Warehouse Scale Computers
Advanced Computer Architecture
COMP 140
Monday June 23, 2014

What is a Warehouse Scale Computer?

- Provides Internet services
  - Search, social networking, online maps, video sharing, online shopping, email, cloud computing, etc.
- Differences with HPC “clusters”:
  - Clusters have higher performance processors and network
  - Clusters emphasize thread-level parallelism, WSCs emphasize request-level parallelism
- Differences with datacenters:
  - Datacenters consolidate different machines and software into one location
  - Datacenters emphasize virtual machines and hardware heterogeneity in order to serve varied customers

Important Design Factors for WSC:

- Cost-performance
  - Small savings add up
- Energy efficiency
  - Affects power distribution and cooling
  - Work per joule
- Dependability via redundancy
- Network I/O
- Interactive and batch processing workloads
- Ample computational parallelism is not important
  - Most jobs are totally independent
  - “Request-level parallelism”
- Operational costs count
  - Power consumption is a primary, not secondary, constraint when designing system
- Scale and its opportunities and problems
  - Can afford to build customized systems since WSC require volume purchase

Are WSCs just clusters?

- “Clusters” are collections of independent computers that are connected together using standard LANs and off-the-shelf switches.
- In some sense, WSCs are just larger, more regular clusters — the regularity is necessary when you have tens of thousands of servers that must be maintained.

- However, WSCs are not as similar to today’s High Performance Computing (HPC) clusters.
  - Both cost a lot of money (millions of dollars) and might have millions of processors
  - HPC servers generally have much faster processors and much faster networks — HPC applications are generally much more interdependent and communicate more frequently.
  - HPC emphasizes thread-level or data-level parallelism, emphasizing latency for a single task rather than bandwidth for independent, request-level tasks.
  - Additionally, HPC clusters generally have long-running tasks that have high server utilization, whereas WSC utilization ranges between 10% and 50% (see the figure on the next slide).

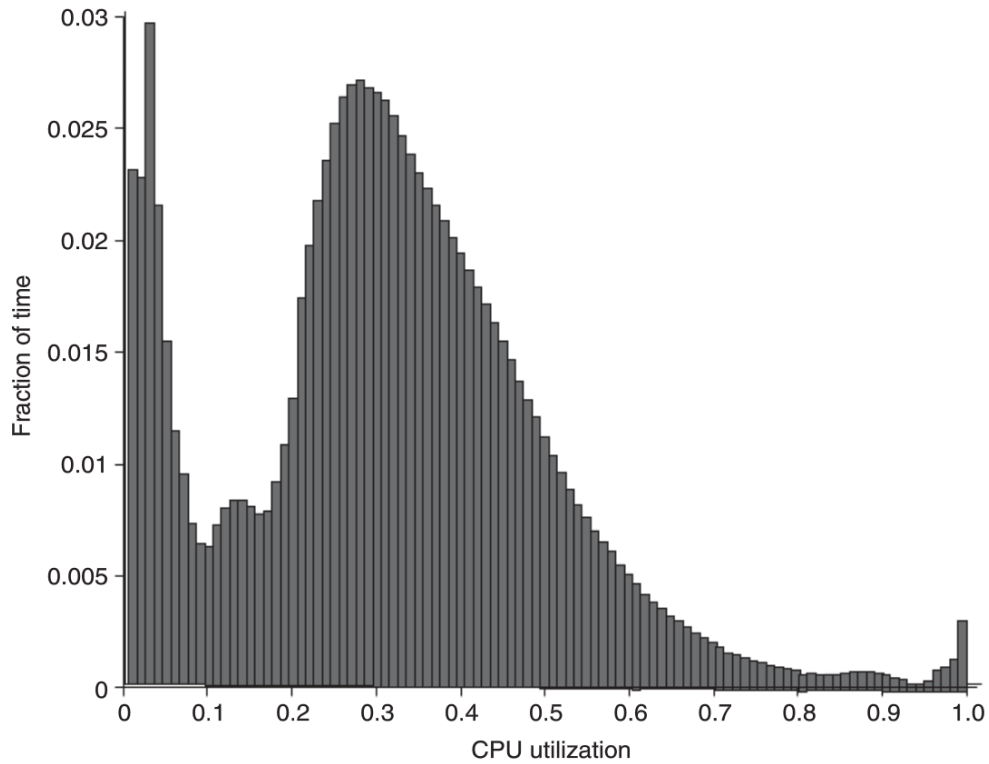


Figure 6.3 Average CPU utilization of more than 5000 servers during a 6-month period at Google. Servers are rarely completely idle or fully utilized, instead operating most of the time at between 10% and 50% of their maximum utilization. (From Figure 1 in Barroso and Hölzle [2007].) The column the third from the right in Figure 6.4 calculates percentages plus or minus 5% to come up with the weightings; thus, 1.2% for the 90% row means that 1.2% of servers were between 85% and 95% utilized.

Programming Models and Workloads

- Batch processing framework: MapReduce
  - developed at Google, has an open source implementation called “Hadoop”
- Basic idea: define two functions, **map**, and **reduce**.
  - **Map:** applies a programmer-supplied function to each logical input record
    - Runs on thousands of computers
    - Takes a value and outputs **key-value pairs** as intermediate values
  - **Reduce**: collapses values using another programmer-supplied function
- MapReduce is *highly parallel* and thus is ideal for WSC.

Map Reduce

- Example: map takes a string and outputs the length of the word as the key, and the word itself as the value (this may seem backwards):
  - map(steve) ➔ 5:steve
  - map(kaitlyn) ➔ 6:kaitlyn
- Map is *stateless* and only requires the input value to compute the output value.
- We can thus run map in parallel
- The reason the key-value pairs are formatted the way they are is because they are grouped together by keys:
- If the output was:

3: the	4 : when
3 : and	5 : steve
3 : you	5 : where
4 : then	8 : savannah
4 : what	8 : research

They would get grouped as:

3 : [the, and, you]
4 : [then, what, when]
5 : [steve, where]
8 : [savannah, research]

Map Reduce

- The reason the key-value pairs are formatted the way they are is because they are grouped together by keys:

- If the output was:

3: the	4 : when
3 : and	5 : steve
3 : you	5 : where
4 : then	8 : savannah
4 : what	8 : research

They would get grouped as:

3 : [the, and, you]
4 : [then, what, when]
5 : [steve, where]
8 : [savannah, research]

Each of these lines would then be passed as an argument to the reduce function, which accepts a key and a list of values. In this instance, we might be trying to figure out how many words of certain lengths exist, so our reduce function will just count the number of items in the list and output the key with the size of the list, like:

3 : 3
4 : 3
5 : 2
8 : 2

The reductions can be done in parallel as well

Map Reduce

- The most common example of MapReduce is to count the number of times a word appears in a corpus. Suppose you were Google and had a copy of the entire Internet. You want a list of every word on the Internet as well as how many times it occurs.

- You “tokenize” the document into words, and pass each word to a mapper. The mapper spits back the word with a value of “1”.
- The grouping phase takes all of the keys (the words), and make a list of 1s.
- The reduce phase takes a key (a word) and a list (a list of 1s for each time the key appeared on the Internet), and sums the list.
- The reducer outputs a word, along with its count.
- You now have a list of every word on the Internet, along with its count

Map Reduce: a bigger example — FriendFace

FriendFace has a list of friends that are bi-directional (if you’re my friend, I’m your friend).

FriendFace also has lots of disk space, and serve hundreds of millions of requests per day. They want to reduce the amount of real-time computation to reduce the processing time of a request.

One common request is the “you and Joe have 230 friends in common.” When you visit someone's profile, you see a list of friends that you have in common. This list doesn't change frequently so it'd be wasteful to recalculate it every time you visited the profile.

We're going to use mapreduce so that we can calculate everyone's common friends once a day and store those results. Later on it's just a quick lookup. We've got lots of disk, it's cheap.

Assume the friends are stored as Person➔[List of Friends], our friends list is then:

A ➔ B C D
B ➔ A C D E
C ➔ A B D E
D ➔ A B C E
E ➔ B C D

Each line will be an argument to a mapper. For every friend in the list of friends, the mapper will output a key-value pair. The key will be a friend along with the person. The value will be the list of friends. The key will be sorted so that the friends are in order, causing all pairs of friends to go to the same reducer.

Assume the friends are stored as Person➔[List of Friends], our friends list is then:

A ➔ B C D
B ➔ A C D E
C ➔ A B D E
D ➔ A B C E
E ➔ B C D

Each line will be an argument to a mapper. For every friend in the list of friends, the mapper will output a key-value pair. The key will be a friend along with the person. The value will be the list of friends. The key will be sorted so that the friends are in order, causing all pairs of friends to go to the same reducer.

Assume the friends are stored as Person➔[List of Friends], our friends list is then:

A ➔ B C D
B ➔ A C D E
C ➔ A B D E
D ➔ A B C E
E ➔ B C D

After all the mappers are done running, you'll have a list like this:

For map(A ➔ B C D) :
(A B) ➔ B C D
(A C) ➔ B C D
(A D) ➔ B C D

For map(B ➔ A C D E) : (Note that A comes before B in the key)
(A B) ➔ A C D E
(B C) ➔ A C D E
(B D) ➔ A C D E
(B E) ➔ A C D E

For map(C ➔ A B D E) :
(A C) ➔ A B D E
(B C) ➔ A B D E
(C D) ➔ A B D E
(C E) ➔ A B D E

For map(D ➔ A B C E) :
(A D) ➔ A B C E
(B D) ➔ A B C E
(C D) ➔ A B C E
(D E) ➔ A B C E

And finally for map(E ➔ B C D):
(B E) ➔ B C D
(C E) ➔ B C D
(D E) ➔ B C D

Before we send these key-value pairs to the reducers, we group them by their keys and get:

(A B) ➔ (A C D E) (B C D)
(A C) ➔ (A B D E) (B C D)
(A D) ➔ (A B C E) (B C D)
(B C) ➔ (A B D E) (A C D E)
(B D) ➔ (A B C E) (A C D E)
(B E) ➔ (A C D E) (B C D)
(C D) ➔ (A B C E) (A B D E)
(C E) ➔ (A B D E) (B C D)
(D E) ➔ (A B C E) (B C D)

Each line will be passed as an argument to a reducer. The reduce function will simply intersect the lists of values and output the same key with the result of the intersection. For example, reduce((A B) ➔ (A C D E) (B C D)) will output (A B) : (C D) and means that friends A and B have C and D as common friends.

The result after reduction is:
(A B) ➔ (C D)
(A C) ➔ (B D)
(A D) ➔ (B C)
(B C) ➔ (A D E)
(B D) ➔ (A C E)
(B E) ➔ (C D)
(C D) ➔ (A B E)
(C E) ➔ (B D)
(D E) ➔ (B C)

Now when D visits B's profile, we can quickly look up (B D) and see that they have three friends in common, (A C E).

Programming Models and Workloads

- MapReduce runtime environment schedules map and reduce task to WSC nodes

- Availability:
  - Use replicas of data across different servers
  - Use relaxed consistency:
    - No need for all replicas to always agree

- Workload demands
  - Often vary considerably

- Variability of performance:
  - Servers don't have to be identical — even slow servers will complete some portion of the work
  - Near the end of a MapReduce job, the system will start backup executions of incomplete tasks on free nodes, and take the result from whichever node finishes first. A little increase of resource usage can make some tasks complete 30% faster.

Computer Architecture of WSC

- WSC often use a hierarchy of networks for interconnection (much like the memory hierarchy inside a desktop computer)
- Each 19” rack holds 48 1U servers connected to a rack switch
- Rack switches are uplinked to switch higher in hierarchy
  - Uplink has 48 / n times lower bandwidth, where n = # of uplink ports
  - “Oversubscription”
- Goal is to maximize locality of communication relative to the rack

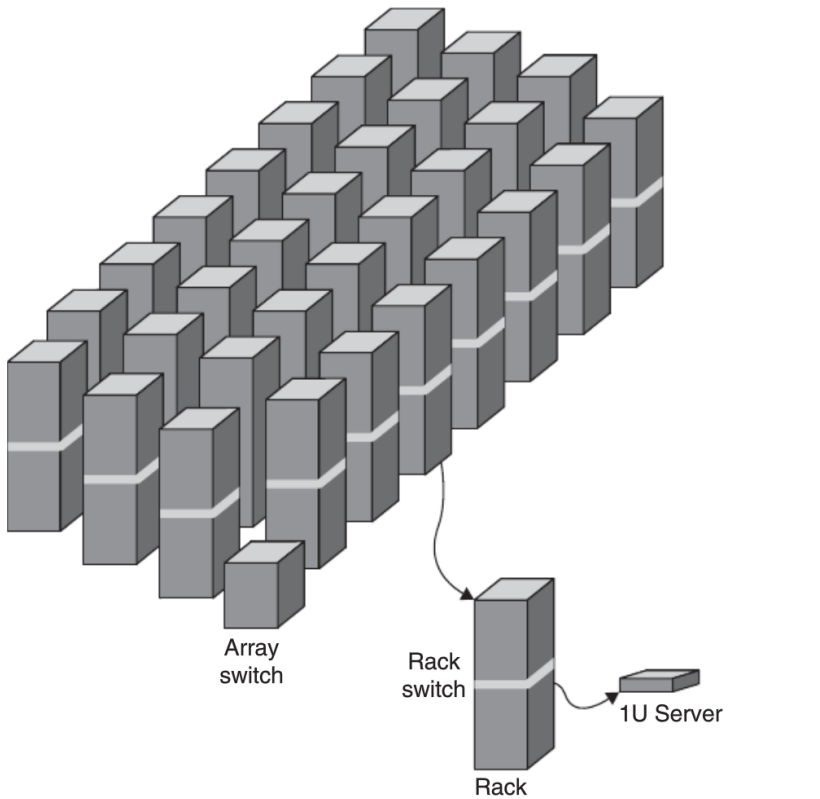
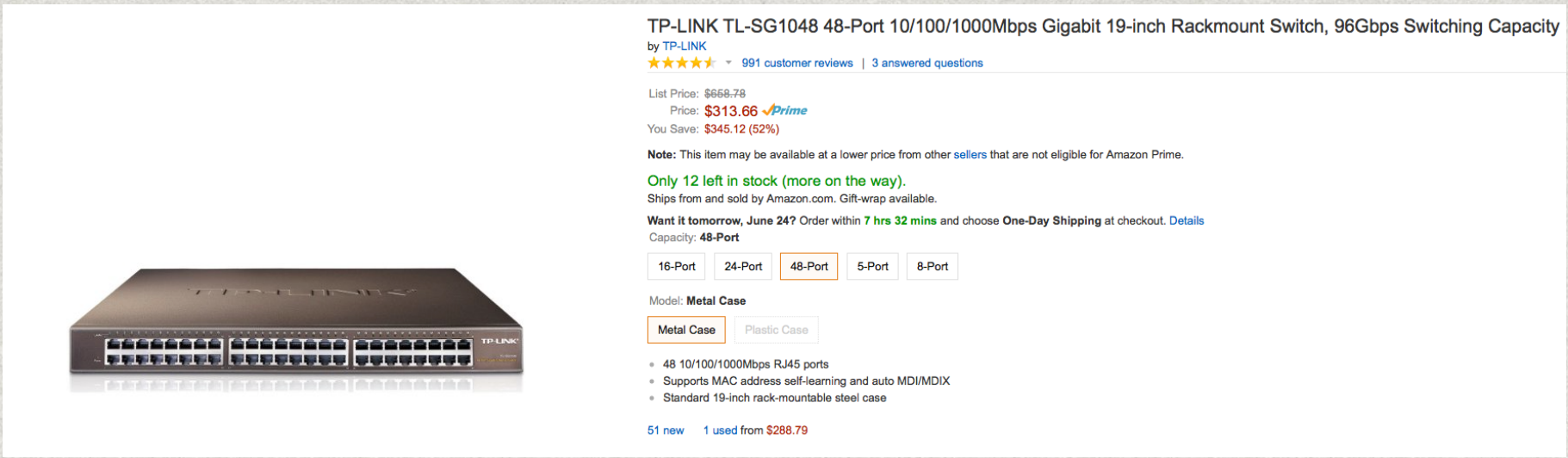


Figure 6.5 Hierarchy of switches in a WSC. (Based on Figure 1.2 of Barroso and Hölzle [2009].)

Rack Switch

- Generally a 48-port switch, one per rack
- Commodity — costs have dropped



- The problem is “oversubscription” — there are only 2-8 uplinks, meaning that the bandwidth leaving the rack is 6-24 times smaller!

Array Switch

- Switch that connects an array of racks
- Array switch should have 10 X the bisection bandwidth of rack switch
- Cost of n-port switch grows as n^2 ($700K per switch!)
- Often utilize content addressable memory chips and FPGAs

WSC Memory Hierarchy

- Servers can access DRAM and disks on other servers using a NUMA-style interface

	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	1040	31,200
Disk capacity (GB)	2000	160,000	4,800,000

Figure 6.6 Latency, bandwidth, and capacity of the memory hierarchy of a WSC [Barroso and Hölzle 2009]. Figure 6.7 plots this same information.

Storage

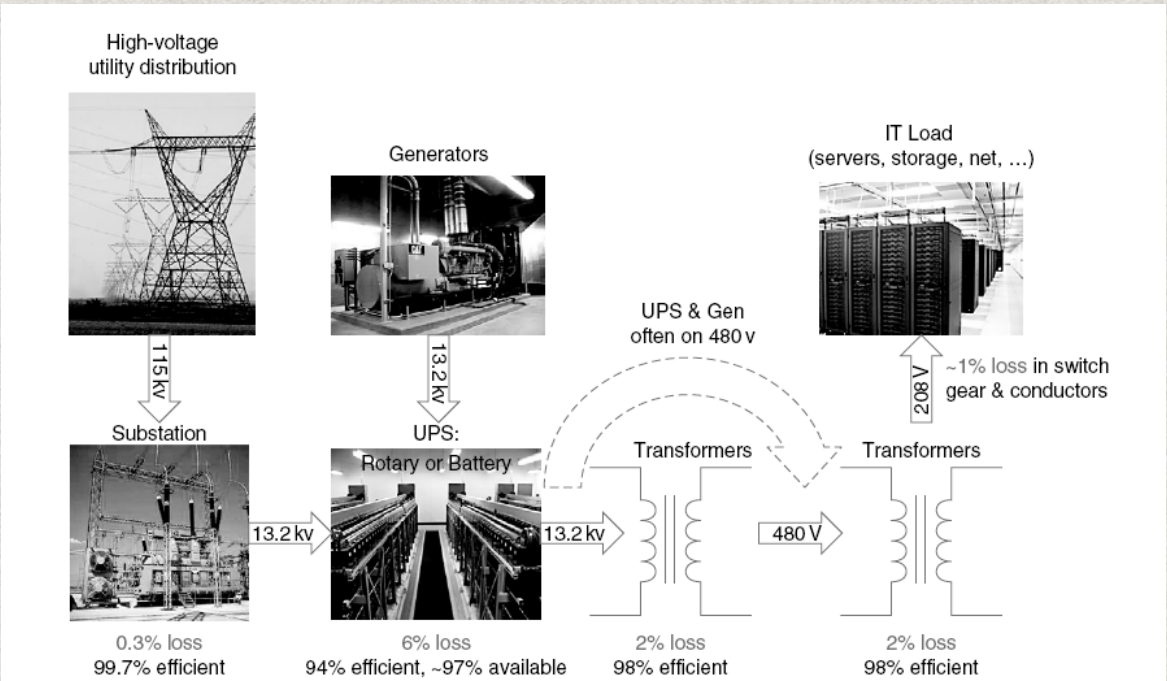
- Use disks inside the servers, or
- Network attached storage through Infiniband
- WSCs generally rely on local disks
- Google File System (GFS) uses local disks and maintains at least three replicas

WSC Storage

- Warehouse scale storage often makes different decisions than single servers.
- E.g., A single server might have multiple disks that share information in a RAID. The storage system itself is considered reliable, because the RAID machinery handles corruption.
- A WSC, on the other hand, might simply make complete replicas of data onto multiple disks in different servers. There will be cross-server redundancy rather than within-a-server redundancy.
  - A failure of an entire server (which happens regularly with thousands of servers) does not affect the availability of the data.
- WSC can also implement “relaxed consistency” across servers, for certain data.
- Relaxed consistency means that two servers may disagree on the data, but will exhibit “eventual consistency” and will eventually be consistent.
  - E.g., if you upload a video to YouTube, it needs to be (1) transcoded to a format that YouTube accepts, and (2) distributed across servers so it can be efficiently served when requested.
  - YouTube does multiple passes over the video, and therefore different servers end up with different versions. A requester might get a new copy or an old copy, but eventually all the copies will be new.

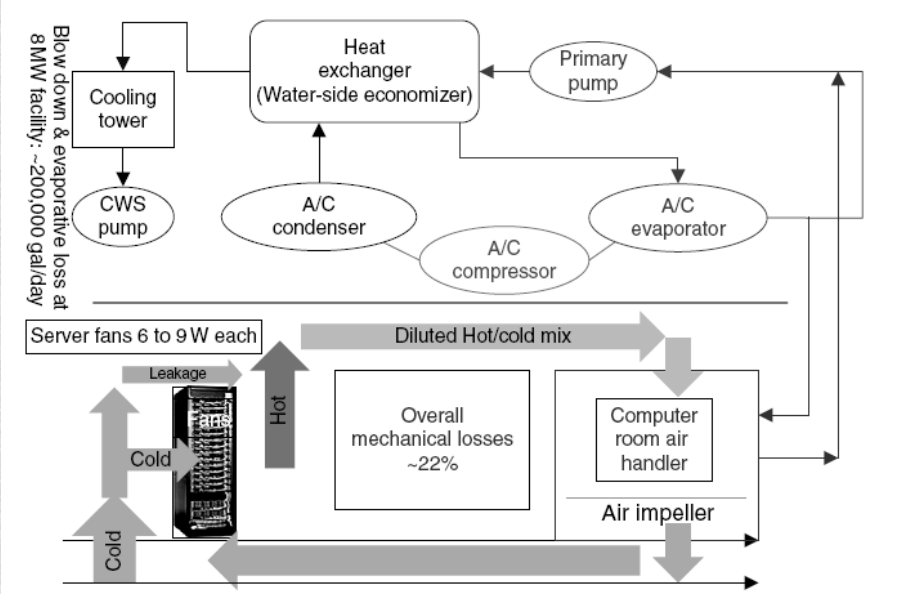
Infrastructure and Costs of WSC

- Location of WSC
  - Proximity to Internet backbones, electricity cost, property tax rates, low risk from earthquakes, floods, and hurricanes
- Power distribution:



Infrastructure and Costs of WSC

- Cooling
  - Air conditioning used to cool server room
  - 64 F – 71 F
    - Keep temperature higher (closer to 71 F, though Google keeps theirs even higher)
  - Cooling towers can also be used
    - Minimum temperature is “wet bulb temperature”



Infrastructure and Costs of WSC

- Cooling system also uses water (evaporation and spills)
  - E.g. 70,000 to 200,000 gallons per day for an 8 MW facility

- Power cost breakdown:
  - Chillers: 30-50% of the power used by the IT equipment
  - Air conditioning: 10-20% of the IT power, mostly due to fans

- How man servers can a WSC support?
  - Each server:
    - “Nameplate power rating” gives maximum power consumption
    - To get actual, measure power under actual workloads
- Oversubscribe cumulative server power by 40%, but monitor power closely

Measuring the Efficiency of a WSC

- We use a simple metric, called the “Power Utilization Effectiveness” (PUE):
  - PUE = (Total Facility power) / (IT equipment power)
- PUE must be greater than 1, and the bigger the PUE, the less efficient the WSC
- PUE goals are 1.3 or less.

- Performance
  - Latency is important metric because it is seen by users
  - Bing study: users will use search less as response time increases
  - Service Level Objectives (SLOs)/Service Level Agreements (SLAs)
  - E.g. 99% of requests be below 100 ms

Server delay (ms)	Increased time to next click (ms)	Queries/ user	Any clicks/ user	User satisfaction	Revenue/ user
50	--	--	--	--	--
200	500	--	–0.3%	–0.4%	--
500	1200	--	–1.0%	–0.9%	–1.2%
1000	1900	–0.7%	–1.9%	–1.6%	–2.8%
2000	3100	–1.8%	–4.4%	–3.8%	–4.3%

Figure 6.12 Negative impact of delays at Bing search server on user behavior Schurman and Brutlag [2009].

Cost of a WSC

- Capital expenditures (CAPEX)
  - Cost to build a WSC

Size of facility (critical load watts)	8,000,000
Average power usage (%)	80%
Power usage effectiveness	1.45
Cost of power ($/kwh)	$0.07
% Power and cooling infrastructure (% of total facility cost)	82%
CAPEX for facility (not including IT equipment)	**$88,000,000**
Number of servers	45,978
Cost/server	$1450
CAPEX for servers	**$66,700,000**
Number of rack switches	1150
Cost/rack switch	$4800
Number of array switches	22
Cost/array switch	$300,000
Number of layer 3 switches	2
Cost/layer 3 switch	$500,000
Number of border routers	2
Cost/border router	$144,800
CAPEX for networking gear	**$12,810,000**
Total CAPEX for WSC	**$167,510,000**
Server amortization time	3 years
Networking amortization time	4 years
Facilities amortization time	10 years
Annual cost of money	5%

Figure 6.13 Case study for a WSC, based on Hamilton [2010], rounded to nearest $5000. Internet bandwidth costs vary by application, so they are not included here. The remaining 18% of the CAPEX for the facility includes buying the property and the cost of construction of the building. We added people costs for security and facilities management in Figure 6.14, which were not part of the case study. Note that Hamilton's estimates were done before he joined Amazon, and they are not based on the WSC of a particular company.

Cost of a WSC

- Operational expenditures (OPEX)
  - Cost to operate a WSC

Expense (% total)	Category	Monthly cost	Percent monthly cost
Amortized CAPEX (85%)	Servers	$2,000,000	53%
	Networking equipment	$290,000	8%
	Power and cooling infrastructure	$765,000	20%
	Other infrastructure	$170,000	4%
OPEX (15%)	Monthly power use	$475,000	13%
	Monthly people salaries and benefits	$85,000	2%
	Total OPEX	$3,800,000	100%

Figure 6.14 Monthly OPEX for Figure 6.13, rounded to the nearest $5000. Note that the 3-year amortization for servers means you need to purchase new servers every 3 years, whereas the facility is amortized for 10 years. Hence, the amortized capital costs for servers are about 3 times more than for the facility. People costs include 3 security guard positions continuously for 24 hours a day, 365 days a year, at $20 per hour per person, and 1 facilities person for 24 hours a day, 365 days a year, at $30 per hour. Benefits are 30% of salaries. This calculation doesn't include the cost of network bandwidth to the Internet, as it varies by application, nor vendor maintenance fees, as that varies by equipment and by negotiations.

Cloud Computing

- WSCs offer economies of scale that cannot be achieved with a datacenter:
  - 5.7 times reduction in storage costs
  - 7.1 times reduction in administrative costs
  - 7.3 times reduction in networking costs
  - This has given rise to cloud services such as Amazon Web Services
    - “Utility Computing”
    - Based on using open source virtual machine and operating system software

Amazon Web Services (A true “utility computing” infrastructure)

- If you have a credit card, you can purchase compute time on AWS.
- Big-time users: Netflix (!), “Farmville”
- Amazon made some interesting business decisions:
  - Virtual Machines — a user does not know the underlying hardware, and cannot affect other users. Each user sets up a VM, and it is distributed across the WSC. Additionally, VMs can limit hardware use, allowing for tiered pricing.
  - Low cost: initially (2006), AWS cost $0.10 per hour per instance, which was startlingly low. Today, you can get started for free.
  - Open Source Software — Linux, other open source tools means that Amazon doesn’t have to license the software
  - No guarantee of service (initially) — originally, AWS only offered “best effort” — you might not get the service if it went down. But, Amazon S3 was designed for 99.999999999% durability — the chance of permanently losing your data is 1 in 100 *billion*.
  - No contract — sign up and quit at your leisure.

Amazon Web Services — On-demand services

Instance	Per hour	Ratio to small	Compute units	Virtual cores	Compute units/core	Memory (GB)	Disk (GB)	Address size
Micro	$0.020	0.5–2.0	0.5–2.0	1	0.5–2.0	0.6	EBS	32/64 bit
Standard Small	$0.085	1.0	1.0	1	1.00	1.7	160	32 bit
Standard Large	$0.340	4.0	4.0	2	2.00	7.5	850	64 bit
Standard Extra Large	$0.680	8.0	8.0	4	2.00	15.0	1690	64 bit
High-Memory Extra Large	$0.500	5.9	6.5	2	3.25	17.1	420	64 bit
High-Memory Double Extra Large	$1.000	11.8	13.0	4	3.25	34.2	850	64 bit
High-Memory Quadruple Extra Large	$2.000	23.5	26.0	8	3.25	68.4	1690	64 bit
High-CPU Medium	$0.170	2.0	5.0	2	2.50	1.7	350	32 bit
High-CPU Extra Large	$0.680	8.0	20.0	8	2.50	7.0	1690	64 bit
Cluster Quadruple Extra Large	$1.600	18.8	33.5	8	4.20	23.0	1690	64 bit

Compute Provisioning Risks

- If you provide a service but don't know how much computing you'll need, you risk either *over-provisioning* or *under-provisioning*. Start-up companies can be hurt by both of these, but cloud computing providers (e.g., Amazon Web Services) greatly mitigate the risks.
- Example: Zynga and FarmVille
  - Before FarmVille was announced, the largest online social game had about 5 million daily players.
  - FarmVille had 1 million players four days after launch, and 10 million after 60 days.
  - After 270 days, it had 28 million daily players, and 75 million monthly players.
  - They would never have been able to keep up with the demand had they not been using a cloud service (they were using AWS). AWS was able to grow with the number of users (and shrink when FarmVille lost popularity).

- Netflix began streaming video in 2007, using their own datacenters. However, as they grew, they realized that migration to AWS made more sense.

A Google Warehouse-Scale Computer

- Google has shared information about their datacenters up to 2007 (not new, but at this scale *everything* is a company secret).
- WSCs are now built with shipping containers (they used to be standard shipping containers, but now they are generally customized).
- Shipping containers are modular — the only external connections are networking, power, and cooling.
- The 2007 version contains 45 40-foot long containers in a 300-foot by 250-foot space, or 75,000sq-ft. To fit in the warehouse, 30 of the containers are stacked two-high.
- The data center is probably in Dalles, Oregon, which has a moderate climate near cheap hydroelectric power and an Internet backbone fiber location (Google has an energy subsidiary that can buy and sell energy like any utility).
- The WSC has a PUE of 1.23.
  - 85% of the energy overhead goes to cooling losses, 15% goes to power losses.

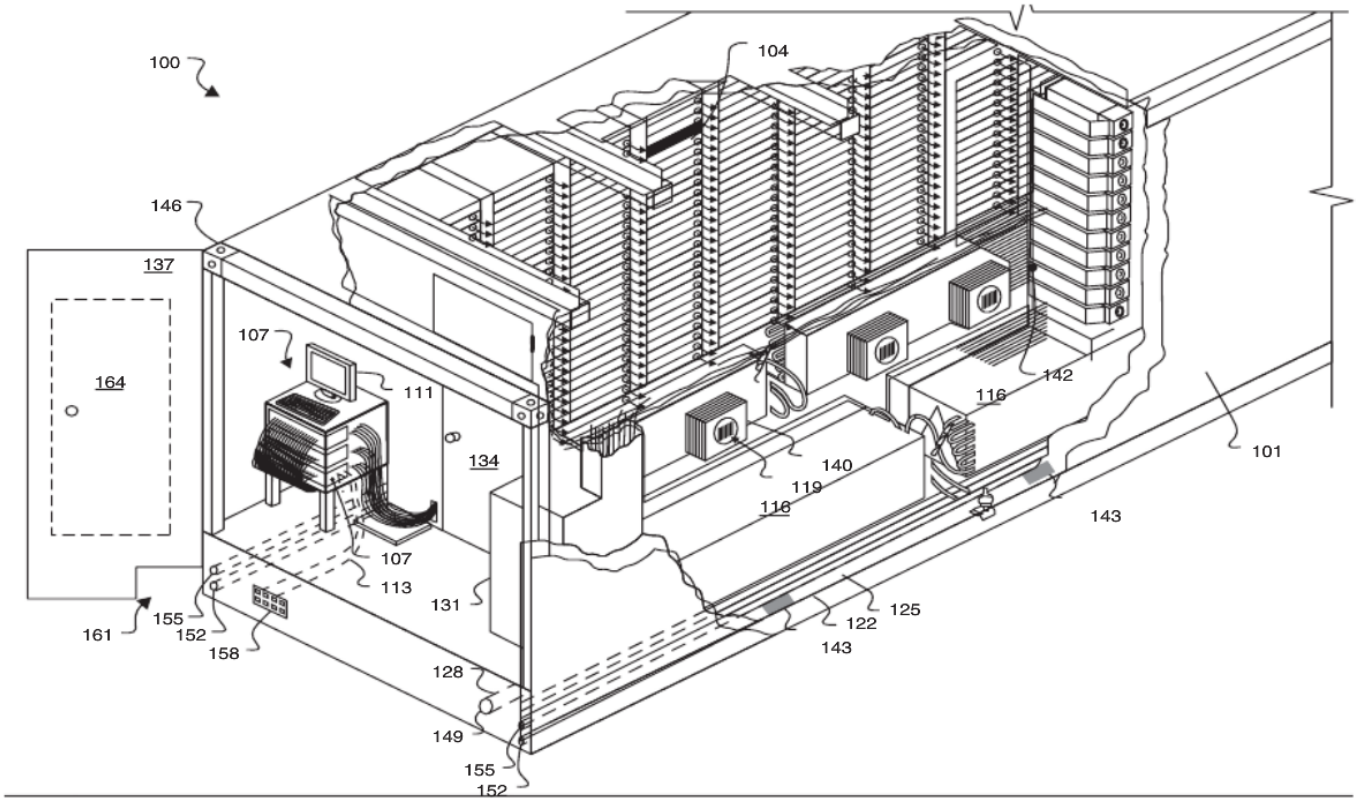


Figure 6.19 Google customizes a standard 1AAA container: 40 x 8 x 9.5 feet (12.2 x 2.4 x 2.9 meters). The servers are stacked up to 20 high in racks that form two long rows of 29 racks each, with one row on each side of the container. The cool aisle goes down the middle of the container, with the hot air return being on the outside. The hanging rack structure makes it easier to repair the cooling system without removing the servers. To allow people inside the container to repair components, it contains safety systems for fire detection and mist-based suppression, emergency egress and lighting, and emergency power shut-off. Containers also have many sensors: temperature, airflow pressure, air leak detection, and motion-sensing lighting. A video tour of the datacenter can be found at *http://www.google.com/corporate/green/datacenters/summit.html*. Microsoft, Yahoo!, and many others are now building modular datacenters based upon these ideas but they have stopped using ISO standard containers since the size is inconvenient.

- Each container holds up to 1160 servers
- 45 containers have space for 52,200 servers
- The servers are stacked 20 high in racks that form two long rows of 29 racks.
- The switches are 48-port, 1Gbit/sec Ethernet switches, placed in every other rack
- https://www.youtube.com/watch?v=M-3bdWCOSMQ#t=21

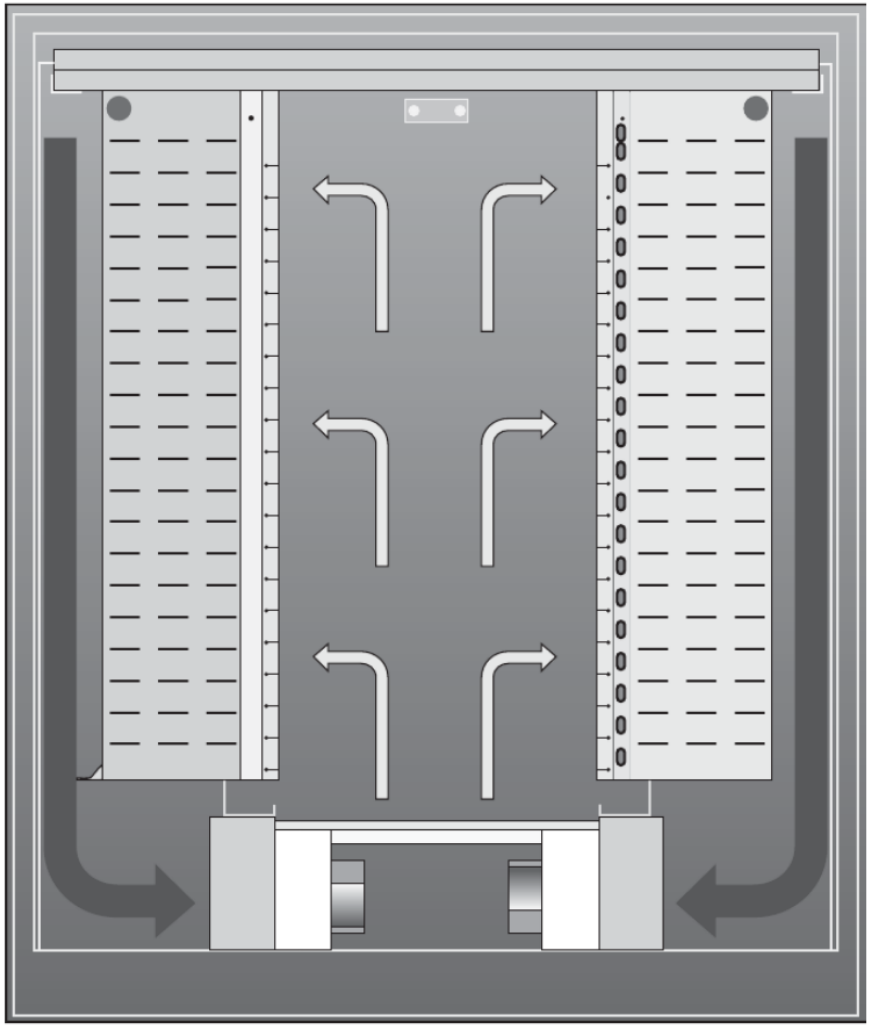
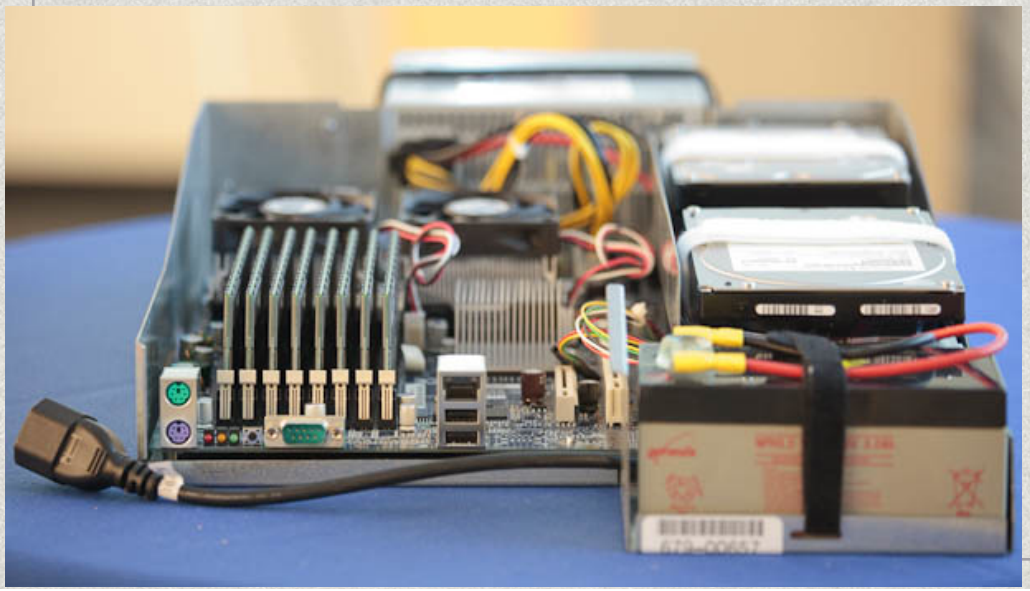


Figure 6.20 Airflow within the container shown in Figure 6.19. This cross-section diagram shows two racks on each side of the container. Cold air blows into the aisle in the middle of the container and is then sucked into the servers. Warm air returns at the edges of the container. This design isolates cold and warm airflows.

Cooling

- Cooling is below a raised floor that blows into the aisle between the racks
- Hot air is returned from behind the racks
- Variable speed fans run at the lowest speeds necessary to cool the racks.
- The cold air is kept at 81ºF (hot for many data centers, though that is changing — servers are now allowed to get up to 115ºF)
  - "hot spots" are avoided by design
- External chillers can simply use the ambient temperature outside (from external cooling towers).
- The cooling towers do need heaters if it gets too cold (to avoid ice).

Cooling Towers (and deer)



A Google Server

- Google skips the 220V conversion and supplies 12V to the motherboard.
- UPS is simply a battery (bottom right corner of the bottom picture)
- One or two hard drives per server
- 8GB memory
- No extra sheet metal!

- Extensive automatic monitoring is done to track the health of the servers.
  - Diagnostics run continually
  - Automated solutions solve many problems (server freeze➔reboot, etc.)
  - Repair technicians use the automated tests for quick repair (e.g., they don't diagnose if the automatic test says a part is bad).
  - Goal: less than 1% in the repair queue at any time.

References

- MapReduce explanation and example: http://stevekrenzel.com/finding-friends-with-mapreduce
- Google Data Center information: http://www.cnet.com/news/google-uncloaks-once-secret-server-10209580/
- More information about Google's Data Centers: http://www.wired.com/2012/10/ff-inside-google-data-center/all/

- Google Data Center videos: https://www.youtube.com/watch?v=M-3bdWCOSMQ#t=21 https://www.youtube.com/watch?v=L-NJupECSJQ https://www.youtube.com/watch?v=zRwPSFpLX8I
- Amazon Web Services: https://aws.amazon.com