

David Patterson
Electrical Engineering and Computer Sciences
University of California, Berkeley

http://www.eecs.berkeley.edu/~pattrsn
http://vlsi.cs.berkeley.edu/cs252-s06

Review

- Caches contain all information on state of cached memory blocks
- Snooping cache over shared medium for smaller MP by invalidating other cached copies on write
- Sharing cached data ⇒ Coherence (values returned by a read), Consistency (when a written value will be returned by a read)
- Snooping and Directory Protocols similar; bus makes snooping easier because of broadcast (snooping => uniform memory access)
- Directory has extra data structure to keep track of state of all cache blocks
- Distributing directory => scalable shared address multiprocessor
  => Cache coherent, Non uniform memory access

Outline

- Consistency
- Cross Cutting Issues
- Fallacies and Pitfalls
- Administrivia
- Sun T1 ("Niagara") Multiprocessor
- 2 paper discussion

Another MP Issue: Memory Consistency Models

- What is consistency? When must a processor see the new value? e.g., seems that

```
P1:  A = 0;                    P2:  B = 0;
     .....                          .....
     A = 1;                         B = 1;
L1:  if (B == 0) ...           L2:  if (A == 0) ...
```

- Impossible for both if statements L1 & L2 to be true?
  - What if write invalidate is delayed & processor continues?
- Memory consistency models: what are the rules for such cases?
- Sequential consistency: result of any execution is the same as if the accesses of each processor were kept in order and the accesses among different processors were interleaved ⇒ assignments before ifs above
  - SC: delay all memory accesses until all invalidates done

Memory Consistency Model

- Schemes faster execution to sequential consistency
- Not an issue for most programs; they are synchronized
  - A program is synchronized if all access to shared data are ordered by synchronization operations

```
write (x)
...
release (s) {unlock}
...
acquire (s) {lock}
...
read(x)
```

- Only those programs willing to be nondeterministic are not synchronized: "data race": outcome f(proc. speed)
- Several Relaxed Models for Memory Consistency since most programs are synchronized; characterized by their attitude towards: RAR, WAR, RAW, WAW to different addresses

Relaxed Consistency Models: The Basics

- Key idea: allow reads and writes to complete out of order, but to use synchronization operations to enforce ordering, so that a synchronized program behaves as if the processor were sequentially consistent
  - By relaxing orderings, may obtain performance advantages
  - Also specifies range of legal compiler optimizations on shared data
  - Unless synchronization points are clearly defined and programs are synchronized, compiler could not interchange read and write of 2 shared data items because might affect the semantics of the program
- 3 major sets of relaxed orderings:

1. W→R ordering (all writes completed before next read)
   - Because retains ordering among writes, many programs that operate under sequential consistency operate under this model, without additional synchronization. Called processor consistency
2. W → W ordering (all writes completed before next write)
3. R → W and R → R orderings, a variety of models depending on ordering restrictions and how synchronization operations enforce ordering

- Many complexities in relaxed consistency models; defining precisely what it means for a write to complete; deciding when processors can see values that it has written

Mark Hill observation

- Instead, use speculation to hide latency from strict consistency model
  - If processor receives invalidation for memory reference before it is committed, processor uses speculation recovery to back out computation and restart with invalidated memory reference

1. Aggressive implementation of sequential consistency or processor consistency gains most of advantage of more relaxed models
2. Implementation adds little to implementation cost of speculative processor
3. Allows the programmer to reason using the simpler programming models

Cross Cutting Issues: Performance Measurement of Parallel Processors

- Performance: how well scale as increase Proc
- Speedup fixed as well as scaleup of problem
  - Assume benchmark of size n on p processors makes sense: how scale benchmark to run on m * p processors?
  - Memory-constrained scaling: keeping the amount of memory used per processor constant
  - Time-constrained scaling: keeping total execution time, assuming perfect speedup, constant
- Example: 1 hour on 10 P, time ~ $O(n^3)$, 100 P?
  - Time-constrained scaling: 1 hour ⇒ $10^{1/3}n$ ⇒ 2.15n scale up
  - Memory-constrained scaling: 10n size ⇒ $10^3/10$ ⇒ 100X or 100 hours! 10X processors for 100X longer???
  - Need to know application well to scale: # iterations, error tolerance

Fallacy: Amdahl's Law doesn't apply to parallel computers

- Since some part linear, can't go 100X?
- 1987 claim to break it, since 1000X speedup
  - researchers scaled the benchmark to have a data set size that is 1000 times larger and compared the uniprocessor and parallel execution times of the scaled benchmark. For this particular algorithm the sequential portion of the program was constant independent of the size of the input, and the rest was fully parallel—hence, linear speedup with 1000 processors
- Usually sequential scale with data too

Fallacy: Linear speedups are needed to make multiprocessors cost-effective

- Mark Hill & David Wood 1995 study
- Compare costs SGI uniprocessor and MP
- Uniprocessor = $38,400 + $100 * MB
- MP = $81,600 + $20,000 * P + $100 * MB
- 1 GB, uni = $138k v. mp = $181k + $20k * P
- What speedup for better MP cost performance?
- 8 proc = $341k; $341k/138k ⇒ 2.5X
- 16 proc ⇒ need only 3.6X, or 25% linear speedup
- Even if need some more memory for MP, not linear

Fallacy: Scalability is almost free

- "build scalability into a multiprocessor and then simply offer the multiprocessor at any point on the scale from a small number of processors to a large number"
- Cray T3E scales to 2048 CPUs vs. 4 CPU Alpha
  - At 128 CPUs, it delivers a peak bisection BW of 38.4 GB/s, or 300 MB/s per CPU (uses Alpha microprocessor)
  - Compaq Alphaserver ES40 up to 4 CPUs and has 5.6 GB/s of interconnect BW, or 1400 MB/s per CPU
- Build apps that scale requires significantly more attention to load balance, locality, potential contention, and serial (or partly parallel) portions of program. 10X is very hard

Pitfall: Not developing SW to take advantage (or optimize for) multiprocessor architecture

- SGI OS protects the page table data structure with a single lock, assuming that page allocation is infrequent
- Suppose a program uses a large number of pages that are initialized at start-up
- Program parallelized so that multiple processes allocate the pages
- But page allocation requires lock of page table data structure, so even an OS kernel that allows multiple threads will be serialized at initialization (even if separate processes)

Answers to 1995 Questions about Parallelism

- In the 1995 edition of this text, we concluded the chapter with a discussion of two then current controversial issues.

1. What architecture would very large scale, microprocessor-based multiprocessors use?
2. What was the role for multiprocessing in the future of microprocessor architecture?

Answer 1. Large scale multiprocessors did not become a major and growing market ⇒ clusters of single microprocessors or moderate SMPs

Answer 2. Astonishingly clear. For at least for the next 5 years, future MPU performance comes from the exploitation of TLP through multicore processors vs. exploiting more ILP

Cautionary Tale

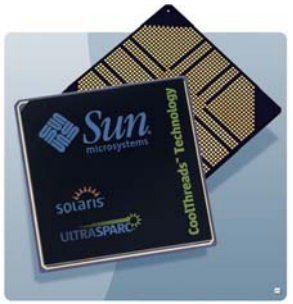
- Key to success of birth and development of ILP in 1980s and 1990s was software in the form of optimizing compilers that could exploit ILP
- Similarly, successful exploitation of TLP will depend as much on the development of suitable software systems as it will on the contributions of computer architects
- Given the slow progress on parallel software in the past 30+ years, it is likely that exploiting TLP broadly will remain challenging for years to come

CS 252 Administrivia

- Wednesday March 15 MP Future Directions and Review
- Monday March 20 Quiz 5-8 PM 405 Soda
- Monday March 20 lecture – Q&A, problem sets with Archana
- Wednesday March 22 no class: project meetings in 635 Soda
- Spring Break March 27 – March 31
- Chapter 6 Storage
- Interconnect Appendix

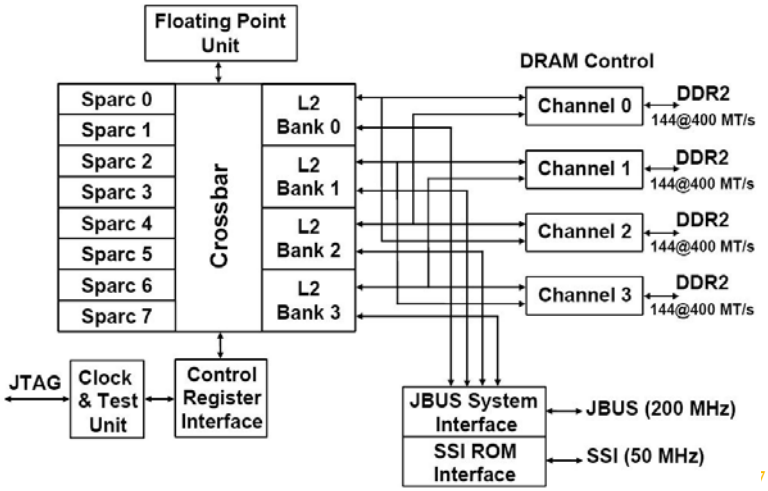
T1 ("Niagara")

- 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



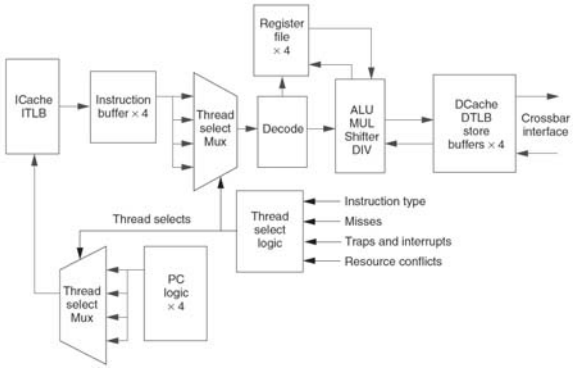
T1 Architecture

- Also ships with 6 or 4 processors



T1 pipeline

- Single issue, in-order, 6-deep pipeline: F, S, D, E, M, W
- 3 clock delays for loads & branches.
- Shared units:
  - L1 $, L2 $
  - TLB
  - X units
  - pipe registers



- Hazards:
  - Data
  - Structural

T1 Fine-Grained Multithreading

- Each core supports four threads and has its own level one caches (16KB 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

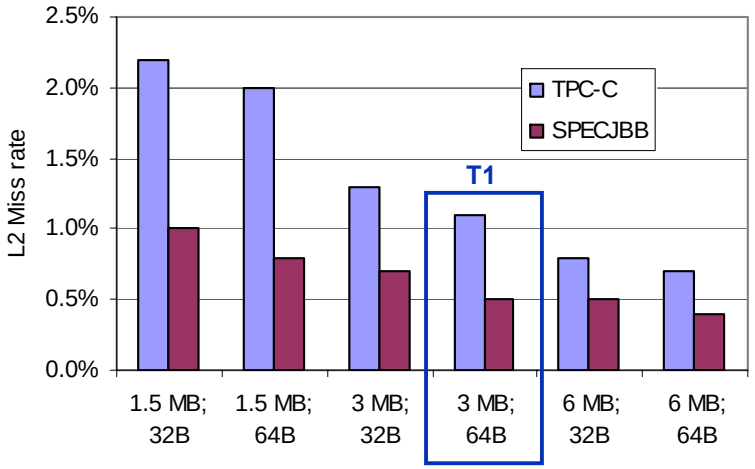
Memory, Clock, Power

- 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
  - 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

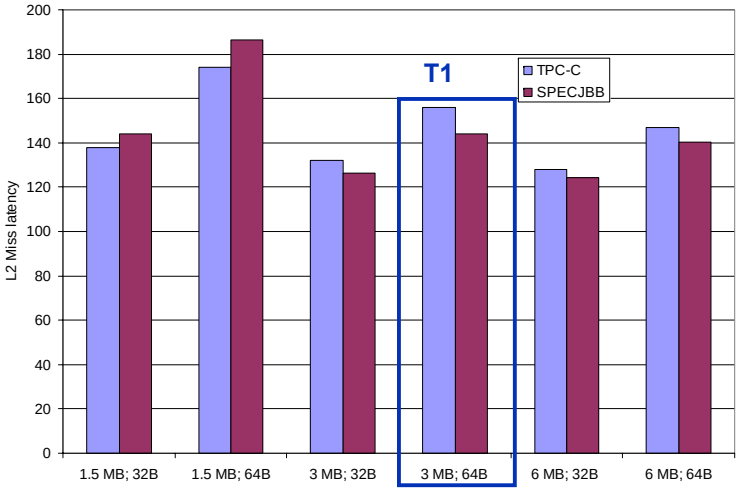
Write through
- allocate LD
- no-allocate ST

- Coherency is enforced among the L1 caches by a directory associated with each L2 cache block
- Used to track which L1 caches have copies of an L2 block
- By associating each L2 with a particular memory bank and enforcing the subset property, T1 can place the directory at L2 rather than at the memory, which reduces the directory overhead
- L1 data cache is write-through, only invalidation messages are required; the data can always be retrieved from the L2 cache
- 1.2 GHz at ≈72W typical, 79W peak power consumption

Miss Rates: L2 Cache Size, Block Size



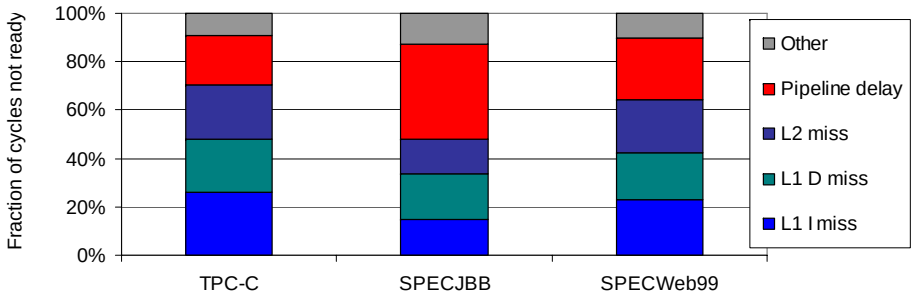
Miss Latency: L2 Cache Size, Block Size



CPI Breakdown of Performance

Benchmark	Per Thread CPI	Per core CPI	Effective CPI for 8 cores	Effective IPC for 8 cores
TPC-C	7.20	1.80	0.23	4.4
SPECJBB	5.60	1.40	0.18	5.7
SPECWeb99	6.60	1.65	0.21	4.8

Not Ready Breakdown



- **TPC-C - store buffer full is largest contributor**
- **SPEC-JBB - atomic instructions are largest contributor**
- **SPECWeb99 - both factors contribute**

Performance: Benchmarks + Sun Marketing

Benchmark\Architecture	Sun Fire T2000	IBM p5-550 with 2 dual-core Power5 chips	Dell PowerEdge
SPECjbb2005 (Java server software) business operations/ sec	63,378	61,789	24,208 (SC1425 with dual single-core Xeon)
SPECweb2005 (Web server performance)	14,001	7,881	4,850 (2850 with two dual-core Xeon processors)
NotesBench (Lotus Notes performance)	16,061	14,740	

SPECjappServer 2004 Dual Node		
	Sun Fire T2000	HP rx4640
Space (RU)	2	4
Watts	320	1,303
Performance (SPECjapp JOPs)	615	471
Performance / Watt	1.922	0.361
SWaP	0.96	0.09

Space, Watts, and Performance

HP marketing view of T1 Niagara

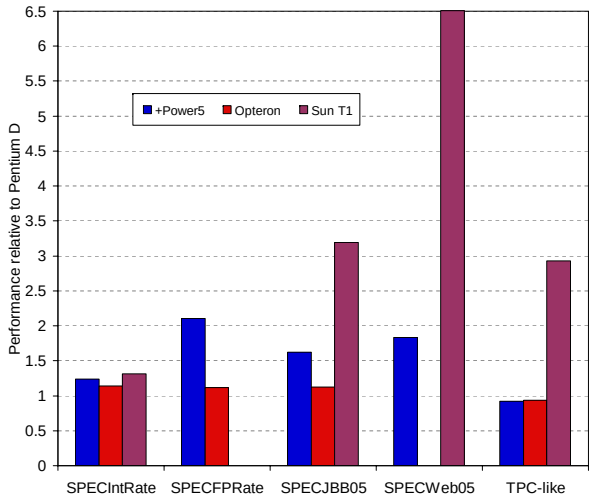
1. Sun's radical UltraSPARC T1 chip is made up of individual cores that have much slower single thread performance when compared to the higher performing cores of the Intel Xeon, Itanium, AMD Opteron or even classic UltraSPARC processors.
2. The Sun Fire T2000 has poor floating-point performance, by Sun's own admission.
3. The Sun Fire T2000 does not support commerical Linux or Windows® and requires a lock-in to Sun and Solaris.
4. The UltraSPARC T1, aka CoolThreads, is new and unproven, having just been introduced in December 2005.
5. In January 2006, a well-known financial analyst downgraded Sun on concerns over the UltraSPARC T1's limitation to only the Solaris operating system, unique requirements, and longer adoption cycle, among other things. [10]

- Where is the compelling value to warrant taking such a risk?
- http://h71028.www7.hp.com/ERC/cache/280124-0-0-0-121.html

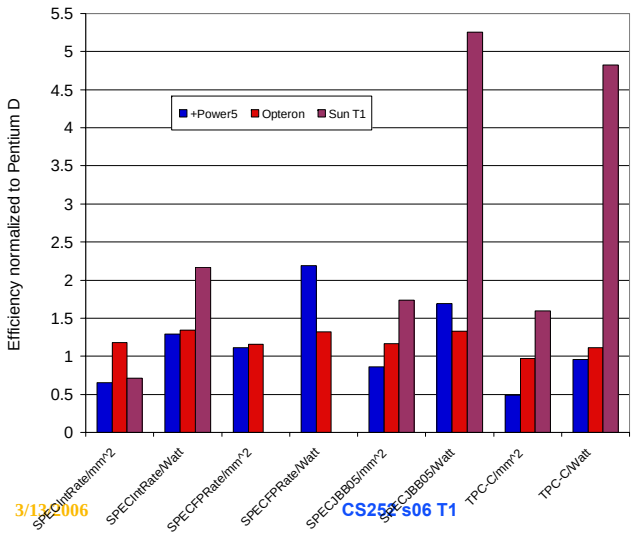
Microprocessor Comparison

Processor	SUN T1	Opteron	Pentium D	IBM Power 5
Cores	**8**	2	2	2
Instruction issues / clock / 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	1MB / core	1MB/ 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

Performance Relative to Pentium D



Performance/mm^2, Performance/Watt



Niagara 2

- Improve performance by increasing threads supported per chip from 32 to 64
  - 8 cores * 8 threads per core
- Floating-point unit for each core, not for each chip
- Hardware support for encryption standards EAS, 3DES, and elliptical-curve cryptography
- Niagara 2 will add a number of 8x PCI Express interfaces directly into the chip in addition to integrated 10Gigabit 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

Amdahl's Law Paper

- Gene Amdahl, "Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities", AFIPS Conference Proceedings, (30), pp. 483-485, 1967.
- How long is paper?
- How much of it is Amdahl's Law?
- What other comments about parallelism besides Amdahl's Law?

Parallel Programmer Productivity

- Lorin Hochstein *et al* "Parallel Programmer Productivity: A Case Study of Novice Parallel Programmers." International Conference for High Performance Computing, Networking and Storage (SC'05). Nov. 2005
- What did they study?
- What is argument that novice parallel programmers are a good target for High Performance Computing?
- How can account for variability in talent between programmers?
- What programmers studied?
- What programming styles investigated?
- How big multiprocessor?
- How measure quality?
- How measure cost?

Parallel Programmer Productivity

- Lorin Hochstein *et al* "Parallel Programmer Productivity: A Case Study of Novice Parallel Programmers." International Conference for High Performance Computing, Networking and Storage (SC'05). Nov. 2005
- What hypotheses investigated?
- What were results?
- Assuming these results of programming productivity reflect the real world, what should architectures of the future do (or not do)?
- How would you redesign the experiment they did?
- What other metrics would be important to capture?
- Role of Human Subject Experiments in Future of Computer Systems Evaluation?