“Multicore: This is the one which will have the biggest impact on us. We have never had a problem to solve like this. A breakthrough is needed in how applications are done on multicore devices.”

– Bill Gates

“It’s time we rethought some of the basics of computing. It’s scary and lots of fun at the same time.”

– Burton Smith
Multicore: friend... or foe?

Anastasia Ailamaki
for Nikos Hardavellas, Ryan Johnson, and Ippokratis Pandis

Carnegie Mellon University
and
École Polytechnique Fédérale de Lausanne
How my talk relates to David Dewitt’s

• Similarities:
  ▫ It is a brand new talk
  ▫ I am grateful for my GREAT systems students
  ▫ It is about using A LOT of available computation

• Differences:
  ▫ It is not about Map/Reduce
  ▫ It has an offensive number of graphs
  ▫ I am not allowed to make jokes about Mike
Hardware Integration Trends

Moore’s Law: 2x transistors ➔ 2x cores ➔ 2x caches

What is the impact on DBMS performance???
Challenge #1: Available parallelism grows exponentially

Our workloads cannot keep 100’s of cores busy
Challenge #2: On-chip L2 cache sizes grow exponentially

Larger cache \(\rightarrow\) 50% drop in performance!
Challenge #3: Cache latency = f(block location)

Accessing L2 is not simple anymore
Outline

• Overview of HW Trends and impact on DBMS

• Database workload facts
  • Cache trends
  • Core trends
  • Cordoba approach
Fact #1: Memory-resident performance is important

• Because it matters
  □ In the 70s and the 80s, only disk I/O
  □ Since the 90s, mostly memory accesses
    (large memories, smart storage managers)
  □ Companies ship memory-bound OLTP/DSS
    (100GB is a memory-resident dataset!)

• Because it tells the truth
  □ Shao04: Can study architectural behavior of
    large workloads using memory-resident
    datasets.

Memory-size workloads reveal inefficiencies
Fact #2: Data dependencies in instruction stream

Memory-level parallelism = $\text{AVG}_{\text{MissCycles}} (\#\text{OfMisses})$

MLP = 1.4 for OLTP, 3.8 for DSS (i.e., too low)*

DB workloads cannot exploit available parallelism

*Ranganathan et al., ASPLOS98
Core trends

Increasing number of cores, cannot use all of them.
Parallelism in multi-core technology

- **Fat Camp (FC)**
  - wide-issue, OOO
  - e.g., IBM Power5

- **Lean Camp (LC)**
  - in-order, multi-threaded
  - e.g., Sun UltraSparc T1

> FC: parallelism within thread, LC: across threads
## Processor Design Taxonomy

<table>
<thead>
<tr>
<th>Technology</th>
<th>Fat Camp (FC)</th>
<th>Lean Camp (LC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Issue Width</td>
<td>Wide (4+)</td>
<td>Narrow (1-2)</td>
</tr>
<tr>
<td>Execution Order</td>
<td>Out-of-Order</td>
<td>In-order</td>
</tr>
<tr>
<td>Pipeline Size</td>
<td>Deep (14+)</td>
<td>Shallow (5-6)</td>
</tr>
<tr>
<td>HW Threads</td>
<td>Few (1-2)</td>
<td>Many (4+)</td>
</tr>
<tr>
<td>#cores/chip</td>
<td>Few (x)</td>
<td>Many (3x)</td>
</tr>
</tbody>
</table>
How Camps Address Stalls

- **LC**: good for saturated workloads only
- **FC**: better for unsaturated
  (BUT DBs cannot exploit instruction-level parallelism!)

**LC**
- thread1
- thread2
- thread3

**Saturated**
(many threads)

- 1 cycle
- computation
- data stall

(time)
Performance of Saturated Workloads

Higher is better

Norm. Throughput

<table>
<thead>
<tr>
<th></th>
<th>FC</th>
<th>LC</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLTP</td>
<td>1.0</td>
<td>1.5</td>
</tr>
<tr>
<td>DSS</td>
<td>1.0</td>
<td>1.5</td>
</tr>
</tbody>
</table>

LC better because enough threads to keep cores busy
Increasing Core Count Improves Perf.  
(provided we use them all)

What if not enough threads to keep cores busy?

Fat-camp
16MB L2

OLTP: TPCC/DB2
100 wh, 64 clients

DSS: TPCH/DB2
1GB database
Core Counts Rise Exponentially

- Increasingly difficult to saturate all cores
- Workloads tend to become unsaturated
Performance of Unsaturated Workloads

With few threads, lean cores worse than fat ones
But really both camps are underutilized
Few Threads ➔ Underutilized Cores

Need aggressive parallelism to keep cores busy!
Cache trend #1

More, slower L2 may hurt performance.
Impact of Slower Caches on Execution

Even in today’s tech., 18-35% of time on cache

- OLTP: TPCC/DB2
  - 100 wh, 64 clients
- DSS: TPCH/DB2
  - 1GB database

- P4 Xeon 7100 (2006)
  - 4 cores
- PIII Xeon 500 (1999)
- Itanium2 9050 (2006)
On-chip Caches: The New Bottleneck

- Bottleneck shifts from memory to L2-hit stalls

4-core CMP
DSS: TPCH/DB2
1GB database
Time Breakdown: Cache Integration

Overall performance drops as function of cache size!

Carnegie Mellon © 2008 StagedDB/CMP project team
We lose half the potential throughput

- ▲ - DSS-const
- ▲ - DSS-real
- ▀ - OLTP-const
- ▀ - OLTP-real

Need to increase L1 locality

4-core CMP
OLTP: TPCC/DB2
100 wh, 64 clients
DSS: TPCH/DB2
1GB database
Cache trend #2

Distributed L2 makes data placement interesting.
Taming cache latency: distributed L2

Access time = f (physical location of data)
Data placement determines performance
Technique: Attract Data To Core

Fast access to core-private data
Coherence In Distributed Caches

- Slow for read-write shared data
- Why should we care?
Cache accesses: a breakdown

Shared read-write data dominate #cache accesses
Shared Accesses Dominate Cache Time

Need to maximize core-private data
Restructure software, cooperative scheduling

Cache: 0.5MB/core
OLTP: TPCC/DB2
100 WH
64 clients

© 2008 StagedDB/CMP project team
Cordoba

Locality, parallelism, and prediction of data movement.
Staged programming paradigm

Monolithic server

Conventional

One server
Request-level parallelism
Very large footprint

SOA-style server

Many services
Operator-level parallelism
Much smaller footprint

No need to change algorithms

© 2008 StagedDB/CMP project team
Staged Query Processing

Expose work sharing opportunities

Data movement predictability

© 2008 StagedDB/CMP project team
Conclusion: Relationships need work

• Hardware parallelism scales exponentially
  □ How to keep 100’s of cores busy?
    ➡️ Need to enhance software parallelism

• On-chip L2 caches grow exponentially
  □ Stalls for L2 cache hits dominate execution
    ➡️ Need to enhance L1 locality

• Data latency a function of block location in cache
  □ In-cache data placement determines perf.
    ➡️ Need to enhance data affinity to cores
Thank You!

For more information:

http://www.cs.cmu.edu/~stageddb

and

Poster Session