Agenda

- Great Ideas in Computer Architecture
- Administrivia
- PostPC Era: From Phones to Datacenters
- Technology Break
- Warehouse Scale Computers in Depth

CS61c is NOT really about C Programming

- It is about the hardware-software interface
  - What does the programmer need to know to achieve the highest possible performance
- Languages like C are closer to the underlying hardware, unlike languages like Scheme!
  - Allows us to talk about key hardware features in higher level terms
  - Allows programmer to explicitly harness underlying hardware parallelism for high performance

Old School CS61c

New School CS61c

Personal Mobile Devices
Warehouse Scale Computer

**New-School Machine Structures**

- **Parallel Requests**
  - Assigned to computer
  - e.g., Search "Katz"
- **Parallel Threads**
  - Assigned to core
  - e.g., Lookup, Ads
- **Parallel Instructions**
  - >1 instruction @ one time
  - e.g., 5 pipelined instructions
- **Parallel Data**
  - >1 data item @ one time
  - e.g., Add of 4 pairs of words
- **Hardware descriptions**
  - All gates functioning in parallel at same time

**Great Idea #1: Levels of Representation/Interpretation**

- **High Level Language**
  - Program in e.g., C
- **Compiler**
  - e.g., Assembly Language Program (e.g., MIPS)
- **Assembler**
  - Machine Language Program (MIPS)
- **Hardware Architecture Description**
  - e.g., block diagrams
- **Logic Circuit Description**
  - Circuit Schematic Diagrams

**6 Great Ideas in Computer Architecture**

1. Layers of Representation/Interpretation
2. Moore’s Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy
Great Idea #3: Principle of Locality/ Memory Hierarchy

Great Idea #4: Parallelism

Great Idea #5: Performance Measurement and Improvement
- Matching application to underlying hardware to exploit:
  - Locality
  - Parallelism
  - Special hardware features, like specialized instructions (e.g., matrix manipulation)
- Latency
  - How long to set the problem up
  - How much faster does it execute once it gets going
  - It is all about time to finish

Great Idea #6: Dependability via Redundancy
- Applies to everything from datacenters to storage to memory
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)

Increasing transistor density reduces the cost of redundancy.
Agenda

- Great Ideas in Computer Architecture
- Administrivia
- Technology Break
- From Phones to Datacenters

Course Information

- Instructors:
  - Randy Katz, Dave Patterson
- Teaching Assistants:
  - Andrew Gearhart, Conor Hughes, Yunsup Lee, Ari Rabkin, Charles Reiss, Andrew Waterman, Vasily Volkov
- Textbooks:
  - Average 15 pages of reading/week
    - Barros & Holst, The Datacenter as a Computer, 1st Edition
- Google Group:
  - 61CSpring2011UCB-announce: announcements from staff
  - 61CSpring2011UCB-disc: Q&A, discussion by anyone in 61C
  - Email Andrew Gearhart agearh@gmail.com to join

Reminders

- Discussions and labs will be held this week
  - Switching Sections: if you find another 61C student willing to swap discussion AND lab, talk to your TAs
  - Partner (only project 3 and extra credit): OK if partners mix sections but have same TA
- First homework assignment due this Sunday January 23rd by 11:59:59 PM
  - There is reading assignment as well on course page

Course Organization

- Grading
  - Participation and Altruism (5%)
  - Homework (5%)
  - Labs (20%)
  - Projects (40%)
    1. Data Parallelism (Map-Reduce on Amazon EC2)
    2. Computer Instruction Set Simulator (C)
    3. Performance Tuning of a Parallel Application/Matrix Multiply using cache blocking, SIMD, MIMD (OpenMP, due with partner)
    4. Computer Processor Design (Logisim)
  - Extra Credit: Matrix Multiply Competition, anything goes
    - Midterm (10%): 6-9 PM Tuesday March 8
    - Final (20%): 11:30-2:30 PM Monday May 9

EECS Grading Policy

- [http://www.eecs.berkeley.edu/Policies/ugrad_grading.shtml](http://www.eecs.berkeley.edu/Policies/ugrad_grading.shtml)
  - “A typical GPA for courses in the lower division is 2.7. This GPA would result, for example, from 17% A’s, 50% B’s, 20% C’s, 10% D’s, and 3% F’s. A class whose GPA falls outside the range 2.5 - 2.9 should be considered atypical.”
- Fall 2010: GPA 2.81
  - 26% A’s, 47% B’s, 17% C’s, 3% D’s, 6% F’s
- Fall 2009: GPA 2.71
  - 26% A’s, 47% B’s, 17% C’s, 3% D’s, 6% F’s
- Fall 2008: GPA 2.95
  - 26% A’s, 47% B’s, 17% C’s, 3% D’s, 6% F’s
- Fall 2007: GPA 2.67
  - 26% A’s, 47% B’s, 17% C’s, 3% D’s, 6% F’s
Late Policy

• Assignments due Sundays at 11:59:59 PM
• Late homeworks not accepted (100% penalty)
• Late projects get 20% penalty, accepted up to Tuesdays at 11:59:59 PM
  – No credit if more than 48 hours late
  – No “slip days” in 61C

• Used by Dan Garcia and a few faculty to cope with 100s of students who often procrastinate without having to hear the excuses, but not widespread in EECS courses
• More late assignments if everyone has no-cost options; better to learn now how to cope with real deadlines

Policy on Assignments and Independent Work

• With the exception of laboratories and assignments that explicitly permit you to work in groups, all homeworks and projects are to be YOUR work and your work ALONE.
• You are encouraged to discuss your assignments with other students, and extra credit will be assigned to students who help others, particularly by answering questions on the Google Group, but we expect that what you hand is yours.
• It is NOT acceptable to copy solutions from other students.
• It is NOT acceptable to copy (or start your) solutions from the Web.
• We have tools and methods, developed over many years, for detecting this. You WILL be caught, and the penalties WILL be severe.
• At the minimum a ZERO for the assignment, possibly an F in the course, and a letter to your university record documenting the incidence of cheating.
• (We caught people last semester!)

Architecture of a Lecture

<table>
<thead>
<tr>
<th>Time (minutes)</th>
<th>Full Attention</th>
<th>Administrivia</th>
<th>Tech break</th>
<th>“And in conclusion...”</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>50</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>53</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>80</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Agenda

• Great Ideas in Computer Architecture
• Administrivia
• PostPC Era: From Phones to Datacenters
• Technology Break
• Warehouse Scale Computer in Depth
Computer Eras: Mainframe 1950s-60s

“Big Iron”: IBM, UNIVAC, ... build $1M computers for businesses => COBOL, Fortran, timesharing OS

Minicomputer Eras: 1970s

Using integrated circuits, Digital, HP... build $10k computers for labs, universities => C, UNIX OS

PC Era: Mid 1980s - Mid 2000s

Using microprocessors, Apple, IBM, ... build $1k computer for 1 person => Basic, Java, Windows OS

PostPC Era: Late 2000s - ??

Personal Mobile Devices (PMD): Relying on wireless networking, Apple, Nokia, ... build $500 smartphone and tablet computers for individuals => Objective C, Android OS

Cloud Computing: Using Local Area Networks, Amazon, Google, ... build $200M Warehouse Scale Computers with 100,000 servers for Internet Services for PMDs => MapReduce, Ruby on Rails

Advanced RISC Machine (ARM) instruction set inside the iPhone

You will how to design and program a related RISC computer: MIPS

iPhone Innards

You will about multiple processors, data level parallelism, caches in 61C
The Big Switch: Cloud Computing

"A hundred years ago, companies stopped generating their own power with steam engines and dynamos and plugged into the newly built electric grid. The cheap power pumped out by electric utilities didn't just change how businesses operate. It set off a chain reaction of economic and social transformations that brought the modern world into existence. Today, a similar revolution is under way. Hooked up to the Internet's global computing grid, massive information-processing plants have begun pumping data and software code into our homes and businesses. This time, it's computing that's turning into a utility."

Why Cloud Computing Now?

- "The Web Space Race": Build-out of extremely large datacenters (10,000's of commodity PCs)
  - Build-out driven by growth in demand (more users)
  - Infrastructure software and Operational expertise
- Discovered economy of scale: 5-7x cheaper than provisioning a medium-sized (1000 servers) facility
- More pervasive broadband Internet so can access remote computers efficiently
- Commoditization of HW & SW
  - Standardized software stacks

Coping with Failures

- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

Agenda

- Great Ideas in Computer Architecture
- Administrivia
- PostPC Era: From Phones to Datacenters
- Technology Break
- Warehouse Scale Computers in Depth

Coping with Failures

- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

Warehouse Scale Computers

- Massive scale datacenters: 10,000 to 100,000 servers + networks to connect them together
  - Emphasize cost-efficiency
  - Attention to power: distribution and cooling
- Homogeneous hardware/software
- Offer small number of very large applications (Internet services): search, social networking, video sharing
- Very highly available: <1 hour down/year
  - Must cope with failures common at scale
E.g., Google’s Oregon WSC

Equipment Inside a WSC

Server (in rack format): 1 ¾ inches high “1U”, x 19 inches x 16-20 inches: 8 cores, 16 GB DRAM, 4x1 TB disk

Array (aka cluster): 16-32 server racks + larger local area network switch (“array switch”) 10X faster => cost 100X: cost f(N^3)

Server, Rack, Array

Google Server Internals

Coping with Performance in Array

Lower latency to DRAM in another server than local disk
Higher bandwidth to local disk than to DRAM in another server

<table>
<thead>
<tr>
<th></th>
<th>Local</th>
<th>Rack</th>
<th>Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Racks</td>
<td>8</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>Servers</td>
<td>8</td>
<td>80</td>
<td>2400</td>
</tr>
<tr>
<td>Cores (Processors)</td>
<td>8</td>
<td>640</td>
<td>19,200</td>
</tr>
<tr>
<td>DRAM Capacity (GB)</td>
<td>16</td>
<td>1,280</td>
<td>38,400</td>
</tr>
<tr>
<td>Disk Capacity (GB)</td>
<td>4,000</td>
<td>320,000</td>
<td>9,600,000</td>
</tr>
<tr>
<td>DRAM Latency (microseconds)</td>
<td>0.1</td>
<td>100</td>
<td>300</td>
</tr>
<tr>
<td>Disk Latency (microseconds)</td>
<td>10,000</td>
<td>11,000</td>
<td>12,000</td>
</tr>
<tr>
<td>DRAM Bandwidth (MB/sec)</td>
<td>20,000</td>
<td>100</td>
<td>10</td>
</tr>
<tr>
<td>Disk Bandwidth (MB/sec)</td>
<td>200</td>
<td>100</td>
<td>10</td>
</tr>
</tbody>
</table>

Datacenter Power

- DRAM: 30%
- Disks: 10%
- CPUs: 33%
- Other (server): 22%
- Network: 17%
- Peak Power %
**Coping with Workload Variation**

- Online service: Peak usage 2X off-peak

**Impact of latency, bandwidth, failure, varying workload on WSC software?**

- WSC Software must take care where it places data within an array to get good performance
- WSC Software must cope with failures gracefully
- WSC Software must scale up and down gracefully in response to varying demand
- More elaborate hierarchy of memories, failure tolerance, workload accommodation makes WSC software development more challenging than software for single computer

**Power vs. Server Utilization**

- Server power usage as load varies idle to 100%
- Uses ½ peak power when idle!
- Uses ½ peak power when 10% utilized! 90% @ 50%
- Most servers in WSC utilized 10% to 50%
- Goal should be **Energy-Proportionality**: % peak load = % peak energy

**Power Usage Effectiveness**

- Overall WSC Energy Efficiency: amount of computational work performed divided by the total energy used in the process
- Power Usage Effectiveness (PUE): Total building power / IT equipment power
  - An power efficiency measure for WSC, not including efficiency of servers, networking gear
  - 1.0 = perfection

**PUE in the Wild (2007)**

**High PUE: Where Does Power Go?**

- Uninterruptable Power Supply (battery)
- Power Distribution Unit
- Servers + Networking
- Computer Room Air Conditioner
- Chiller cools warm water from Air Conditioner
- FIGURE 5.1 LBNL survey of the power usage efficiency of 24 datacenters, 2007 (Gonsberg et al.)
Google WSC A PUE: 1.24

1. Careful air flow handling
   • Don’t mix server hot air exhaust with cold air (separate warm aisle from cold aisle)
   • Short path to cooling so little energy spent moving cold or hot air long distances
   • Keeping servers inside containers helps control air flow

Google WSC A PUE: 1.24

2. Elevated cold aisle temperatures
   • 81°F instead of traditional 65°- 68°F
   • Found reliability OK if run servers hotter

3. Use of free cooling
   • Cool warm water outside by evaporation in cooling towers
   • Locate WSC in moderate climate so not too hot or too cold

Google WSC A PUE: 1.24

4. Per-server 12-V DC UPS
   • Rather than WSC wide UPS, place single battery per server board
   • Increases WSC efficiency from 90% to 99%

5. Measure vs. estimate PUE, publish PUE, and improve operation

Google WSC PUE: Quarterly Avg

PUE

- www.google.com/corporate/green/datacenters/measuring.htm

Summary

- CS61c: Learn 6 great ideas in computer architecture to enable high performance programming via parallelism, not just learn C
  1. Layers of Representation/Interpretation
  2. Moore’s Law
  3. Principle of Locality/Memory Hierarchy
  4. Parallelism
  5. Performance Measurement and Improvement
  6. Dependability via Redundancy
- Post PC Era: Parallel processing, smart phone to WSC
- WSC SW must cope with failures, varying load, varying HW latency bandwidth
- WSC HW sensitive to cost, energy efficiency