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
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
Outline

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

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

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>UltraSparc</th>
<th>PII Xeon</th>
<th>Alpha 21164</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>US-II</td>
<td>US-llii</td>
<td></td>
</tr>
<tr>
<td>speed</td>
<td>296 MHz</td>
<td>300 MHz</td>
<td>400 MHz</td>
</tr>
<tr>
<td>out of order?</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>instruction set</td>
<td>RISC</td>
<td>RISC</td>
<td>CISC</td>
</tr>
</tbody>
</table>
# Cache Hierarchies

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>UltraSparc</th>
<th>PII Xeon</th>
<th>Alpha 21164</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>US-II</td>
<td>US-IIi</td>
<td></td>
</tr>
<tr>
<td>L1 D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size, assoc</td>
<td>16KB, DM</td>
<td>16KB, DM</td>
<td>16KB, 2-way</td>
</tr>
<tr>
<td>block/subblock</td>
<td>32/16</td>
<td>32/16</td>
<td>32/32</td>
</tr>
<tr>
<td>inclusion by L2</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>yes</td>
</tr>
<tr>
<td>L1 I</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size, assoc</td>
<td>16KB, 2-way</td>
<td>16KB, 2-way</td>
<td>16KB, 4-way</td>
</tr>
<tr>
<td>block/subblock</td>
<td>32/32</td>
<td>32/32</td>
<td>32/32</td>
</tr>
<tr>
<td>inclusion by L2</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>no</td>
</tr>
<tr>
<td>L2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size, assoc</td>
<td>2 MB, DM</td>
<td>512KB, DM</td>
<td>512KB, 4-way</td>
</tr>
<tr>
<td>block/subblock</td>
<td>64/64</td>
<td>64/64</td>
<td>32/32</td>
</tr>
<tr>
<td>inclusion by L3</td>
<td>N/A</td>
<td>N/A</td>
<td>yes</td>
</tr>
<tr>
<td>L3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>size, assoc</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>block/subblock</td>
<td>N/A</td>
<td>N/A</td>
<td>4 MB / DM</td>
</tr>
</tbody>
</table>

© 2001 Anastassia Ailamaki   CAECW’01
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)
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
Execution Time Breakdown

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

© 2001 Anastassia Ailamaki

CAECW’01
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)

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>PII Xeon</th>
<th>Alpha 21164</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch frequency</td>
<td>RS, Q1, Q6</td>
<td>18%</td>
</tr>
<tr>
<td></td>
<td>Q12, Q14</td>
<td>22%</td>
</tr>
<tr>
<td>Branch misprediction rate</td>
<td>RS, Q1, Q6</td>
<td>3.5%</td>
</tr>
<tr>
<td></td>
<td>Q12, Q14</td>
<td>1%</td>
</tr>
<tr>
<td>Branch penalty (cycles)</td>
<td>15</td>
<td>5</td>
</tr>
<tr>
<td><em>Branch misprediction stalls</em></td>
<td>2-11%</td>
<td>1%</td>
</tr>
</tbody>
</table>

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

**Improvement on data miss rates**

Larger cache line = lower miss rates
(leads to higher performance given bandwidth)
Sub-Blocking / Associativity

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

High associativity, no sub-blocking
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.