CS 61C: Great Ideas in Computer Architecture (Machine Structures)

Course Introduction

Instructors:
Randy H. Katz
David A. Patterson
http://inst.eecs.Berkeley.edu/~cs61c/sp11

Agenda

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

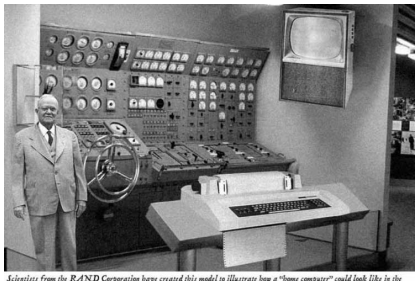
Agenda

- 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

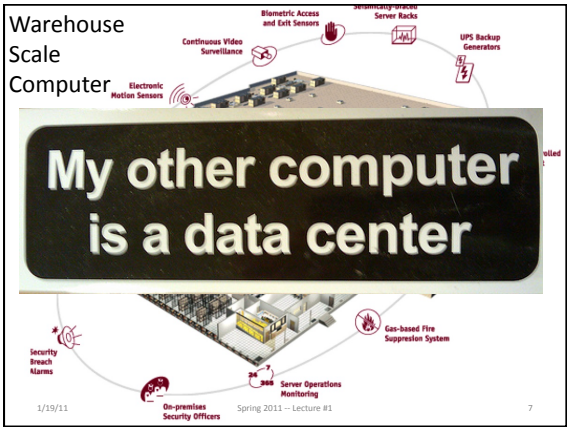


Scientists from the RAND Corporation have created this model to illustrate how a "home computer" could look like in the year 2004. However the needed technology will not be economically feasible for the average home. Also the scientists readily admit that the computer will require not yet invented technology to actually work, but 10 years from now scientific progress is expected to solve these problems. With teletype interface and the Fortran language, the computer will be easy to use.

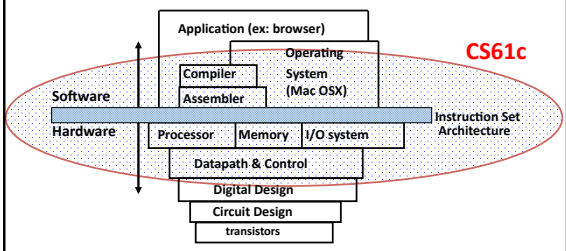
New School CS61c

Personal Mobile Devices

Warehouse Scale Computer



Old-School Machine Structures



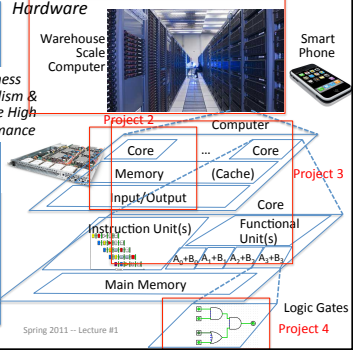
New-School Machine Structures (It's a bit more complicated!)

Project 1

Software *Hardware*

- 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

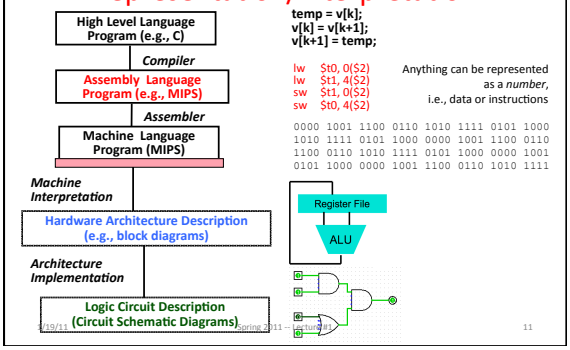
Harness Parallelism & Achieve High Performance



6 Great Ideas in Computer Architecture

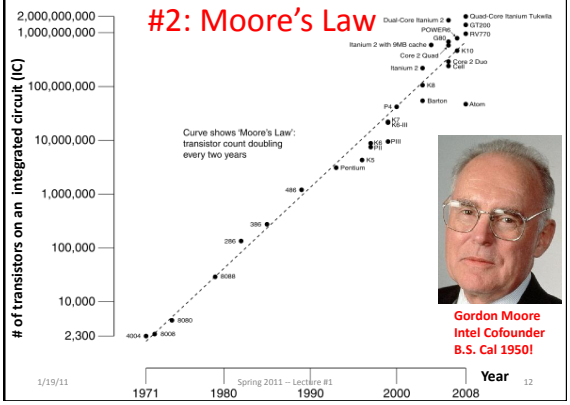
1. Layers of Representation/Interpretation
2. Moore's Law
3. Principle of Locality/Memory Hierarchy
4. Parallelism
5. Performance Measurement & Improvement
6. Dependability via Redundancy

Great Idea #1: Levels of Representation/Interpretation

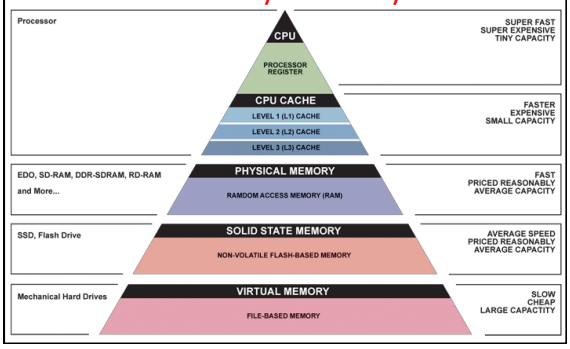


#2: Moore's Law

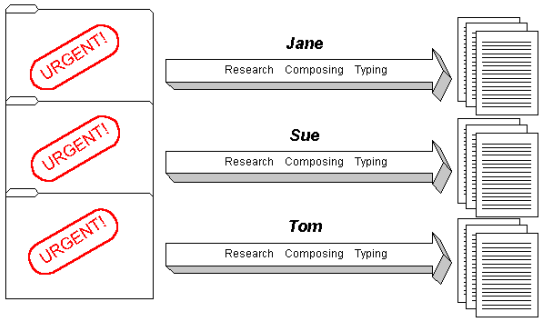
Predicts: 2X Transistors / chip every 2 years



Great Idea #3: Principle of Locality/ Memory Hierarchy



Great Idea #4: Parallelism

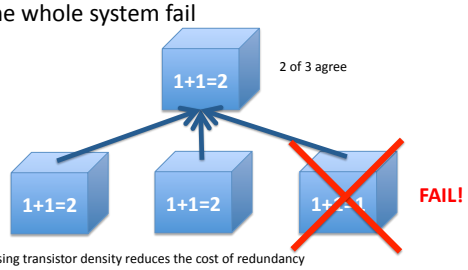


Great Idea #5: Performance Measurement and Improvement

- 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

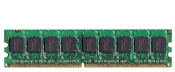
- Redundancy so that a failing piece doesn't make the whole system fail



Increasing transistor density reduces the cost of redundancy

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 of so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)



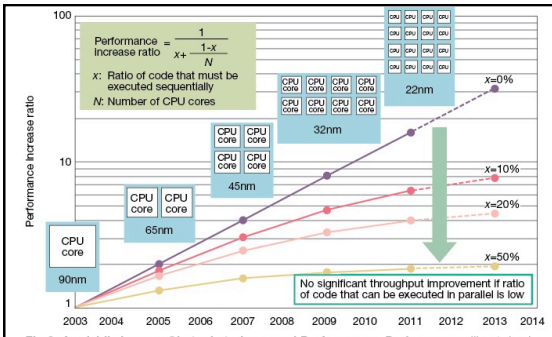
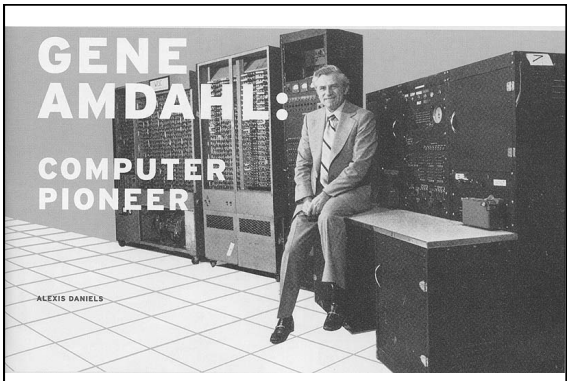


Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.



Agenda

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

Course Information

- Course Web: http://inst.eecs.Berkeley.edu/~cs61c/sp11
- 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
  - Patterson & Hennessey, *Computer Organization and Design*, 4th Edition (not ≤3rd Edition, not Asian version 4th edition)
  - Kernighan & Ritchie, *The C Programming Language*, 2nd Edition
  - Barroso & Holzle, *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
  "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
- Job/Intern Interviews: They grill you with technical questions, so it's what you say, not your GPA
  (New 61c gives good stuff to say)

	Fall	Spring
2010	2.81	2.81
2009	2.71	2.81
2008	2.95	2.74
2007	2.67	2.76

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!)

SAN FRANCISCO — It's 1 p.m. on a Thursday and Dianne Bates, 40, juggles three screens. She listens to a few songs on her iPod, then taps out a quick e-mail on her iPhone and turns her attention to the high-definition television.

Just another day at the gym.

Your Brain on Computers

At the University of California, San Francisco, scientists have found that when rats have a new experience, like exploring an unfamiliar area, their brains show new patterns of activity. But only when the rats take a break from their exploration do they process those patterns in a way that seems to create a persistent memory of the experience.

Interactive Feature

The Unplugged Challenge

Enlarge This Image

Jim Wilson/The New York Times

Loren Frank, a professor of physiology, said downtime lets the brain go over experiences, "solidify them and turn them into permanent

FACEBOOK TWITTER RECOMMEND COMMENTS (206) SIGN IN TO E-MAIL PRINT SINGLE PAGE REPRINTS SHARE

BLACK SWAN DEC. 1

itasks, she is also in fast loops on an in a downtown is in good and elsewhere, and other to get work done — ntidote to boredom.

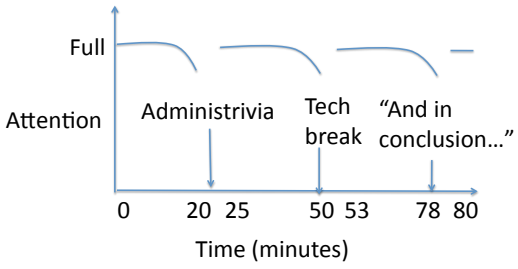
hich in the last few years have become full-fledged with high-speed Internet connections, let people relie ium of exercising, the grocery store line, stoplights o in the dinner conversation.

The technology makes the tiniest windows of time entertaining, and potentially productive. But scientists point to an unanticipated side effect: when people keep their brains busy with digital input, they are forfeiting downtime that could allow them to better learn and remember information, or come up with new ideas.

Ms. Bates, for example, might be clearer-headed if she went for a run outside, away from her devices, research suggests.

The Rules (and we really mean it!)

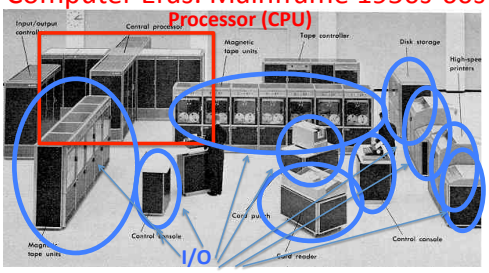
Architecture of a Lecture



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

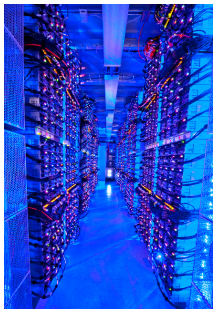
PC Era: Mid 1980s - Mid 2000s

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

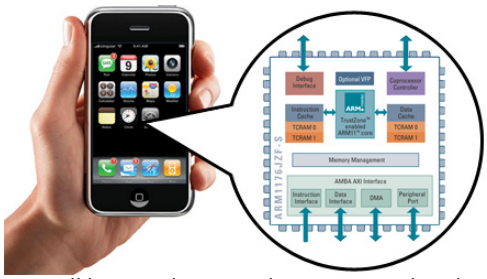
PostPC Era: Late 2000s - ??

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

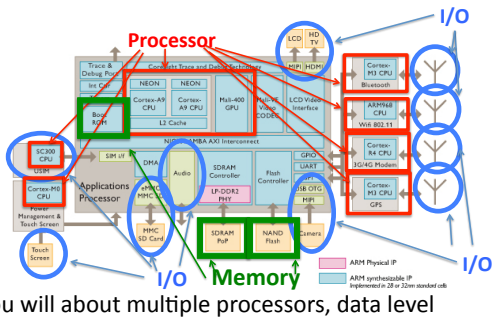


Advanced RISC Machine (ARM) instruction set inside the iPhone



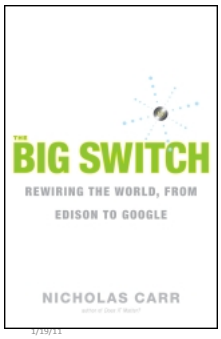
You will how to design and program a related RISC computer: MIPS

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 fails?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

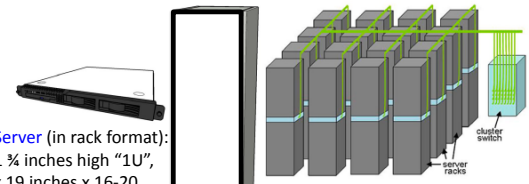
50,000 x 4 = 200,000 disks
200,000 x 4% = 8000 disks fail
365 days x 24 hours = 8760 hours

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



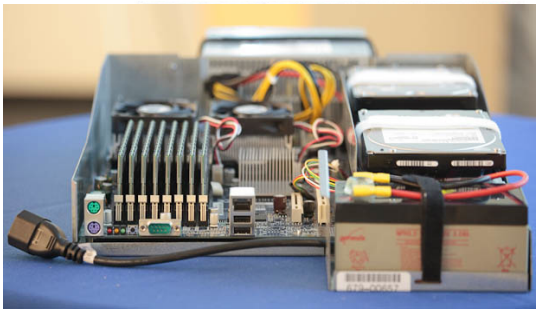
Server (in rack format): 1 ¾ inches high "1U", x 19 inches x 16-20 inches: 8 cores, 16 GB DRAM, 4x1 TB disk

7 foot Rack: 40-80 servers + Ethernet local area network (1-10 Gbps) switch in middle ("rack switch")

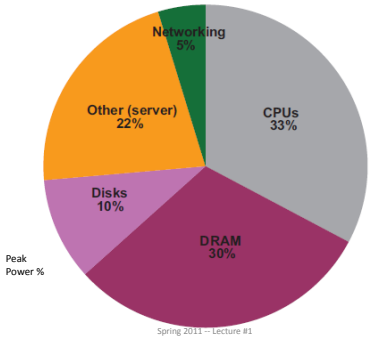
Array (aka cluster): 16-32 server racks + larger local area network switch ("array switch") 10X faster => cost 100X: cost f(N^2)

Server, Rack, Array

Google Server Internals



Datacenter Power

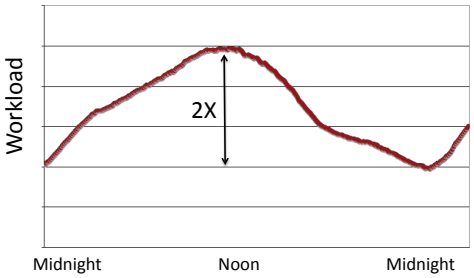


Coping with Performance in Array

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

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

Coping with Workload Variation

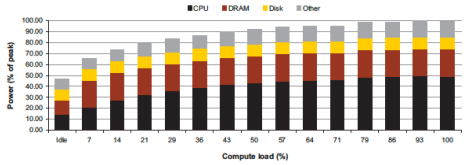


- Online service: Peak usage 2X off-peak

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 ⅔ 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)

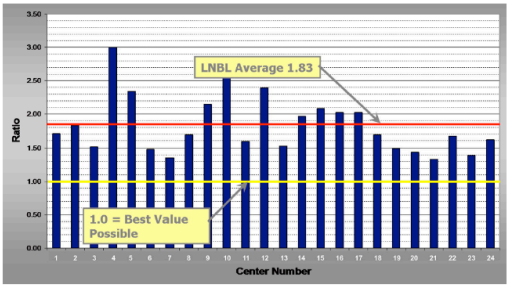
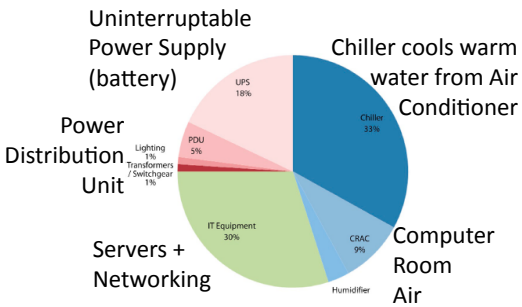


FIGURE 5.1: LBNL survey of the power usage efficiency of 24 datacenters, 2007 (Greenberg et al.)

High PUE: Where Does Power Go?



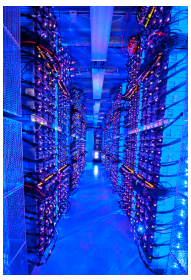
Google WSC A PUE: 1.24

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

Containers in WSCs

Inside WSC

Inside Container



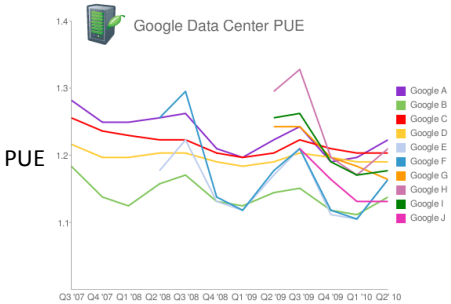
Google WSC A PUE: 1.24

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



- 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