EECS 570
Lecture 5
Applications
Winter 2018
Prof. Satish Narayanasamy
http://www.eecs.umich.edu/courses/eecs570/
Special thanks to Babak Falsafi (EPFL) for ecocloud slides

Slides developed in part by Profs. Falsafi, Hardavellas, Nowatzyk, Mytkowicz and Wenisch of EPFL, Northwestern, CMU, Microsoft, U-M.
Announcements

Project proposal due Wednesday via Canvas

Programming Assignment 1 due Friday 2/2 11:59pm
  • Upload zip in Canvas

Project kick-off meetings – sign up to meet
Readings

For Today:


For Friday:

- Michael Scott. *Shared-Memory Synchronization*. Morgan & Claypool Synthesis Lectures on Computer Architecture (Ch. 1, 4.0-4.3.3, 5.0-5.2.5)

Applications
What is a “scientific application”

Frequent characteristics:

- Compute intensive, usually FP heavy (but not always, e.g., logic simulation, theorem proving, cryptography)
- Process large data sets
- Single problem: wall-clock time to answer matters
- Core code footprints tend to be small
  - Kernels – small pieces of critical code; typically inner loops
- Data access patterns often predictable
- Vectorization often works
Traditional Server Software (a.k.a Scale-up)

- Historically, primary market for multiprocessor systems
- Examples:
  - Database systems: Oracle, DB2, SQLServer, PostGres, MySQL
  - Business apps: SAP, BAAN, PeopleSoft
  - Data analysis: large scale graph processing
  - Web-servers
    - Static content
    - Dynamic content: database integration + business logic
    - Web 2.0: user-supplied content
  - Infrastructure apps: J2EE
Why study database apps?

• They are economically important

• They share characteristics of many other apps (filesystems, web search, etc.)

• The vendors have spent a lot of time optimizing (generally, they won’t have silly bottlenecks)
Key characteristics

• Large, complex, monolithic software systems

• Designed for MP systems
  □ Clusters (distributed databases)
  □ Shared Memory

• Subsumes many OS functions
  □ File system
  □ Scheduling and multi-threading
  □ Memory management

• Designed for high reliability (ACID properties)
  □ Atomicity: a transaction happens or doesn’t
  □ Consistency: the state of the DB remains consistent
  □ Isolation: transactions are independent
  □ Durability: once performed, transactions are permanent
  □ *Aside*: we will see these ideas pop up in architecture context again with transactional memory
How are they different from Sci Apps?

- Requires tuning: knowledge-intensive, difficult
- Competitive market: deliberate obfuscation/ benchmark gaming
- Large instruction footprints (I$ matters)
- Huge data footprints (TLBs matter)
- Weird access types (cross-endian, non-cacheable, etc.)
- Latency, not bandwidth bound
- Dynamic memory allocation, sometimes garbage collection
- More pointer-chasing, fewer arrays
- No single obvious “working set”
  - multiple working sets with varying temporal locality
- Unpredictable sharing patterns
- Data & lock contention
DBMS Structure

Source: Silberschatz, Korth, Sudarshan. Database System Concepts
Fundamental Data Structures

Page

B+ Tree

Source: Ailamaki, DeWitt & Hill

Where does time go: Microbenchmarks

- Compute time < 50% of total time

Where does time go: Memory stalls breakdown

• L1 instruction and L2 data stalls dominate

Standardized Benchmarks

- Transaction Processing Council (TPC)
  - Strict scaling, disclosure, auditing rules
  - Running these for real is hard: big hardware, 20-50 engineers, months of effort
  - Running them in simulations is also hard: scaling, non-determinism

- Two flavors of benchmark
  - Online transaction processing (OLTP): TPC-C
    - Lots of small transactions
    - Lots of locking, concurrency, I/O; memory-latency bound
  - Decision support system (DSS): TPC-H
    - Large, complex read-only queries
    - Often compute bound (given enough disks)
    - Highly parallel
      - Data partitioning
      - Parallel operators
Performance of DB Workloads on Shared Memory with OoO CPUs

[Ranganathan et al - ISCA 98]

Examines impact of ILP and multiprocessing on DSS & OLTP

- Based on extensive simulations

- Explores:
  - Multiple issue
  - Out-of-order (including window size)
  - Number of outstanding misses
  - Instruction/branch prediction effects
  - Impact of multiprocessing & memory consistency
  - Ways to mitigate instruction & coherence misses
Lots of ILP

Multiple issue helps, but OoO helps more

L2 hits account for most data stalls

Multiple outstanding misses are critical
OLTP - Impact of ILP

- Instruction cache misses & synch are now issues
- Less ILP, but 2-way still helps a lot
- Coherence (dirty) & DTLB misses cause most read stalls
- 2 outstanding misses is critical, but more doesn’t help
Impact of Multiprocessing

- Coherence misses & sync are dramatic in OLTP
Impact of Memory Consistency

- SC = sequential consistency
- PC = a system with a write buffer (loads bypass stores)
- RC = wait only at synchronization instructions

Massive performance difference!
- We will revisit this later in the course...
Cloud Computing Software (scale-out)
What about “cloud computing” software?

[Ferdman et al - ASPLOS 2012]

Emerging workloads:
• Scale out
• Often data intensive
• Like conventional server workloads

Different from CPU benchmark suites:
• Use of FP
• Do not exercise the memory hierarchy
• Similar to conventional server workloads [CIDR’07]
CloudSuite: A Benchmark Suite of Emerging Scale-Out Workloads

Publicly released

Alpha version:

- Analytics (Classification)
- Data serving (YCSB)
- Simulation (Cloud9)
- Streaming (Darwin)
- Web frontend (Cloudstone)
- Web search (Nutch)
Ran Experiments on Nehalem Blades

<table>
<thead>
<tr>
<th></th>
<th>Hardware Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processor</td>
<td>Intel Xeon 5670, 6 cores, 32nm @2.93GHz</td>
</tr>
<tr>
<td>CMP Size</td>
<td>6 OoO cores</td>
</tr>
<tr>
<td>Superscalar width</td>
<td>4-wide issue</td>
</tr>
<tr>
<td>Reorder buffer</td>
<td>128 entries</td>
</tr>
<tr>
<td>Load/Store buffer</td>
<td>48/32 entries</td>
</tr>
<tr>
<td>Reservation stations</td>
<td>36 entries</td>
</tr>
<tr>
<td>L1 Cache</td>
<td>split I/D, 32KB, 4-cycles access latency</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>6-core CMP: 256KB per core, 12-cycles access latency</td>
</tr>
<tr>
<td>LLC (L3) cache</td>
<td>12MB, cycles 39-cycles access latency</td>
</tr>
<tr>
<td>Memory</td>
<td>24GB, 180/280 cycles access latency local/remote DRAM</td>
</tr>
</tbody>
</table>

Where does time go?
Execution Breakdown

- Unlike desktop/RMS apps, memory stalls dominate
- Design should be centered around memory
Front-End Inefficiencies

- Instruction fetch: 10-60% of total stalls
- Next-line prefetch. (in the CPU) not efficient
Core Inefficiencies

- Low IPC & MLP despite 4-wide OoO core
- Using SMT doubles MLP
- But, SMT achieves only 30% performance gain
  - Threads compete for core resources
  - Intel’s SMT fetch not effective
Cache Capacity (LLC) Inefficiencies

- Large LLC consumes area, but has diminishing returns
- Results (not shown) indicate much LLC accesses are instructions
Data Prefetching Inefficiencies

- Existing prefetchers are ineffective
- Pointer-intensive patterns [Wenisch 2005]
Bandwidth Inefficiencies

- Low sharing among working threads
- No need for on-chip shared caches
- Today, pin bandwidth is overprovisioned
CloudSuite Conclusions

Corroborate prior findings [CIDR’07]
Scale-out workloads need:
• Simple (multithreaded) cores
• Partitioned caches (no sharing)
• Large on-chip instruction footprints
• Advanced prefetchers
Sirius: An Open End-to-End Voice and Vision Personal Assistant and Its Implications for Future Warehouse Scale Computers

Johann Hauswald, Michael A. Laurenzano, Yunqi Zhang, Cheng Li, Austin Rovinski, Arjun Khurana, Ron Dreslinski, Trevor Mudge, Vinicius Petrucci, Lingjia Tang, Jason Mars

University of Michigan — Ann Arbor, MI
• **Sirius**: full end-to-end with inputs, pre-trained models, and databases
• **Sirius-suite**: 7 kernels with inputs to study each service individually

DjiNN and Tonic: DNN as a Service
How does Sirius work?

Users

Voice Command (VC)  Voice Query (VQ)  Voice-Image Query (VIQ)

Automatic-Speech Recognition (ASR)  Question Answering (QA)  Image Matching (IMM)

Query Taxonomy

IPA Services

Algorithmic Components

HMM/GMM or HMM/DNN  Regular Expression  Feature Extraction

Conditional Random Fields  Stemmer  Feature Description

Automatic-Speech Recognition  Question Answering  Image Matching
Sirius-suite

7 kernels: 92% total execution of Sirius Suite entirely written in C/C++/CUDA
Release includes inputs and models

DjiNN and Tonic: DNN as a Service
## Upgrading Datacenters with COTS Systems

<table>
<thead>
<tr>
<th>Platform</th>
<th>Model</th>
<th>Clock</th>
<th>Threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicore CPU</td>
<td>Intel Xeon E3-1240 V3</td>
<td>3.40 GHz</td>
<td>8</td>
</tr>
<tr>
<td>GPU</td>
<td>NVIDIA GTX 770</td>
<td>1.05 GHz</td>
<td>12288</td>
</tr>
<tr>
<td>Intel Phi</td>
<td>Phi 5110P</td>
<td>1.05 GHz</td>
<td>240</td>
</tr>
<tr>
<td>FPGA</td>
<td>Xilinx Virtex-6 ML605</td>
<td>400 MHz</td>
<td>N/A</td>
</tr>
</tbody>
</table>
## Upgrading Datacenters with COTS Systems

<table>
<thead>
<tr>
<th>Platform</th>
<th>Advantage</th>
<th>Disadvantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multicore CPU</td>
<td>Minor SW changes</td>
<td>Limited speedup</td>
</tr>
<tr>
<td>GPU</td>
<td>Many threads</td>
<td>Programability</td>
</tr>
<tr>
<td>Intel Phi</td>
<td>Manycore</td>
<td>Limited compiler support</td>
</tr>
<tr>
<td>FPGA</td>
<td>Flexible</td>
<td>New implementation</td>
</tr>
</tbody>
</table>
## Acceleration Overview

<table>
<thead>
<tr>
<th>Platform</th>
<th>GMM</th>
<th>DNN</th>
<th>Stemmer</th>
<th>Regex</th>
<th>CRF</th>
<th>FE</th>
<th>FD</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMP</td>
<td>3.5</td>
<td>6.0</td>
<td>4.0</td>
<td>3.9</td>
<td>3.7</td>
<td>5.2</td>
<td>5.9</td>
</tr>
<tr>
<td>GPU</td>
<td>70.0</td>
<td>54.7</td>
<td>6.2</td>
<td>48.0*</td>
<td>3.8*</td>
<td>10.5</td>
<td>120.5</td>
</tr>
<tr>
<td>Intel Phi</td>
<td>1.1</td>
<td>11.2</td>
<td>5.6</td>
<td>1.1</td>
<td>4.7</td>
<td>2.5</td>
<td>12.7</td>
</tr>
<tr>
<td>FPGA</td>
<td>169.0</td>
<td>110.5*</td>
<td>30.0</td>
<td>168.2*</td>
<td>7.5*</td>
<td>34.6*</td>
<td>75.5*</td>
</tr>
</tbody>
</table>
DjiNN and Tonic: DNN as a Service and Its Implications for Future Warehouse Scale Computers

Johann Hauswald, Yiping Kang, Michael A. Laurenzano, Quan Chen, Cheng Li, Trevor Mudge, Ronald G. Dreslinski, Jason Mars, Lingjia Tang

University of Michigan — Ann Arbor, MI
Deep Neural Networks (DNNs)

Inference

"Who", "is", "this"

0.1 “Spiderman”
0.9 “Superman”
0.5 “Batman”

Network Architecture

DjiNN and Tonic: DNN as a Service
DjiNN and Tonic: DNN as a Service

Tonic Suite Applications

- Image Task
- DIG
- FACE

Image Task

- IMC
- DIG
- FACE

Speech Recognition (ASR) Task

- "It's business, Superman"

Natural Language Processing Task

- POS
  - "business" (noun)
  - "Superman" (P. noun)
  - "it's" (VP, B-NP)
  - "business" (NP, I-NP)
- CHK
  - "Superman" (PERSON)
- NER

DjiNN DNN Service

DNN Architecture

- IMC
- DIG
- FACE
- ASR
- POS
- CHK
- NER

Trained Models
DNN as a Service

Unified, highly optimized appliance for DNN