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

Instructors:
Krste Asanovic
Randy H. Katz
http://inst.eecs.Berkeley.edu/~cs61c/F12

Agenda

- Warehouse Scale Computers
- Administrivia
- Data Parallel Map-Reduce

Agenda

- Warehouse Scale Computers
- Administrivia
- Data Parallel Map-Reduce

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

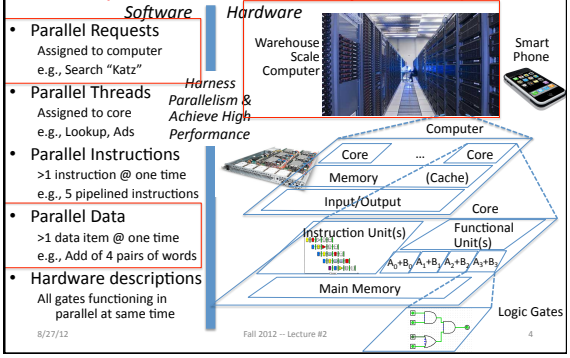
Project 1

Software

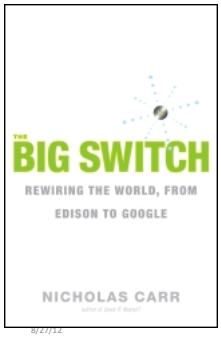
- Parallel Requests
  Assigned to computer
  e.g., Search "Katz"
- Parallel Threads
  Assigned to core
  e.g., Lookup, Ads
- Parallel Instructions
  >1 instruction @ one time
  e.g., 5 pipelined instructions
- Parallel Data
  >1 data item @ one time
  e.g., Add of 4 pairs of words
- Hardware descriptions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

Harness Parallelism & Achieve High Performance

Hardware



The Big Switch: Cloud Computing



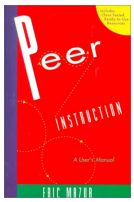
"A hundred years ago, companies stopped generating their own power with steam engines and dynamos and plugged into the newly built electric grid. The cheap power pumped out by electric utilities didn't just change how businesses operate. It set off a chain reaction of economic and social transformations 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

Peer Instruction

- Increase real-time learning in lecture, test understanding of concepts vs. details
  mazur-www.harvard.edu/education/pi.phtml
- As complete a "segment" ask multiple choice question
  - <1 minute: decide yourself, vote
  - <2 minutes: discuss in pairs, then team vote; flash card to pick answer
    - Try to convince partner; learn by teaching
- Mark and save flash cards (handed out in lecture last week)



1 2 3 4

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

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

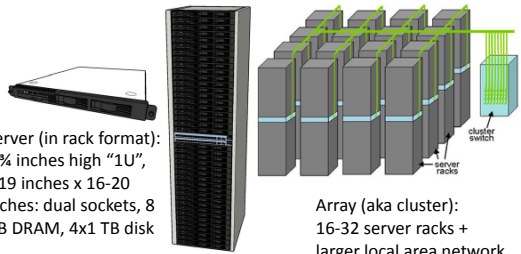
50,000 x 4 = 200,000 disks
200,000 x 4% = 8000 disks fail
365 days x 24 hours = 8760 hours

Warehouse Scale Computers

- Massive scale datacenters: 10,000 to 100,000 servers + networks to connect them together
  - Emphasize cost-efficiency: performance/$
  - 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

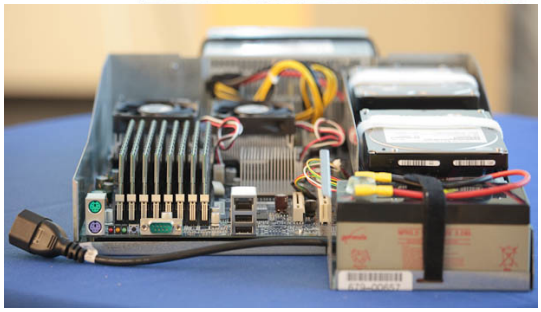


Server (in rack format): 1 ¾ inches high "1U", x 19 inches x 16-20 inches: dual sockets, 8 GB DRAM, 4x1 TB disk

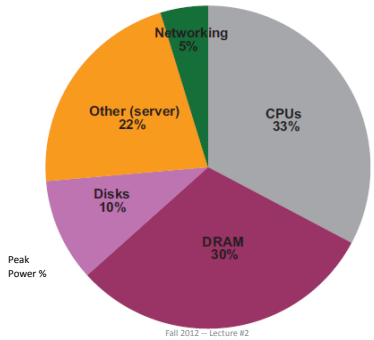
7 foot rack: 40-80 servers + Ethernet local area network (1-10 Gbps) switch in middle ("rack switch")

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

Server Internals



Datacenter Power



Coping with Performance in Array

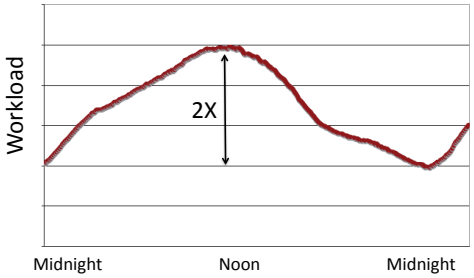
Lower latency to DRAM in another server than local disk

Higher bandwidth to local disk than to DRAM in another server

	Local	Rack	Array
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
DRAM Capacity (GB)	16	1,040	31,200
Disk Capacity (GB)	4,000	320,000	9,600,000

- 2 microprocessors, 8 GB DRAM, 4 1TB disks/server
- 80 servers/rack
- 30 racks / array => 2400 servers / array

Coping with Workload Variation

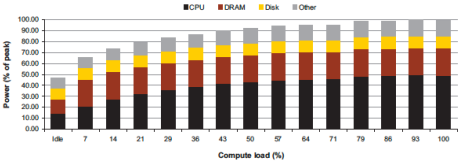


- Online service: Peak usage 2X off-peak

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)

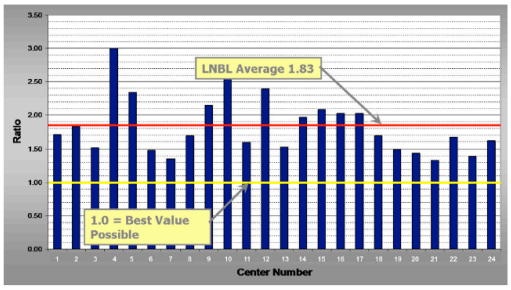
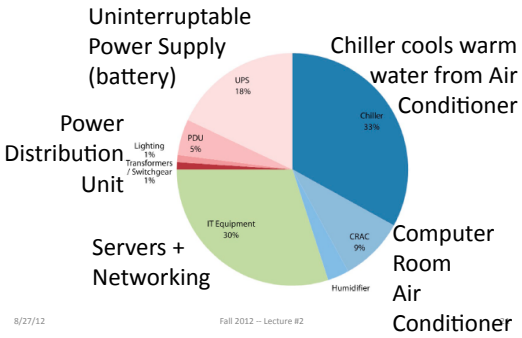


FIGURE 5.1: LBNL survey of the power usage efficiency of 24 datacenters, 2007 (Greenberg et al.)

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

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

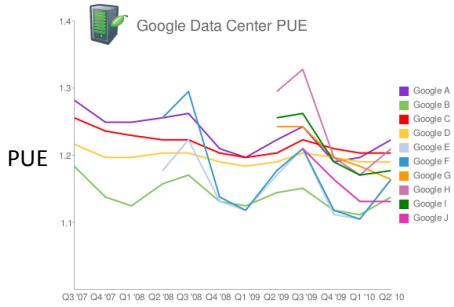
3. Use of free cooling
- Cool warm water outside by evaporation in cooling towers
- Locate WSC in moderate climate so not too hot or too cold

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

Agenda

- Warehouse Scale Computers
- Administrivia
- Data Parallel Map Reduce

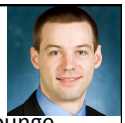
Reminders

- Labs begin THIS week (W, Th)
  - Part of first lab is discussion relevant to first HW
  - Switching Sections: if you find another 61C student willing to swap discussion AND lab, talk to your TAs
  - Partner (only Project 3 and EC): OK if partners mix sections but have same TA
- New Labs: W 1-3, W 3-5, W 9-11 (Nite Owls) in SDH 200 (original labs are in Soda 330)
- Discussions start week AFTER Labor Day
- First HW assignment due 2 September by 11:59:59 PM
  - Reading assignment on course page

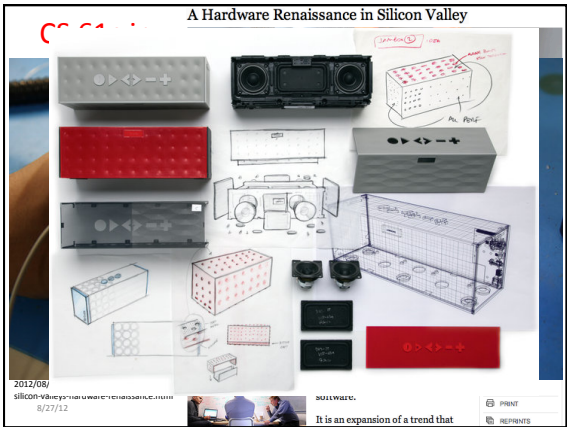
Late Policy

- 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

CS61c in the News



A Hardware Renaissance in Silicon Valley



2012/08/
silicon-valleys-hardware-renaissance.html

software.

It is an expansion of a trend that

PRINT
REPRINTS

CS61c in the News

- Talk *today* at 2 PM in Soda Hall Woz Lounge
  - "Efficiency Challenges in Warehouse-Scale Computers", Prof. Thomas Wenisch, UMich

ABSTRACT: Architects and circuit designers have made enormous strides in managing the energy efficiency and peak power demands of processors and other silicon systems. Sophisticated power management features and modes are now myriad across system components, from DRAM to processors to disks. And yet, despite these advances, typical data centers today suffer embarrassing energy inefficiencies. … In this talk, I discuss what, if anything, can be done to make datacenters more energy-proportional. Specifically, through a case study of Google's Web Search application, I will discuss the applicability of existing and proposed active and idle low-power modes to reduce the power consumed by the primary server components (processor, memory, and disk), while maintaining tight response time constraints, particularly on 95th-percentile latency.

Undergrads are welcome!

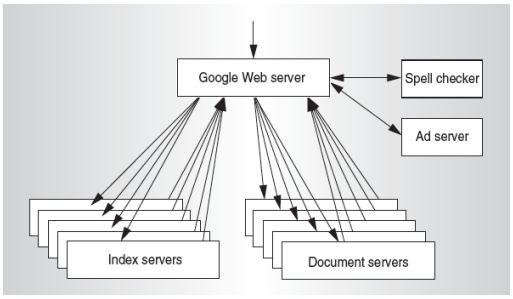
Agenda

- Warehouse Scale Computers
- Administrivia
- Data Parallel Map Reduce (Introduction)

Request-Level Parallelism (RLP)

- Hundreds or thousands of requests per second
  - Not your laptop or cell-phone, but popular Internet services like Google search
  - Such requests are largely independent
    - Mostly involve read-only databases
    - Little read-write (aka "producer-consumer") sharing
    - Rarely involve read–write data sharing or synchronization across requests
- Computation easily partitioned within a request and across different requests

Google Query-Serving Architecture

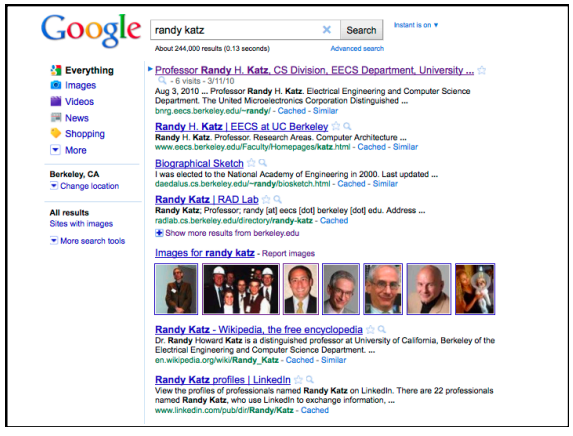


Anatomy of a Web Search

- Google "Randy H. Katz"
  1. Direct request to "closest" Google Warehouse Scale Computer
  2. Front-end load balancer directs request to one of many clusters of servers within WSC
  3. Within cluster, select one of many Google Web Servers (GWS) to handle the request and compose the response pages
  4. GWS communicates with Index Servers to find documents that contain the search words, "Randy", "Katz", uses location of search as well
  5. Return document list with associated relevance score

Anatomy of a Web Search

- In parallel,
  - Ad system: books by Katz at Amazon.com
  - Images of Randy Katz
- Use docids (document IDs) to access indexed documents
- Compose the page
  - Result document extracts (with keyword in context) ordered by relevance score
  - Sponsored links (along the top) and advertisements (along the sides)



Anatomy of a Web Search

- Implementation strategy
  - Randomly distribute the entries
  - Make many copies of data (aka "replicas")
  - Load balance requests across replicas
- Redundant copies of indices and documents
  - Breaks up hot spots, e.g., "Justin Bieber"
  - Increases opportunities for request-level parallelism
  - Makes the system more tolerant of failures

Question: Which statements are NOT TRUE about about Request Level Parallelism?

- ☐ RLP runs naturally independent requests in parallel
- ☐ RLP also runs independent tasks within a request
- ☐ RLP typically uses equal number of reads and writes
- ☐ Search uses redundant copies of indices and data to deliver parallelism

Question: Which statements are NOT TRUE about about Request Level Parallelism?

- ☐ RLP runs naturally independent requests in parallel
- ☐ RLP also runs independent tasks within a request
- ☐ RLP typically uses equal number of reads and writes
- ☐ Search uses redundant copies of indices and data to deliver parallelism

Data-Level Parallelism (DLP)

- Two kinds
  - Lots of data in memory that can be operated on in parallel (e.g., adding together two arrays)
  - Lots of data on many disks that can be operated on in parallel (e.g., searching for documents)
- 3rd project does memory-based Data Level Parallelism (DLP)
- 1st project does DLP across 1000s of servers and disks using MapReduce

Problem Trying To Solve

- How process large amounts of raw data (crawled documents, request logs, …) every day to compute derived data (inverted indicies, page popularity, …) when computation conceptually simple but input data large and distributed across 100s to 1000s of servers so that finish in reasonable time?
- Challenge: Parallelize computation, distribute data, tolerate faults without obscuring simple computation with complex code to deal with issues
- Jeffrey Dean and Sanjay Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Communications of the ACM*, Jan 2008.

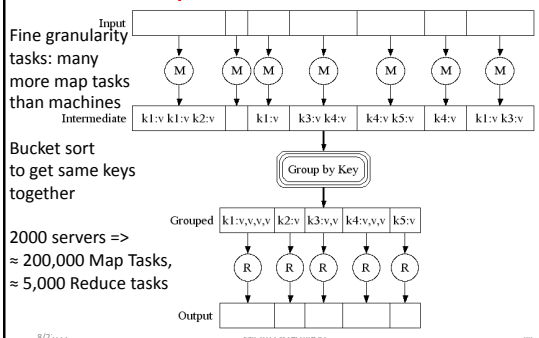
MapReduce Solution

- Apply Map function to user supplied record of key/value pairs
- Compute set of intermediate key/value pairs
- Apply Reduce operation to all values that share same key to combine derived data properly
  - Often produces smaller set of values
  - Typically 0 or 1 output value per Reduce invocation
- User supplies Map and Reduce operations in functional model so can parallelize, re-execute for fault tolerance

Data-Parallel "Divide and Conquer" (MapReduce Processing)

- Map:
  - Slice data into "shards" or "splits", distribute these to workers, compute sub-problem solutions
    `map(in_key,in_value)->list(out_key,intermediate value)`
    - Processes input key/value pair
    - Produces set of intermediate pairs
- Reduce:
  - Collect and combine sub-problem solutions
    `reduce(out_key,list(intermediate_value))->list(out_value)`
    - Combines all intermediate values for a particular key
    - Produces a set of merged output values (usually just one)
- Fun to use: focus on problem, let MapReduce library deal with messy details

MapReduce Execution



Google Uses MapReduce For …

- Web crawl: Find outgoing links from HTML documents, aggregate by target document
- Google Search: Generating inverted index files using a compression scheme
- Google Earth: Stitching overlapping satellite images to remove seams and to select high-quality imagery
- Google Maps: Processing all road segments on Earth and render map tile images that display segments
- More than 10,000 MR programs at Google in 4 years, run 100,000 MR jobs per day (2008)

MapReduce Popularity at Google

	Aug-04	Mar-06	Sep-07	Sep-09
Number of MapReduce jobs	29,000	171,000	2,217,000	3,467,000
Average completion time (secs)	634	874	395	475
Server years used	217	2,002	11,081	25,562
Input data read (TB)	3,288	52,254	403,152	544,130
Intermediate data (TB)	758	6,743	34,774	90,120
Output data written (TB)	193	2,970	14,018	57,520
Average number servers / job	157	268	394	488

What if Ran Google Workload on EC2?

	Aug-04	Mar-06	Sep-07	Sep-09
Number of MapReduce jobs	29,000	171,000	2,217,000	3,467,000
Average completion time (secs)	634	874	395	475
Server years used	217	2,002	11,081	25,562
Input data read (TB)	3,288	52,254	403,152	544,130
Intermediate data (TB)	758	6,743	34,774	90,120
Output data written (TB)	193	2,970	14,018	57,520
Average number of servers per job	157	268	394	488
Average Cost/job EC2	$17	$39	$26	$38
Annual Cost if on EC2	$0.5M	$6.7M	$57.4M	$133.1M

Question: Which statements are NOT TRUE about about MapReduce?

- ☐ Users express computation as two functions, Map and Reduce, and supply code for them
- ☐ MapReduce is well-matched to parallel processing of small data sets
- ☐ There are typically many more Map Tasks than Reduce Tasks (e.g., 40:1)
- ☐ MapReduce hides details of parallelization, fault tolerance, locality optimization, and load balancing

Question: Which statements are NOT TRUE about about MapReduce?

- ☐ Users express computation as two functions, Map and Reduce, and supply code for them
- ☐ MapReduce is well-matched to parallel processing of small data sets
- ☐ There are typically many more Map Tasks than Reduce Tasks (e.g., 40:1)
- ☐ MapReduce hides details of parallelization, fault tolerance, locality optimization, and load balancing

49

"And in Conclusion, …"

- 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
- Request-Level Parallelism
  - High request volume, each largely independent of other
  - Use replication for better request throughput, availability
- MapReduce Data Parallelism
  - Map: Divide large data set into pieces for independent parallel processing
  - Reduce: Combine and process intermediate results to obtain final result