CS 61C:
Great Ideas in Computer Architecture

Course Summary and Wrap-up

Instructor:
Sung Roa Yoon

Slides:
Randy H. Katz

http://inst.eecs.Berkeley.edu/~cs61c/fa13
New “Great Ideas”
RISC Machines + Cloud Servers
My other computer is a data center
CS61c is NOT about C Programming

• It’s 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 Python!
  – Allows us to talk about key hardware features in higher level terms
  – Allows programmer to explicitly harness underlying hardware parallelism for high performance: “programming for performance”
New-School Machine Structures
(It’s a bit more complicated!)

- **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

- **Programming Languages**

---

**Leverage Parallelism & Achieve High Performance**

**Project 1**

**Project 2**

**Project 3**

**Project 4**

**Computer**

- Core
- Memory
- Input/Output

**Hardware**

- Warehouse Scale Computer

**Software**

- Smart Phone
- Software
- Leverage Parallelism & Achieve High Performance
- Computer
- Logic Gates
- Project 4

---

Fall 2013 -- Lecture #1
Great Ideas in Computer Architecture

1. Design for Moore’s Law
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. Dependability via Redundancy
5. Memory Hierarchy
6. Performance via Parallelism/Pipelining/Prediction
Powers of Ten inspired 61C Overview

• Going Top Down cover 3 Views
1. Architecture (when possible)
2. Physical Implementation of that architecture
3. Programming system for that architecture and implementation (when possible)

• See http://www.powersof10.com/film
Earth

$10^7$ meters
The Dalles, Oregon \(10^4\) meters
The Dalles, Oregon \(10^4\) meters
Google’s Oregon WSC $10^3$ meters
Google Warehouse

- 90 meters by 75 meters, 10 Megawatts
- Contains 40,000 servers, 190,000 disks
- Power Utilization Effectiveness: 1.23
  - 85% of 0.23 overhead goes to cooling losses
  - 15% of 0.23 overhead goes to power losses
- Contains 45, 40-foot long containers
  - 8 feet x 9.5 feet x 40 feet
- 30 stacked as double layer, 15 as single layer
Containers in WSCs

$10^2$ meters

100 meters
Google Container

$10^1$ meters
Google Container

- 2 long rows, each with 29 racks
- Cooling below raised floor
- Hot air returned behind racks
Server (in rack format):

7 foot Rack: servers + Ethernet local area network switch in middle ("rack switch")

Array (aka cluster):
server racks + larger local area network switch ("array switch") 10X faster => cost 100X: cost $f(N^2)$
Google Rack

• Google rack with 20 servers + Network Switch in the middle
• 48-port 1 Gigabit/sec Ethernet switch every other rack
• Array switches connect to racks via multiple 1 Gbit/s links
• 2 datacenter routers connect to array switches over 10 Gbit/s links
public static class SumReduce extends Reducer<Text, LongWritable, Text, LongWritable> {

/** Actual reduce function.
 * @param key Word.
 * @param values Values for this word (partial counts).
 * @param context ReducerContext object for accessing output,
 * configuration information, etc.
 */

@Override
public void reduce(Text key, Iterable<LongWritable> values,
                   Context context) throws IOException, InterruptedException {
    long sum = 0L;
    for (LongWritable value : values) {
        sum += value.get();
    }
    context.write(key, new LongWritable(sum));
}
}
Great Ideas in Computer Architecture

1. Design for Moore’s Law
   -- WSC, Container, Rack
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. Dependability via Redundancy
   -- Multiple WSCs, Multiple Racks, Multiple Switches
5. Memory Hierarchy
6. Performance via Parallelism/Pipelining/Prediction
   -- Task level Parallelism, Data Level Parallelism
Google Server Internals

10^{-1} \text{ meters}
Google Board Details

• Supplies only 12 volts
• Battery per board vs. large battery room
  – Improves PUE: 99.99% efficient local battery vs 94% for battery room
• 2 SATA Disk Drives
  – 1 Terabyte capacity each
  – 3.5 inch disk drive
  – 7200 RPM
• 2 AMD Opteron Microprocessors
  – Dual Core, 2.2 GHz
• 8 DIMMs
  – 8 GB DDR2 DRAM
• 1 Gbit/sec Ethernet Network Interface Card
#include <omp.h>
#include <stdio.h>
static long num_steps = 100000;
int value[num_steps];
int reduce()
{
    int i;
    int sum = 0;

    #pragma omp parallel for private(x) reduction(+:sum)
    for (i=1; i<= num_steps; i++)
    {
        sum = sum + value[i];
    }

}
Great Ideas in Computer Architecture

1. **Design for Moore’s Law**
   -- More transistors = Multicore + SIMD
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. Dependability via Redundancy
5. **Memory Hierarchy**
   -- More transistors = Cache Memories
6. **Performance via Parallelism/Pipelining/Prediction**
   -- Thread-level Parallelism
centimeters

10-2 meters

AMD Opteron Microprocessor

12/8/13

Fall 2013 -- Lecture #27
AMD Opteron Microarchitecture

Instruction cache

Instruction prefetch and decode

RISC-operation queue

Dispatch and register remaining

Register file

Branch prediction

72 physical registers

Integer and floating-point operation queue

Integer ALU, Multiplier

Integer ALU

Integer ALU

Floating point Adder /SSE

Floating point Multiplier /SSE

Floating point Misc

Load/Store queue

Data cache

Commit unit
AMD Opteron Pipeline Flow

- For integer operations

12 stages (Floating Point is 17 stages)
- Up to 106 RISC-ops in progress
AMD Opteron Microprocessor
AMD Opteron Core

10^{-3} \text{ meters}
void mmult(int n, float *A, float *B, float *C) {
    for ( int i = 0; i < n; i+=4 )
        for ( int j = 0; j < n; j++ )
        {
            __m128 c0 = _mm_load_ps(C+i+j*n);
            for( int k = 0; k < n; k++ )
                c0 = _mm_add_ps(c0, _mm_mul_ps(_mm_load_ps(A+i+k*n),
                                               _mm_load1_ps(B+k+j*n)));
            _mm_store_ps(C+i+j*n, c0);
        }
}
Inner loop from gcc –O -S

Assembly snippet from innermost loop:

```assembly
movaps (%rax), %xmm9
mulps %xmm0, %xmm9
addps %xmm9, %xmm8
movaps 16(%rax), %xmm9
mulps %xmm0, %xmm9
addps %xmm9, %xmm7
movaps 32(%rax), %xmm9
mulps %xmm0, %xmm9
addps %xmm9, %xmm6
movaps 48(%rax), %xmm9
mulps %xmm0, %xmm9
addps %xmm9, %xmm5
```
Great Ideas in Computer Architecture

1. Design for Moore’s Law
2. Abstraction to Simplify Design
   -- *Instruction Set Architecture, Micro-operations*
3. Make the Common Case Fast
4. Dependability via Redundancy
5. Memory Hierarchy
6. *Performance via Parallelism/Pipelining/Prediction*
   -- *Instruction-level Parallelism (superscalar, pipelining)*
   -- *Data-level Parallelism*
SIMD Adder

- Four 32-bit adders that operate in parallel
  - Data Level Parallelism
One 32-bit Adder
1 bit of 32-bit Adder
Complementary MOS Transistors (NMOS and PMOS) of NAND Gate

\[ X \quad Y \quad Z \]

\[ \begin{array}{ccc}
0 \text{ volts} & 0 \text{ volts} & 3 \text{ volts} \\
0 \text{ volts} & 3 \text{ volts} & 3 \text{ volts} \\
3 \text{ volts} & 0 \text{ volts} & 3 \text{ volts} \\
3 \text{ volts} & 3 \text{ volts} & 0 \text{ volts}
\end{array} \]
Physical Layout of NAND Gate

100 nanometers

10^{-7} meters
Scanning Electron Microscope

100 nanometers

Top View

Cross Section

10^{-7} meters
Block Diagram of Static RAM

10^{-6} meters
1 Bit SRAM in 6 Transistors
Physical Layout of SRAM Bit

100 nanometers

10^{-7} meters
SRAM Cross Section

100 nanometers

10^{-7} meters
DIMM Module

• DDR = Double Data Rate
  – Transfers bits on Falling AND Rising Clock Edge
• Has Single Error Correcting, Double Error Detecting Redundancy (SEC/DED)
  – 72 bits to store 64 bits of data
  – Uses “Chip kill” organization so that if single DRAM chip fails can still detect failure
• Average server has 22,000 correctable errors and 1 uncorrectable error per year
DRAM Bits

1 micron

$10^{-6}$ meters
DRAM Cell in Transistors
Physical Layout of DRAM Bit
Cross Section of DRAM Bits

100 nanometers

10^{-7} meters
AMD Dependability

- L1 cache data is SEC/DED protected
- L2 cache and tags are SEC/DED protected
- DRAM is SEC/DED protected with chipkill
- On-chip and off-chip ECC protected arrays include autonomous, background hardware scrubbers
- Remaining arrays are parity protected
  - Instruction cache, tags and TLBs
  - Data tags and TLBs
  - Generally read only data that can be recovered from lower levels
Programming Memory Hierarchy: Cache Blocked Algorithm

- The blocked version of the i-j-k algorithm is written simply as (A,B,C are submatricies of a, b, c)

\[
\text{for } (i=0; i<N/r; i++) \\
\text{  for } (j=0; j<N/r; j++) \\
\text{    for } (k=0; k<N/r; k++) \\
\text{      } C[i][j] += A[i][k]*B[k][j]
\]

- \( r \) = block (sub-matrix) size (Assume \( r \) divides \( N \))
- \( X[i][j] \) = a sub-matrix of \( X \), defined by block row \( i \) and block column \( j \)
Great Ideas in Computer Architecture

1. Design for Moore’s Law
   -- Higher capacities caches and DRAM
2. Abstraction to Simplify Design
3. Make the Common Case Fast
4. Dependability via Redundancy
   -- Parity, SEC/DEC
5. Memory Hierarchy
   -- Caches, TLBs
6. Performance via Parallelism/Pipelining/Prediction
   -- Data-level Parallelism
Course Summary

• As the field changes, cs61c had to change too!
• It is still about the software-hardware interface
  – Programming for performance!
  – Parallelism: Task-, Thread-, Instruction-, and Data-
    MapReduce, OpenMP, C, SSE intrinsics
  – Understanding the memory hierarchy and its impact on application performance
• Interviewers ask what you did this semester!
The Future for Future Cal Alumni

• What’s The Future?
• New Century, Many New Opportunities: Parallelism, Cloud, Statistics + CS, Bio + CS, Society (Health Care, 3rd world) + CS
• Cal heritage as future alumni
  – Hard Working / Can do attitude
  – Never Give Up (“Don’t fall with the ball!”)
  – Smallest on field, 3 big guys charging you: you make a play!
• “The best way to predict the future is to invent it” – Alan Kay (inventor of personal computing vision)
• Future is up to you!