Walking Four Machines by the Shore

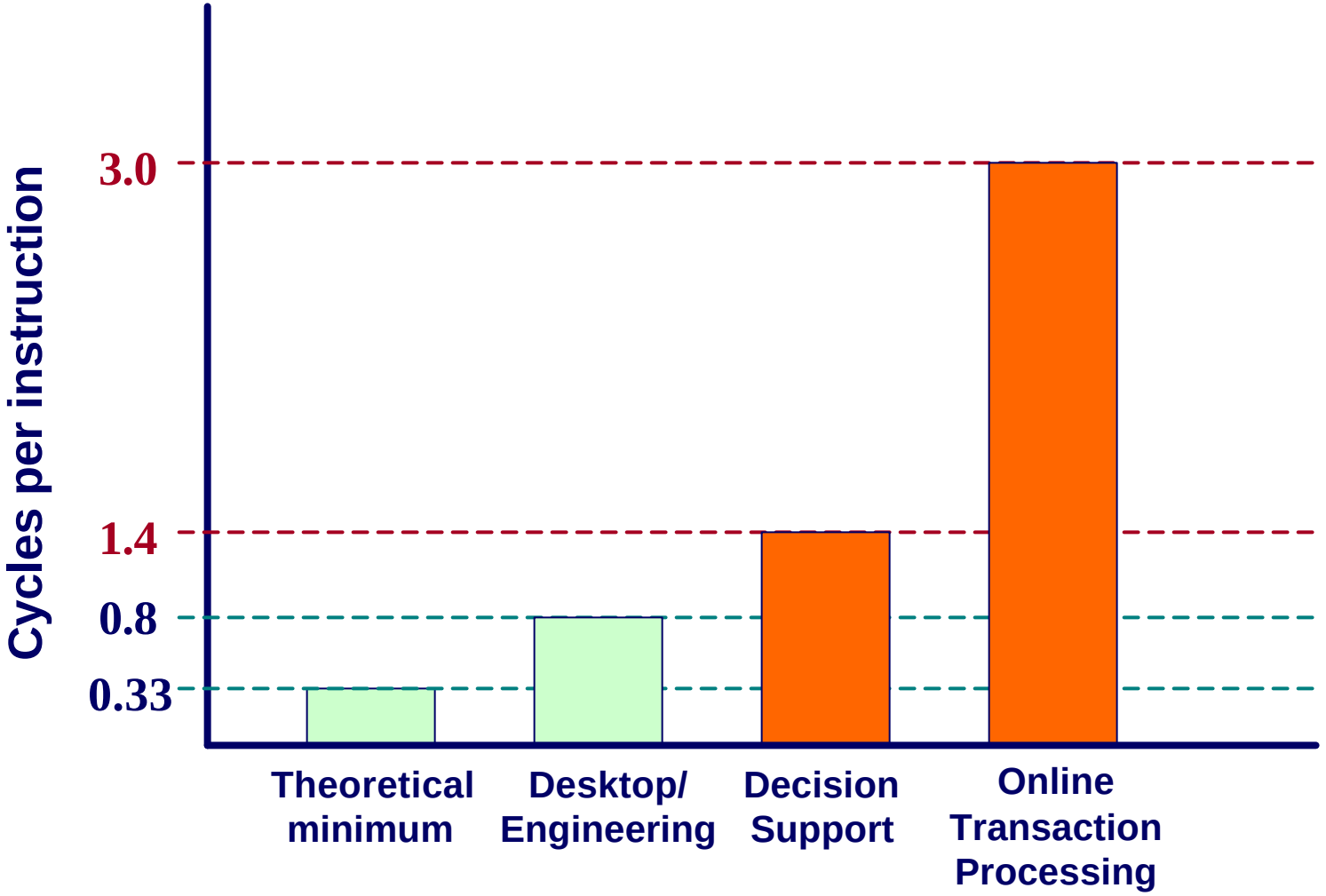
Anastassia Ailamaki

www.cs.cmu.edu/~natassa

with Mark Hill and David DeWitt

University of Wisconsin - Madison

Workloads on Modern Platforms



High CPI for DB workloads

Carnegie Mellon

Previous Work

- DBMSs on modern platforms
  - [Barroso 98], [Keeton 98], [Ailamaki 99], etc
  - Studied one or more DBMSs per platform
  - Located performance bottlenecks
- Cache-conscious software
  - Data placement
  - Optimized use of cache in algorithms
- Instruction stream optimizations
  - Optimized I-cache / branch prediction

Hardware design also affects DBMS behavior

Impact of Architectural Decisions

Shore: A prototype storage manager / DBMS

Compared Shore on four different systems

- different processor architectures/μ-architectures
- different memory subsystems

Found evidence that DBMSs would benefit from

- 2-4 way associative, larger L2, no inclusion
- large blocks, no sub-blocking
- high-accuracy branch prediction
- memory-aggressive execution engine

Steps towards a DSS-centric machine

Carnegie Mellon

Outline

- Introduction
- Experimental setup / methodology
- Processor pipeline
- Branch prediction mechanism
- Memory subsystem
- Conclusions

Carnegie Mellon

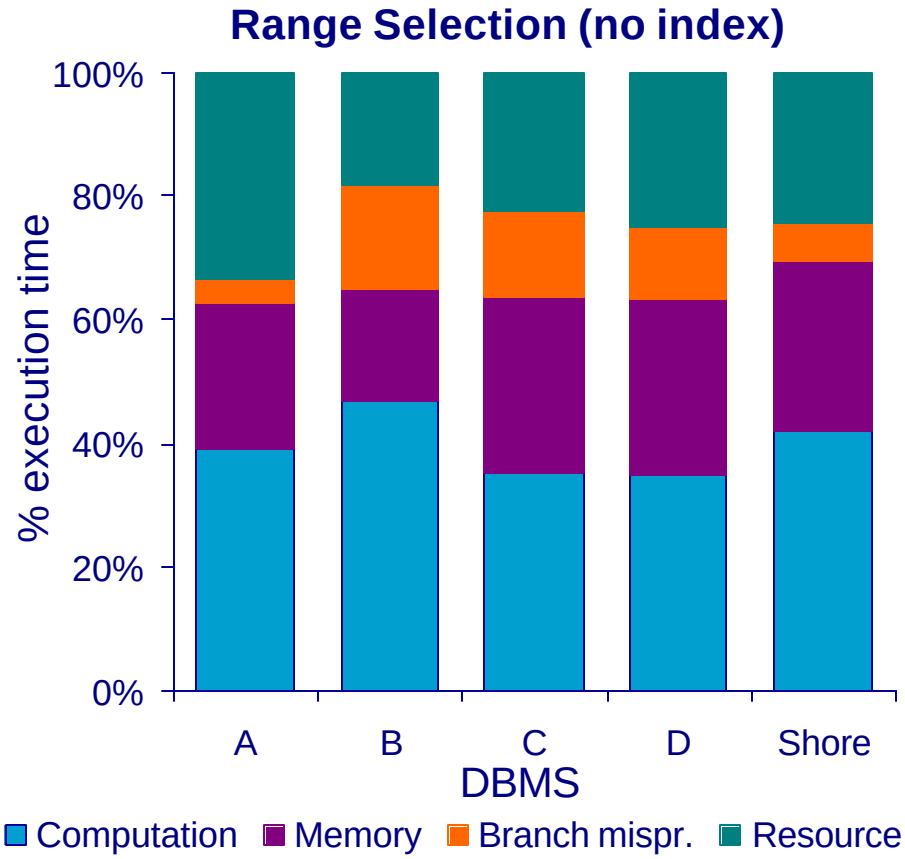
Platform Design Variations

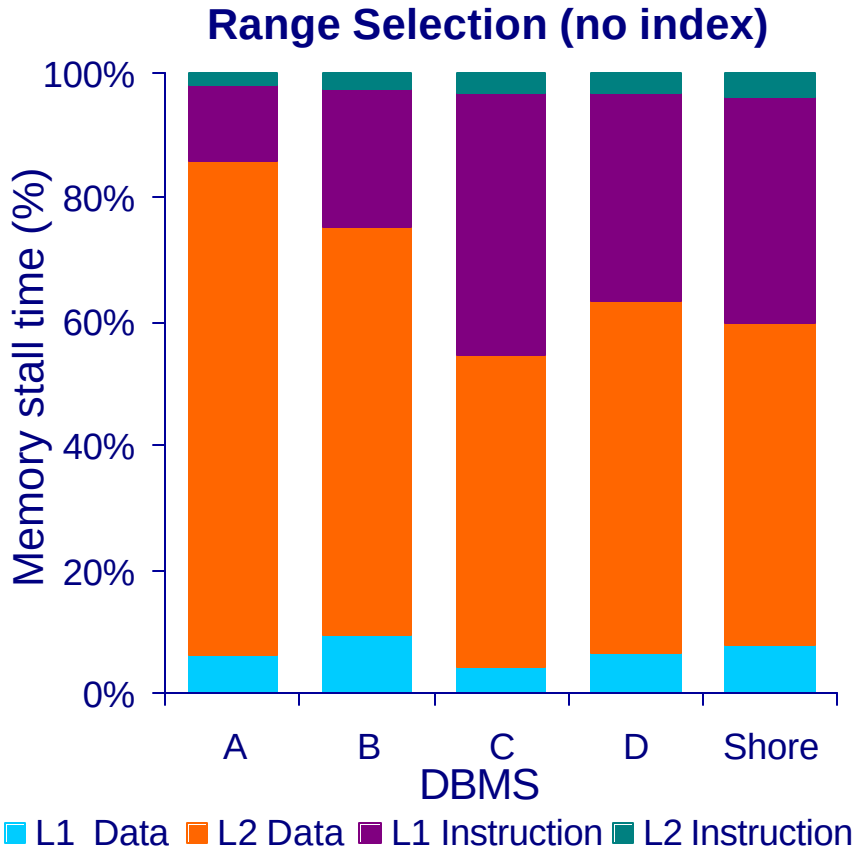
- Architecture
  - RISC or CISC Instruction set
- Microarchitecture
  - Pipeline
    - Speculation (out-of-order, multiple issue)
    - Branch prediction
  - Memory subsystem
    - Cache size, associativity
    - Block size, subblocking
    - Inclusion

Which design favors DSS workloads?

Why Use Shore?

- Range selection query on 4 commercial DBMSs + Shore
- Breakdown of execution & memory delays





We can use Shore to evaluate DSS workload behavior

Carnegie Mellon

Experimental Setup

- Used four machines
  - Sun UltraSparc: US-II and US-II*i*, Solaris 2.6/2.7
  - Intel P6: PII Xeon, Linux v2.2
  - DEC Alpha: 21164A, OSF1 v.4.0
- Architecture and Processor Microarchitecture

Characteristic	UltraSparc		PII Xeon	Alpha 21164
	US-II	US-IIi		
speed	296 MHz	300 MHz	400 MHz	532 MHz
introduced in	1997	1997	1998	1996
out of order?	no	no	yes	no
instruction set	RISC	RISC	CISC	RISC

Cache Hierarchies

Characteristic		UltraSparc		PII Xeon	Alpha 21164
		US-II	US-IIi		
L1 D	*size, assoc*	16KB, DM	16KB, DM	16KB, 2-way	8KB, DM
	block/subblock	32/16	32/16	32/32	32/32
	inclusion by L2	yes	yes	no	yes
L1 I	*size, assoc*	16KB, 2-way	16KB, 2-way	16KB, 4-way	8KB, DM
	block/subblock	32/32	32/32	32/32	32/16
	inclusion by L2	yes	yes	no	no
L2	*size, assoc*	2 MB, DM	512KB, DM	512KB, 4-way	96KB, 3-way
	block/subblock	64/64	64/64	32/32	64/32
	inclusion by L3	N/A	N/A	N/A	yes
L3	*size, assoc*	N/A	N/A	N/A	4 MB / DM
	block/subblock	N/A	N/A	N/A	64/64

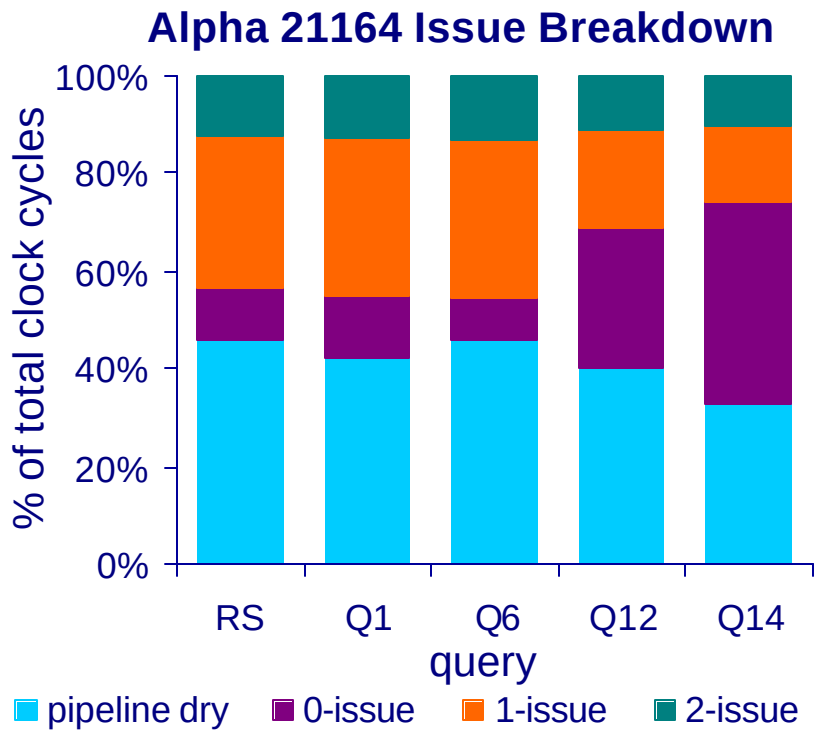
Methodology

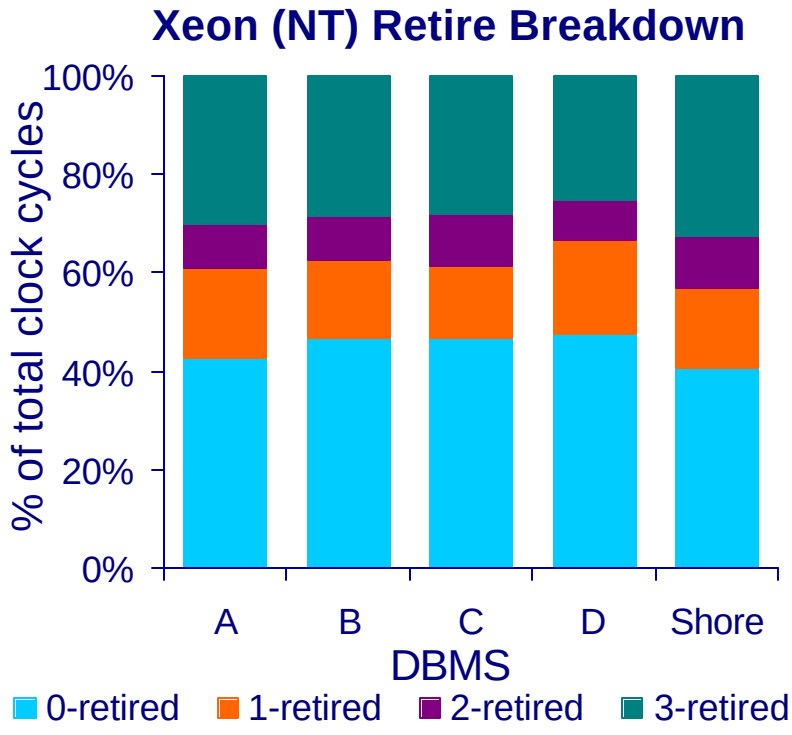
- Compiled Shore with gcc 2.95.2
  - Alpha version not optimized
- Ran DSS workload, 100-MB TPC-H dataset
  - Range Selections w/ variable parameters (RS)
  - TPC-H Q1 and Q6
    - sequential scans, lots of aggregates (*sum*, *avg*, *count*)
  - TPC-H Q12 and Q14
    - Hash Joins, complex 'where' clause, conditional aggregates
- Used processors' counters
  - Sun: run-pic (by Glenn Ammons, modified)
  - PII: PAPI (public-domain counter library)
  - Alpha: DCPI (sampling software by Compaq)

Carnegie Mellon

Issue/Retire Width

- Alpha issues at most 2 instructions / cycle (max=4)
- >60% of time Xeon retires 0/1 instruction (max=3)

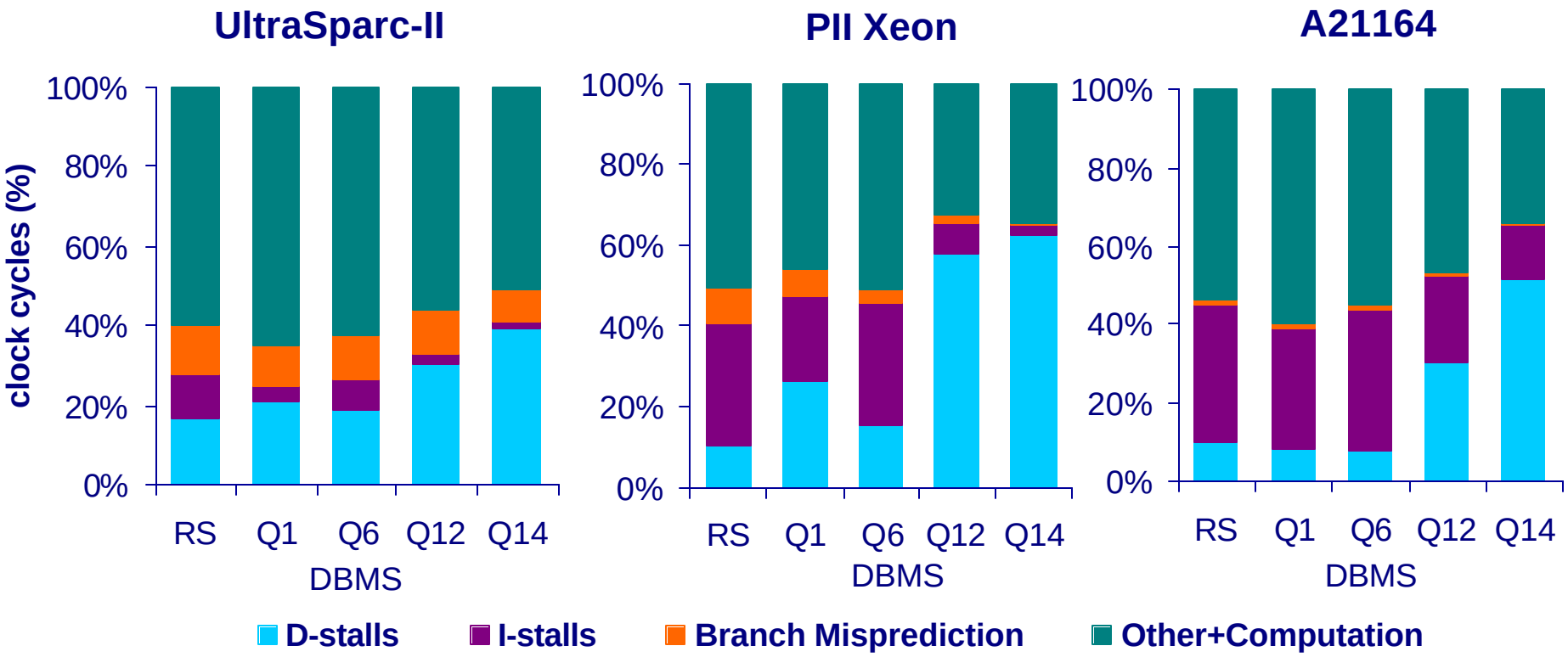




Issue/retire width is not fully exploited

Carnegie Mellon

Execution Time Breakdown



- Memory + branch misprediction stalls = 35-60% of time
- Data accesses: major memory bottleneck (esp. Q12, Q14)

Branch Prediction

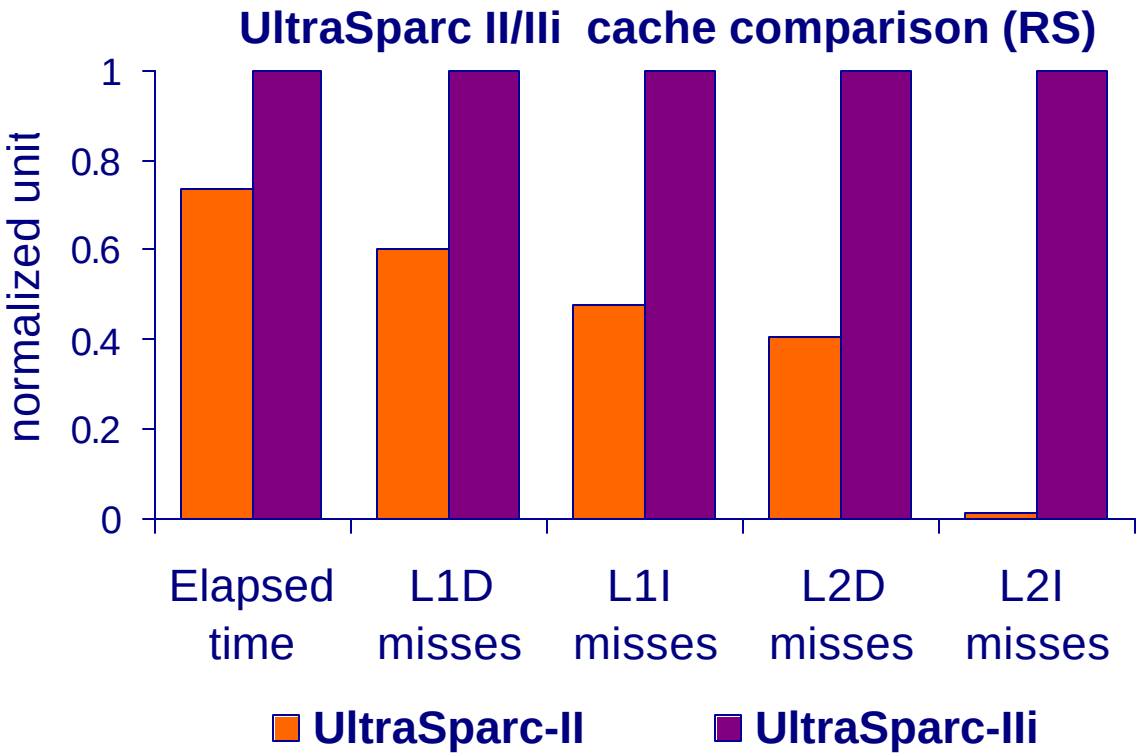
- Branch penalty = frequency*misprediction rate*penalty
- Frequency is typically 20-25%
- In-order processors => lower penalty
- Low misprediction accuracy may break it (e.g., UltraSparc)

Characteristic		PII Xeon	Alpha 21164
Branch frequency	*RS, Q1, Q6*	18%	7%
	Q12, Q14	22%	9%
Branch misprediction rate	*RS, Q1, Q6*	3.5%	15%
	Q12, Q14	1%	6%
Branch penalty (cycles)		15	5
Branch misprediction stalls		**2-11%**	**1%**

High-accuracy branch predictors

Cache Inclusion

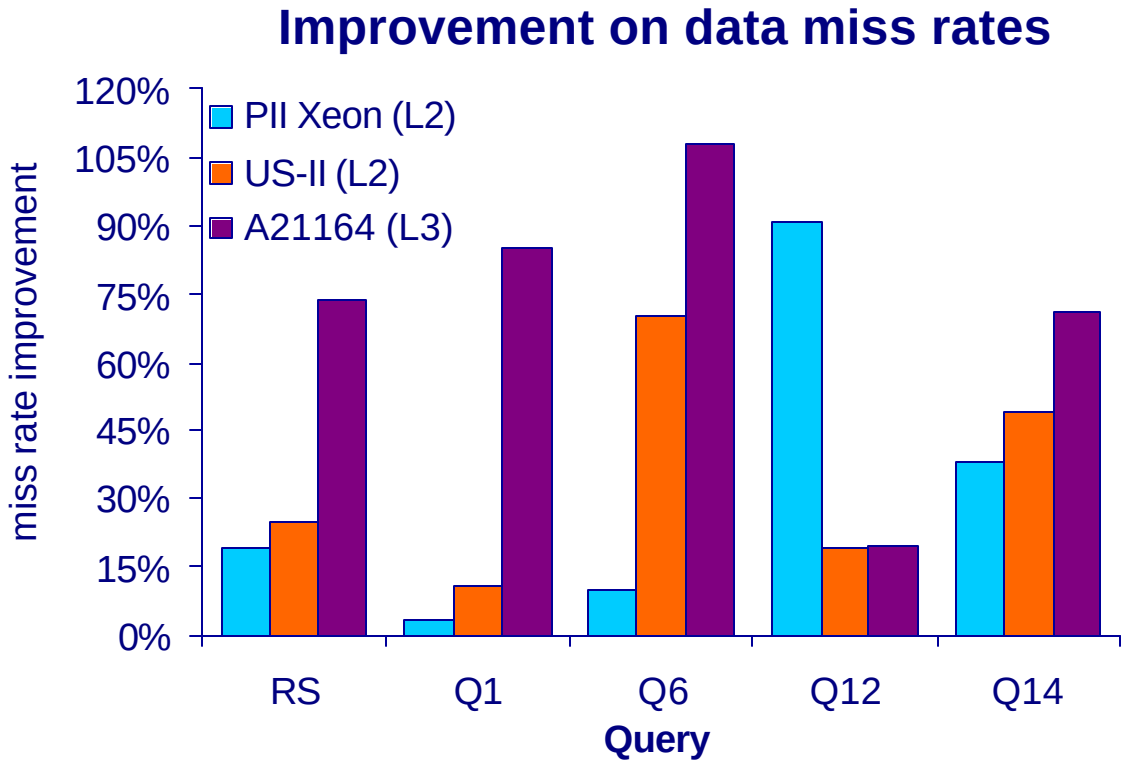
- UltraSparc II: 128-bit L1 interface, 2MB L2 cache
- UltraSparc IIi: 64-bit L1 interface, 512KB L2 cache



Small, DM L2 caches should not maintain inclusion

Cache Block Size

- Compared two data placement algorithms
- Improving locality pays off with larger cache blocks



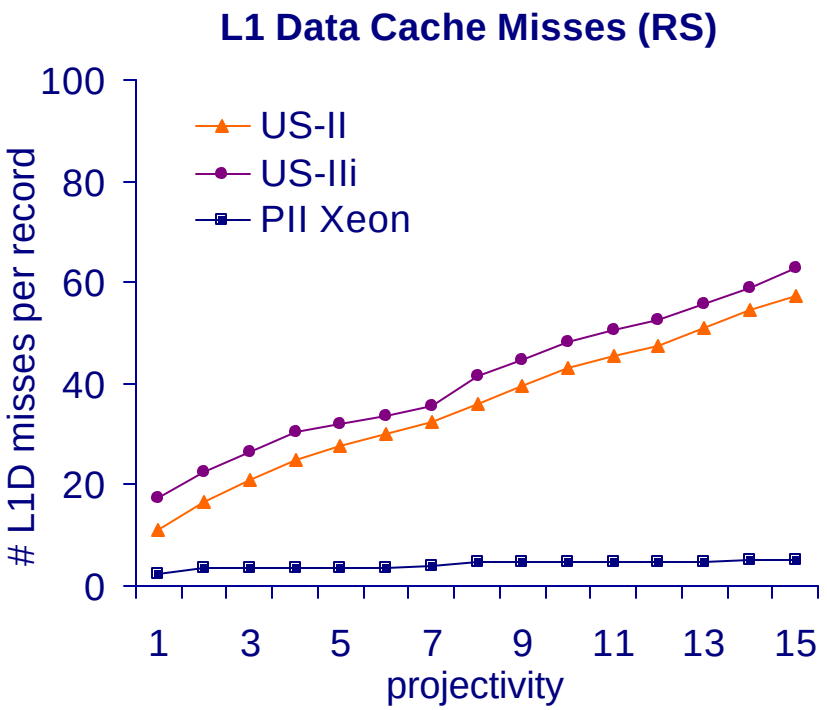
Larger cache line = lower miss rates

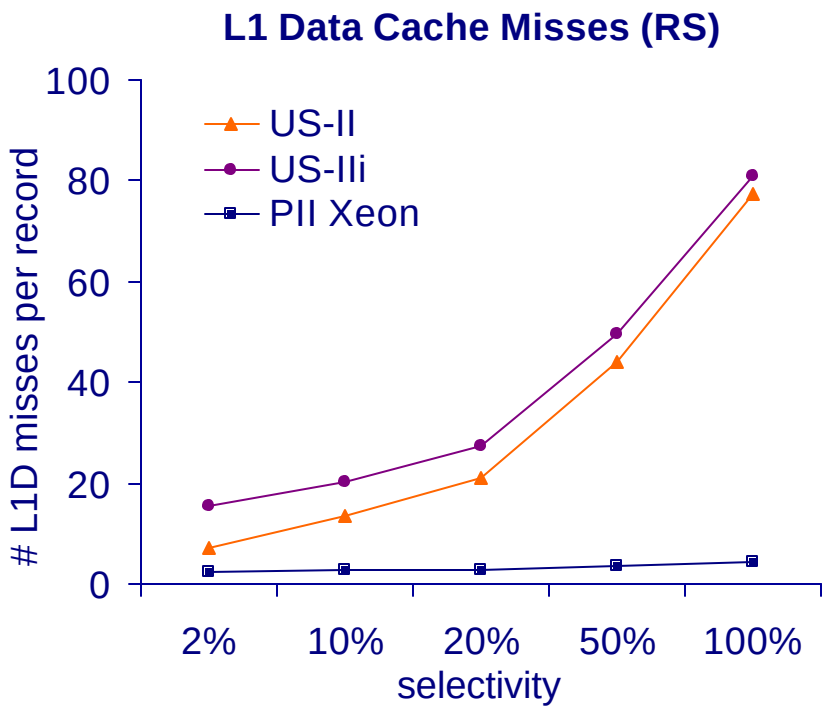
(leads to higher performance given bandwidth)

Carnegie Mellon

Sub-Blocking / Associativity

- UltraSparc: direct-mapped, subblocking (32/16)
- Xeon: 2-way, no subblocking (32/32)
- Range selections





High associativity, no sub-blocking

Carnegie Mellon

Conclusions

- Memory Hierarchy
  - Non-blocking caches
  - >64-byte block, no sub-blocking
  - Generous-sized L1-I (128K) and L2 (> 2MB)
    - A tiny, fast L1/2 with a large, slow L3 won't add much
  - High associativity (2-4)
  - No inclusion (at least for instructions)
- Processor pipeline
  - Issue width is fine, out-of-order overlaps stall time
  - Execution engine to sustain >1 load/store instr.
  - High-accuracy branch prediction

…provided that implementation cost is stable.