

# 6

## Warehouse-Scale Computers to Exploit Request-Level and Data-Level Parallelism

The datacenter is the computer.

**Luiz André Barroso,**  
*Google (2007)*

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.

**Nicholas Carr**  
*The Big Switch: Rewiring the World, from Edison to Google (2008)*

## 6.1 Introduction

Anyone can build a fast CPU. The trick is to build a fast system.

**Seymour Cray**

*Considered the father of the supercomputer*

The warehouse-scale computer (WSC)<sup>1</sup> is the foundation of Internet services many people use every day: search, social networking, online maps, video sharing, online shopping, email services, and so on. The tremendous popularity of such Internet services necessitated the creation of WSCs that could keep up with the rapid demands of the public. Although WSCs may appear to be just large datacenters, their architecture and operation are quite different, as we shall see. Today's WSCs act as one giant machine and cost on the order of \$150M for the building, the electrical and cooling infrastructure, the servers, and the networking equipment that connects and houses 50,000 to 100,000 servers. Moreover, the rapid growth of cloud computing (see [Section 6.5](#)) makes WSCs available to anyone with a credit card.

Computer architecture extends naturally to designing WSCs. For example, Luiz Barroso of Google (quoted earlier) did his dissertation research in computer architecture. He believes an architect's skills of designing for scale, designing for dependability, and a knack for debugging hardware are very helpful in the creation and operation of WSCs.

At this extreme scale, which requires innovation in power distribution, cooling, monitoring, and operations, the WSC is the modern descendant of the supercomputer—making Seymour Cray the godfather of today's WSC architects. His extreme computers handled computations that could be done nowhere else, but were so expensive that only a few companies could afford them. This time the target is providing information technology for the world instead of high-performance computing (HPC) for scientists and engineers; hence, WSCs arguably play a more important role for society today than Cray's supercomputers did in the past.

Unquestionably, WSCs have many orders of magnitude more users than high-performance computing, and they represent a much larger share of the IT market. Whether measured by number of users or revenue, Google is at least 250 times larger than Cray Research ever was.

---

<sup>1</sup> This chapter is based on material from the book *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*, by Luiz André Barroso and Urs Hölzle of Google [2009]; the blog *Perspectives at mvdirona.com* and the talks “Cloud-Computing Economies of Scale” and “Data Center Networks Are in My Way,” by James Hamilton of Amazon Web Services [2009, 2010]; and the technical report *Above the Clouds: A Berkeley View of Cloud Computing*, by Michael Armbrust et al. [2009].

WSC architects share many goals and requirements with server architects:

- *Cost-performance*—Work done per dollar is critical in part because of the scale. Reducing the capital cost of a WSC by 10% could save \$15M.
- *Energy efficiency*—Power distribution costs are functionally related to power consumption; you need sufficient power distribution before you can consume power. Mechanical system costs are functionally related to power: You need to get out the heat that you put in. Hence, peak power and consumed power drive both the cost of power distribution and the cost of cooling systems. Moreover, energy efficiency is an important part of environmental stewardship. Hence, work done per joule is critical for both WSCs and servers because of the high cost of building the power and mechanical infrastructure for a warehouse of computers and for the monthly utility bills to power servers.
- *Dependability via redundancy*—The long-running nature of Internet services means that the hardware and software in a WSC must collectively provide at least 99.99% of availability; that is, it must be down less than 1 hour per year. Redundancy is the key to dependability for both WSCs and servers. While server architects often utilize more hardware offered at higher costs to reach high availability, WSC architects rely instead on multiple cost-effective servers connected by a low-cost network and redundancy managed by software. Furthermore, if the goal is to go much beyond “four nines” of availability, you need multiple WSCs to mask events that can take out whole WSCs. Multiple WSCs also reduce latency for services that are widely deployed.
- *Network I/O*—Server architects must provide a good network interface to the external world, and WSC architects must also. Networking is needed to keep data consistent between multiple WSCs as well as to interface to the public.
- *Both interactive and batch processing workloads*—While you expect highly interactive workloads for services like search and social networking with millions of users, WSCs, like servers, also run massively parallel batch programs to calculate metadata useful to such services. For example, MapReduce jobs are run to convert the pages returned from crawling the Web into search indices (see [Section 6.2](#)).

Not surprisingly, there are also characteristics *not* shared with server architecture:

- *Ample parallelism*—A concern for a server architect is whether the applications in the targeted marketplace have enough parallelism to justify the amount of parallel hardware and whether the cost is too high for sufficient communication hardware to exploit this parallelism. A WSC architect has no such concern. First, batch applications benefit from the large number of independent datasets that require independent processing, such as billions of Web pages from a Web crawl. This processing is *data-level parallelism* applied to data in storage instead of data in memory, which we saw in [Chapter 4](#). Second, interactive Internet service applications, also known as *software as a service* (*SaaS*), can benefit from millions of independent users of interactive Internet

services. Reads and writes are rarely dependent in SaaS, so SaaS rarely needs to synchronize. For example, search uses a read-only index and email is normally reading- and writing-independent information. We call this type of easy parallelism *request-level parallelism*, as many independent efforts can proceed in parallel naturally with little need for communication or synchronization; for example, journal-based updating can reduce throughput demands. Given the success of SaaS and WSCs, more traditional applications such as relational databases have been weakened to rely on request-level parallelism. Even read-/write-dependent features are sometimes dropped to offer storage that can scale to the size of modern WSCs.

- *Operational costs count*—Server architects usually design their systems for peak performance within a cost budget and worry about power only to make sure they don't exceed the cooling capacity of their enclosure. They usually ignore operational costs of a server, assuming that they pale in comparison to purchase costs. WSCs have longer lifetimes—the building and electrical and cooling infrastructure are often amortized over 10 or more years—so the operational costs add up: Energy, power distribution, and cooling represent more than 30% of the costs of a WSC in 10 years.
- *Scale and the opportunities/problems associated with scale*—Often extreme computers are extremely expensive because they require custom hardware, and yet the cost of customization cannot be effectively amortized since few extreme computers are made. However, when you purchase 50,000 servers and the infrastructure that goes with it to construct a single WSC, you *do* get volume discounts. WSCs are so massive internally that you get economy of scale even if there are not many WSCs. As we shall see in [Sections 6.5](#) and [6.10](#), these economies of scale led to cloud computing, as the lower per-unit costs of a WSC meant that companies could rent them at a profit below what it costs outsiders to do it themselves. The flip side of 50,000 servers is failures. [Figure 6.1](#) shows outages and anomalies for 2400 servers. Even if a server had a mean time to failure (MTTF) of an amazing 25 years (200,000 hours), the WSC architect would need to design for 5 server failures a day. [Figure 6.1](#) lists the annualized disk failure rate as 2% to 10%. If there were 4 disks per server and their annual failure rate was 4%, with 50,000 servers the WSC architect should expect to see one disk fail per *hour*:

---

**Example** Calculate the availability of a service running on the 2400 servers in [Figure 6.1](#). Unlike a service in a real WSC, in this example the service cannot tolerate hardware or software failures. Assume that the time to reboot software is 5 minutes and the time to repair hardware is 1 hour.

**Answer** We can estimate service availability by calculating the time of outages due to failures of each component. We'll conservatively take the lowest number in each category in [Figure 6.1](#) and split the 1000 outages evenly between four components. We ignore slow disks—the fifth component of the 1000 outages—since

Approx. number events in 1st year	Cause	Consequence
1 or 2	Power utility failures	Lose power to whole WSC; doesn't bring down WSC if UPS and generators work (generators work about 99% of time).
4	Cluster upgrades	Planned outage to upgrade infrastructure, many times for evolving networking needs such as recabling, to switch firmware upgrades, and so on. There are about 9 planned cluster outages for every unplanned outage.
1000s	Hard-drive failures	2% to 10% annual disk failure rate [Pinheiro 2007]
	Slow disks	Still operate, but run 10x to 20x more slowly
	Bad memories	One uncorrectable DRAM error per year [Schroeder et al. 2009]
	Misconfigured machines	Configuration led to ~30% of service disruptions [Barroso and Hölzle 2009]
	Flaky machines	1% of servers reboot more than once a week [Barroso and Hölzle 2009]
5000	Individual server crashes	Machine reboot, usually takes about 5 minutes

**Figure 6.1** List of outages and anomalies with the approximate frequencies of occurrences in the first year of a new cluster of 2400 servers. We label what Google calls a cluster an *array*; see Figure 6.5. (Based on Barroso [2010].)

they hurt performance but not availability, and power utility failures, since the uninterruptible power supply (UPS) system hides 99% of them.

$$\begin{aligned} \text{Hours Outage}_{\text{service}} &= (4 + 250 + 250 + 250) \times 1 \text{ hour} + (250 + 5000) \times 5 \text{ minutes} \\ &= 754 + 438 = 1192 \text{ hours} \end{aligned}$$

Since there are  $365 \times 24$  or 8760 hours in a year, availability is:

$$\text{Availability}_{\text{system}} = \frac{(8760 - 1192)}{8760} = \frac{7568}{8760} = 86\%$$

That is, without software redundancy to mask the many outages, a service on those 2400 servers would be down on average one day a week, or *zero* nines of availability!

As Section 6.10 explains, the forerunners of WSCs are *computer clusters*. Clusters are collections of independent computers that are connected together using standard local area networks (LANs) and off-the-shelf switches. For workloads that did not require intensive communication, clusters offered much more cost-effective computing than shared memory multiprocessors. (Shared memory multiprocessors were the forerunners of the multicore computers discussed in Chapter 5.) Clusters became popular in the late 1990s for scientific computing and then later for Internet services. One view of WSCs is that they are just the logical evolution from clusters of hundreds of servers to tens of thousands of servers today.

A natural question is whether WSCs are similar to modern clusters for high-performance computing. Although some have similar scale and cost—there are HPC designs with a million processors that cost hundreds of millions of dollars—they generally have much faster processors and much faster networks between the nodes than are found in WSCs because the HPC applications are more interdependent and communicate more frequently (see [Section 6.3](#)). HPC designs also tend to use custom hardware—especially in the network—so they often don’t get the cost benefits from using commodity chips. For example, the IBM Power 7 microprocessor alone can cost more and use more power than an entire server node in a Google WSC. The programming environment also emphasizes thread-level parallelism or data-level parallelism (see [Chapters 4 and 5](#)), typically emphasizing latency to complete a single task as opposed to bandwidth to complete many independent tasks via request-level parallelism. The HPC clusters also tend to have long-running jobs that keep the servers fully utilized, even for weeks at a time, while the utilization of servers in WSCs ranges between 10% and 50% (see [Figure 6.3](#) on page 440) and varies every day.

How do WSCs compare to conventional datacenters? The operators of a conventional datacenter generally collect machines and third-party software from many parts of an organization and run them centrally for others. Their main focus tends to be consolidation of the many services onto fewer machines, which are isolated from each other to protect sensitive information. Hence, virtual machines are increasingly important in datacenters. Unlike WSCs, conventional datacenters tend to have a great deal of hardware and software heterogeneity to serve their varied customers inside an organization. WSC programmers customize third-party software or build their own, and WSCs have much more homogeneous hardware; the WSC goal is to make the hardware/software in the warehouse act like a single computer that typically runs a variety of applications. Often the largest cost in a conventional datacenter is the people to maintain it, whereas, as we shall see in [Section 6.4](#), in a well-designed WSC the server hardware is the greatest cost, and people costs shift from the topmost to nearly irrelevant. Conventional datacenters also don’t have the scale of a WSC, so they don’t get the economic benefits of scale mentioned above. Hence, while you might consider a WSC as an extreme datacenter, in that computers are housed separately in a space with special electrical and cooling infrastructure, typical datacenters share little with the challenges and opportunities of a WSC, either architecturally or operationally.

Since few architects understand the software that runs in a WSC, we start with the workload and programming model of a WSC.

---

## 6.2

### **Programming Models and Workloads for Warehouse-Scale Computers**

If a problem has no solution, it may not be a problem, but a fact—not to be solved, but to be coped with over time.

In addition to the public-facing Internet services such as search, video sharing, and social networking that make them famous, WSCs also run batch applications, such as converting videos into new formats or creating search indexes from Web crawls.

Today, the most popular framework for batch processing in a WSC is MapReduce [Dean and Ghemawat 2008] and its open-source twin Hadoop. Figure 6.2 shows the increasing popularity of MapReduce at Google over time. (Facebook runs Hadoop on 2000 batch-processing servers of the 60,000 servers it is estimated to have in 2011.) Inspired by the Lisp functions of the same name, Map first applies a programmer-supplied function to each logical input record. Map runs on thousands of computers to produce an intermediate result of key-value pairs. Reduce collects the output of those distributed tasks and collapses them using another programmer-defined function. With appropriate software support, both are highly parallel yet easy to understand and to use. Within 30 minutes, a novice programmer can run a MapReduce task on thousands of computers.

For example, one MapReduce program calculates the number of occurrences of every English word in a large collection of documents. Below is a simplified version of that program, which shows just the inner loop and assumes just one occurrence of all English words found in a document [Dean and Ghemawat 2008]:

```
map(String key, String value):
    // key: document name
    // value: document contents
    for each word w in value:
        EmitIntermediate(w, "1"); // Produce list of all words
reduce(String key, Iterator values):
    // key: a word
    // values: a list of counts
    int result = 0;
    for each v in values:
        result += ParseInt(v); // get integer from key-value pair
    Emit(AsString(result));
```

	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 (seconds)	634	874	395	475
Server years used	217	2002	11,081	25,562
Input data read (terabytes)	3288	52,254	403,152	544,130
Intermediate data (terabytes)	758	6743	34,774	90,120
Output data written (terabytes)	193	2970	14,018	57,520
Average number of servers per job	157	268	394	488

**Figure 6.2 Annual MapReduce usage at Google over time.** Over five years the number of MapReduce jobs increased by a factor of 100 and the average number of servers per job increased by a factor of 3. In the last two years the increases were factors of 1.6 and 1.2, respectively [Dean 2009]. Figure 6.16 on page 459 estimates that running the 2009 workload on Amazon’s cloud computing service EC2 would cost \$133M.



The function `EmitIntermediate` used in the `Map` function emits each word in the document and the value one. Then the `Reduce` function sums all the values per word for each document using `ParseInt()` to get the number of occurrences per word in all documents. The MapReduce runtime environment schedules map tasks and reduce task to the nodes of a WSC. (The complete version of the program is found in Dean and Ghemawat [2004].)

MapReduce can be thought of as a generalization of the single-instruction, multiple-data (SIMD) operation (Chapter 4)—except that you pass a function to be applied to the data—that is followed by a function that is used in a reduction of the output from the `Map` task. Because reductions are commonplace even in SIMD programs, SIMD hardware often offers special operations for them. For example, Intel’s recent AVX SIMD instructions include “horizontal” instructions that add pairs of operands that are adjacent in registers.

To accommodate variability in performance from thousands of computers, the MapReduce scheduler assigns new tasks based on how quickly nodes complete prior tasks. Obviously, a single slow task can hold up completion of a large MapReduce job. In a WSC, the solution to slow tasks is to provide software mechanisms to cope with such variability that is inherent at this scale. This approach is in sharp contrast to the solution for a server in a conventional data-center, where traditionally slow tasks mean hardware is broken and needs to be replaced or that server software needs tuning and rewriting. Performance heterogeneity is the norm for 50,000 servers in a WSC. For example, toward the end of a MapReduce program, the system will start backup executions on other nodes of the tasks that haven’t completed yet and take the result from whichever finishes first. In return for increasing resource usage a few percent, Dean and Ghemawat [2008] found that some large tasks complete 30% faster.

Another example of how WSCs differ is the use of data replication to overcome failures. Given the amount of equipment in a WSC, it’s not surprising that failures are commonplace, as the prior example attests. To deliver on 99.99% availability, systems software must cope with this reality in a WSC. To reduce operational costs, all WSCs use automated monitoring software so that one operator can be responsible for more than 1000 servers.

Programming frameworks such as MapReduce for batch processing and externally facing SaaS such as search rely upon internal software services for their success. For example, MapReduce relies on the Google File System (GFS) (Ghemawat, Gobioff, and Leung [2003]) to supply files to any computer, so that MapReduce tasks can be scheduled anywhere.

In addition to GFS, examples of such scalable storage systems include Amazon’s key value storage system Dynamo [DeCandia et al. 2007] and the Google record storage system Bigtable [Chang 2006]. Note that such systems often build upon each other. For example, Bigtable stores its logs and data on GFS, much as a relational database may use the file system provided by the kernel operating system.

These internal services often make different decisions than similar software running on single servers. As an example, rather than assuming storage is reliable, such as by using RAID storage servers, these systems often make complete



replicas of the data. Replicas can help with read performance as well as with availability; with proper placement, replicas can overcome many other system failures, like those in [Figure 6.1](#). Some systems use erasure encoding rather than full replicas, but the constant is cross-server redundancy rather than within-a-server or within-a-storage array redundancy. Hence, failure of the entire server or storage device doesn't negatively affect availability of the data.

Another example of the different approach is that WSC storage software often uses relaxed consistency rather than following all the ACID (atomicity, consistency, isolation, and durability) requirements of conventional database systems. The insight is that it's important for multiple replicas of data to agree *eventually*, but for most applications they need not be in agreement at all times. For example, eventual consistency is fine for video sharing. Eventual consistency makes storage systems much easier to scale, which is an absolute requirement for WSCs.

The workload demands of these public interactive services all vary considerably; even a popular global service such as Google search varies by a factor of two depending on the time of day. When you factor in weekends, holidays, and popular times of year for some applications—such as photograph sharing services after Halloween or online shopping before Christmas—you can see considerably greater variation in server utilization for Internet services. [Figure 6.3](#) shows average utilization of 5000 Google servers over a 6-month period. Note that less than 0.5% of servers averaged 100% utilization, and most servers operated between 10% and 50% utilization. Stated alternatively, just 10% of all servers were utilized more than 50%. Hence, it's much more important for servers in a WSC to perform well while doing little than to just to perform efficiently at their peak, as they rarely operate at their peak.

In summary, WSC hardware and software must cope with variability in load based on user demand and in performance and dependability due to the vagaries of hardware at this scale.

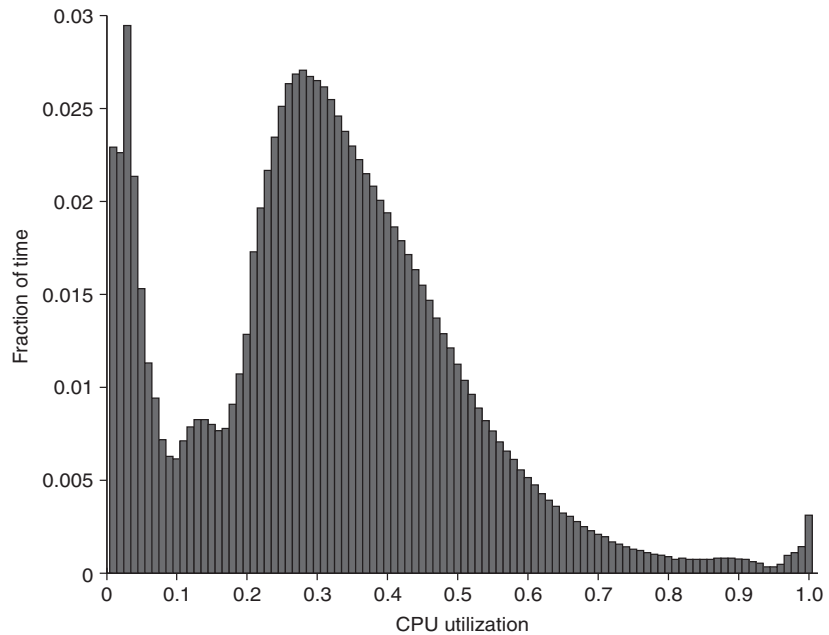
---

**Example** As a result of measurements like those in [Figure 6.3](#), the SPECpower benchmark measures power and performance from 0% load to 100% in 10% increments (see [Chapter 1](#)). The overall single metric that summarizes this benchmark is the sum of all the performance measures (server-side Java operations per second) divided by the sum of all power measurements in watts. Thus, each level is equally likely. How would the numbers summary metric change if the levels were weighted by the utilization frequencies in [Figure 6.3](#)?

**Answer** [Figure 6.4](#) shows the original weightings and the new weighting that match [Figure 6.3](#). These weightings reduce the performance summary by 30% from 3210 ssj\_ops/watt to 2454.

---

Given the scale, software must handle failures, which means there is little reason to buy “gold-plated” hardware that reduces the frequency of failures. The primary impact would be to increase cost. Barroso and Hölzle [2009] found a factor of 20 difference in price-performance between a high-end



**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.

Load	Performance	Watts	SPEC weightings	Weighted performance	Weighted watts	Figure 6.3 weightings	Weighted performance	Weighted watts
100%	2,889,020	662	9.09%	262,638	60	0.80%	22,206	5
90%	2,611,130	617	9.09%	237,375	56	1.20%	31,756	8
80%	2,319,900	576	9.09%	210,900	52	1.50%	35,889	9
70%	2,031,260	533	9.09%	184,660	48	2.10%	42,491	11
60%	1,740,980	490	9.09%	158,271	45	5.10%	88,082	25
50%	1,448,810	451	9.09%	131,710	41	11.50%	166,335	52
40%	1,159,760	416	9.09%	105,433	38	19.10%	221,165	79
30%	869,077	382	9.09%	79,007	35	24.60%	213,929	94
20%	581,126	351	9.09%	52,830	32	15.30%	88,769	54
10%	290,762	308	9.09%	26,433	28	8.00%	23,198	25
0%	0	181	9.09%	0	16	10.90%	0	20
Total	15,941,825	4967		1,449,257	452		933,820	380
				ssj_ops/Watt	3210		ssj_ops/Watt	2454

**Figure 6.4** SPECpower result from Figure 6.17 using the weightings from Figure 6.3 instead of even weightings.

HP shared-memory multiprocessor and a commodity HP server when running the TPC-C database benchmark. Unsurprisingly, Google buys low-end commodity servers.

Such WSC services also tend to develop their own software rather than buy third-party commercial software, in part to cope with the huge scale and in part to save money. For example, even on the best price-performance platform for TPC-C in 2011, including the cost of the Oracle database and Windows operating system doubles the cost of the Dell Poweredge 710 server. In contrast, Google runs Bigtable and the Linux operating system on its servers, for which it pays no licensing fees.

Given this review of the applications and systems software of a WSC, we are ready to look at the computer architecture of a WSC.

---

## 6.3

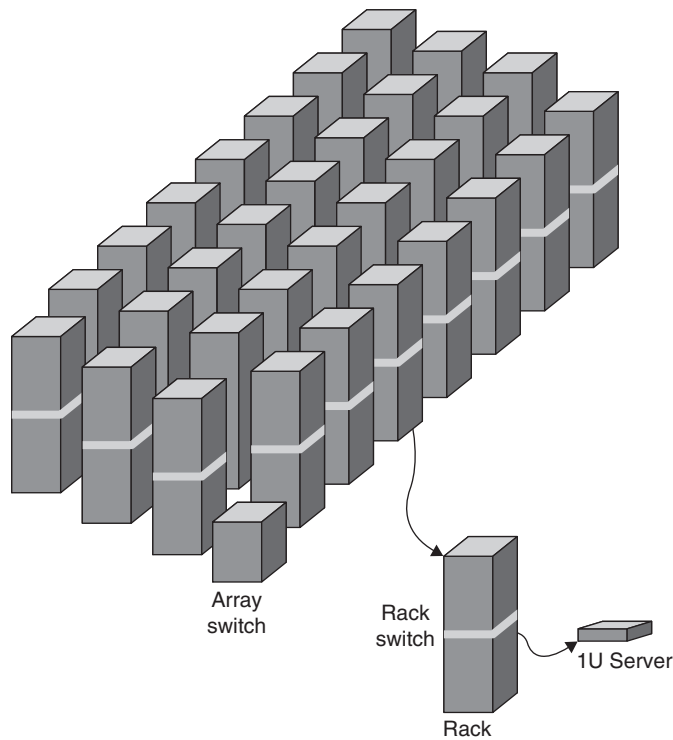
### Computer Architecture of Warehouse-Scale Computers

Networks are the connective tissue that binds 50,000 servers together. Analogous to the memory hierarchy of [Chapter 2](#), WSCs use a hierarchy of networks. [Figure 6.5](#) shows one example. Ideally, the combined network would provide nearly the performance of a custom high-end switch for 50,000 servers at nearly the cost per port of a commodity switch designed for 50 servers. As we shall see in [Section 6.6](#), the current solutions are far from that ideal, and networks for WSCs are an area of active exploration.

The 19-inch (48.26-cm) rack is still the standard framework to hold servers, despite this standard going back to railroad hardware from the 1930s. Servers are measured in the number of rack units (U) that they occupy in a rack. One U is 1.75 inches (4.45 cm) high, and that is the minimum space a server can occupy.

A 7-foot (213.36-cm) rack offers 48 U, so it's not a coincidence that the most popular switch for a rack is a 48-port Ethernet switch. This product has become a commodity that costs as little as \$30 per port for a 1 Gbit/sec Ethernet link in 2011 [[Barroso and Hölzle 2009](#)]. Note that the bandwidth within the rack is the same for each server, so it does not matter where the software places the sender and the receiver as long as they are within the same rack. This flexibility is ideal from a software perspective.

These switches typically offer two to eight uplinks, which leave the rack to go to the next higher switch in the network hierarchy. Thus, the bandwidth leaving the rack is 6 to 24 times smaller—48/8 to 48/2—than the bandwidth within the rack. This ratio is called *oversubscription*. Alas, large oversubscription means programmers must be aware of the performance consequences when placing senders and receivers in different racks. This increased software-scheduling burden is another argument for network switches designed specifically for the datacenter.



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

## Storage

A natural design is to fill a rack with servers, minus whatever space you need for the commodity Ethernet rack switch. This design leaves open the question of where the storage is placed. From a hardware construction perspective, the simplest solution would be to include disks inside the server, and rely on Ethernet connectivity for access to information on the disks of remote servers. The alternative would be to use network attached storage (NAS), perhaps over a storage network like Infiniband. The NAS solution is generally more expensive per terabyte of storage, but it provides many features, including RAID techniques to improve dependability of the storage.

As you might expect from the philosophy expressed in the prior section, WSCs generally rely on local disks and provide storage software that handles connectivity and dependability. For example, GFS uses local disks and maintains at least three replicas to overcome dependability problems. This redundancy covers not just local disk failures, but also power failures to racks and to whole clusters. The eventual consistency flexibility of GFS lowers the cost of keeping replicas consistent, which also reduces the network bandwidth requirements of the storage

system. Local access patterns also mean high bandwidth to local storage, as we'll see shortly.

Beware that there is confusion about the term *cluster* when talking about the architecture of a WSC. Using the definition in [Section 6.1](#), a WSC is just an extremely large cluster. In contrast, Barroso and Hölzle [2009] used the term cluster to mean the next-sized grouping of computers, in this case about 30 racks. In this chapter, to avoid confusion we will use the term *array* to mean a collection of racks, preserving the original meaning of the word cluster to mean anything from a collection of networked computers within a rack to an entire warehouse full of networked computers.

### Array Switch

The switch that connects an array of racks is considerably more expensive than the 48-port commodity Ethernet switch. This cost is due in part because of the higher connectivity and in part because the bandwidth through the switch must be much higher to reduce the oversubscription problem. Barroso and Hölzle [2009] reported that a switch that has 10 times the *bisection bandwidth*—basically, the worst-case internal bandwidth—of a rack switch costs about 100 times as much. One reason is that the cost of switch bandwidth for  $n$  ports can grow as  $n^2$ .

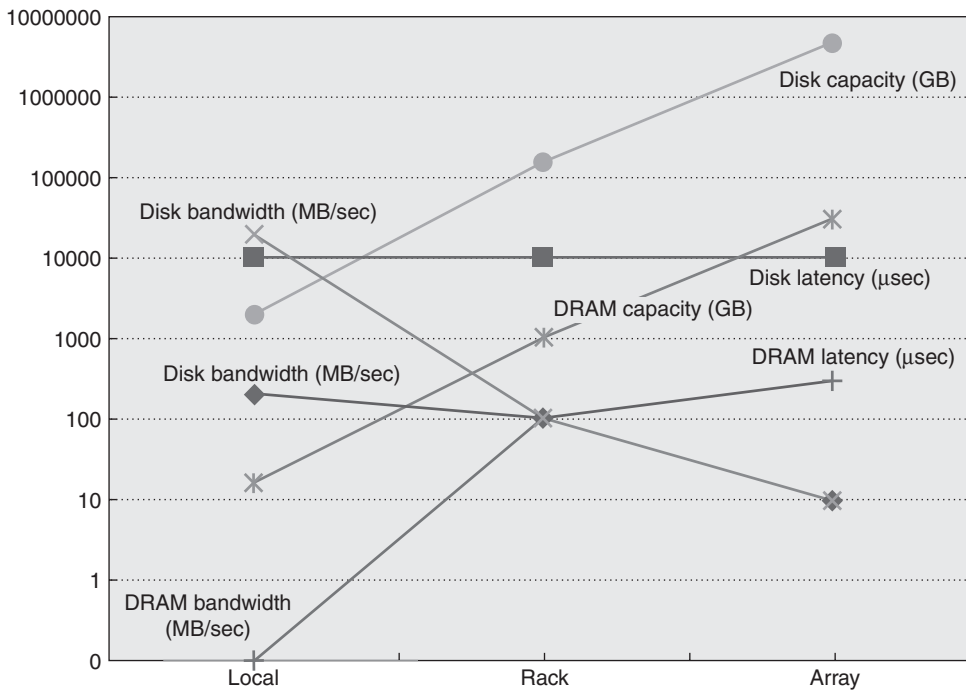
Another reason for the high costs is that these products offer high profit margins for the companies that produce them. They justify such prices in part by providing features such as packet inspection that are expensive because they must operate at very high rates. For example, network switches are major users of content-addressable memory chips and of field-programmable gate arrays (FPGAs), which help provide these features, but the chips themselves are expensive. While such features may be valuable for Internet settings, they are generally unused inside the datacenter.

### WSC Memory Hierarchy

[Figure 6.6](#) shows the latency, bandwidth, and capacity of memory hierarchy inside a WSC, and [Figure 6.7](#) shows the same data visually. These figures are based on the following assumptions [Barroso and Hölzle 2009]:

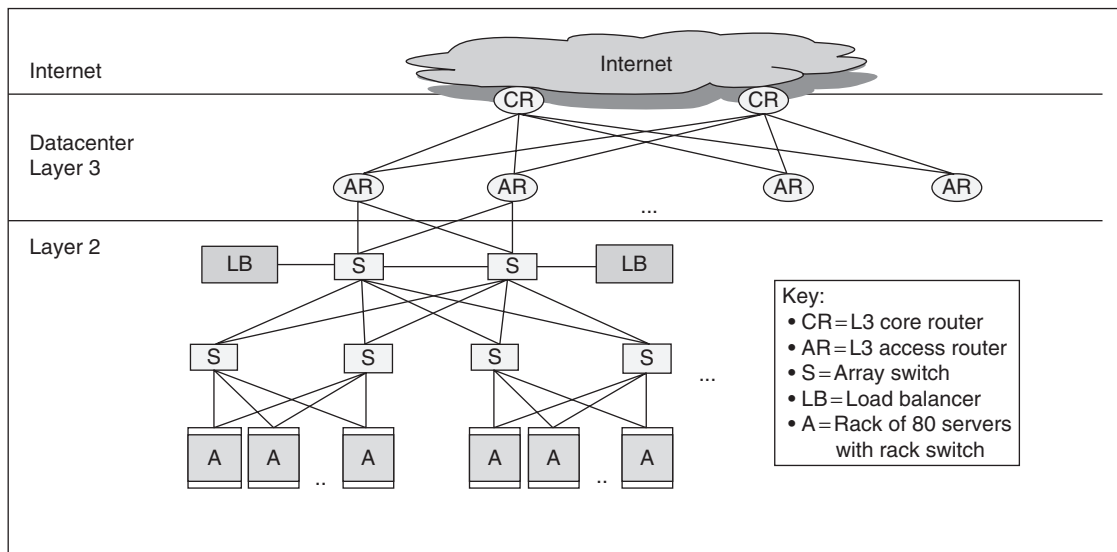
	Local	Rack	Array
DRAM latency (microseconds)	0.1	100	300
Disk latency (microseconds)	10,000	11,000	12,000
DRAM bandwidth (MB/sec)	20,000	100	10
Disk bandwidth (MB/sec)	200	100	10
DRAM capacity (GB)	16	1040	31,200
Disk capacity (GB)	2000	160,000	4,800,000

**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.



**Figure 6.7** Graph of latency, bandwidth, and capacity of the memory hierarchy of a WSC for data in Figure 6.6 [Barroso and Hölzle 2009].

- Each server contains 16 GBytes of memory with a 100-nanosecond access time and transfers at 20 GBytes/sec and 2 terabytes of disk that offers a 10-millisecond access time and transfers at 200 MBytes/sec. There are two sockets per board, and they share one 1 Gbit/sec Ethernet port.
- Every pair of racks includes one rack switch and holds 80 2U servers (see Section 6.7). Networking software plus switch overhead increases the latency to DRAM to 100 microseconds and the disk access latency to 11 milliseconds. Thus, the total storage capacity of a rack is roughly 1 terabyte of DRAM and 160 terabytes of disk storage. The 1 Gbit/sec Ethernet limits the remote bandwidth to DRAM or disk within the rack to 100 MBytes/sec.
- The array switch can handle 30 racks, so storage capacity of an array goes up by a factor of 30: 30 terabytes of DRAM and 4.8 petabytes of disk. The array switch hardware and software increases latency to DRAM within an array to 500 microseconds and disk latency to 12 milliseconds. The bandwidth of the array switch limits the remote bandwidth to either array DRAM or array disk to 10 MBytes/sec.



**Figure 6.8** The Layer 3 network used to link arrays together and to the Internet [Greenberg et al. 2009]. Some WSCs use a separate *border router* to connect the Internet to the datacenter Layer 3 switches.

Figures 6.6 and 6.7 show that network overhead dramatically increases latency from local DRAM to rack DRAM and array DRAM, but both still have more than 10 times better latency than the local disk. The network collapses the difference in bandwidth between rack DRAM and rack disk and between array DRAM and array disk.

The WSC needs 20 arrays to reach 50,000 servers, so there is one more level of the networking hierarchy. Figure 6.8 shows the conventional Layer 3 routers to connect the arrays together and to the Internet.

Most applications fit on a single array within a WSC. Those that need more than one array use *sharding* or *partitioning*, meaning that the dataset is split into independent pieces and then distributed to different arrays. Operations on the whole dataset are sent to the servers hosting the pieces, and the results are coalesced by the client computer.

---

**Example** What is the average memory latency assuming that 90% of accesses are local to the server, 9% are outside the server but within the rack, and 1% are outside the rack but within the array?

**Answer** The average memory access time is

$$(90\% \times 0.1) + (9\% \times 100) + (1\% \times 300) = 0.09 + 9 + 3 = 12.09 \text{ microseconds}$$

or a factor of more than 120 slowdown versus 100% local accesses. Clearly, locality of access within a server is vital for WSC performance.

---



---

**Example** How long does it take to transfer 1000 MB between disks within the server, between servers in the rack, and between servers in different racks in the array? How much faster is it to transfer 1000 MB between DRAM in the three cases?

**Answer** A 1000 MB transfer between disks takes:

Within server =  $1000/200 = 5$  seconds

Within rack =  $1000/100 = 10$  seconds

Within array =  $1000/10 = 100$  seconds

A memory-to-memory block transfer takes

Within server =  $1000/20000 = 0.05$  seconds

Within rack =  $1000/100 = 10$  seconds

Within array =  $1000/10 = 100$  seconds

Thus, for block transfers outside a single server, it doesn't even matter whether the data are in memory or on disk since the rack switch and array switch are the bottlenecks. These performance limits affect the design of WSC software and inspire the need for higher performance switches (see [Section 6.6](#)).

---

Given the architecture of the IT equipment, we are now ready to see how to house, power, and cool it and to discuss the cost to build and operate the whole WSC, as compared to just the IT equipment within it.

## 6.4

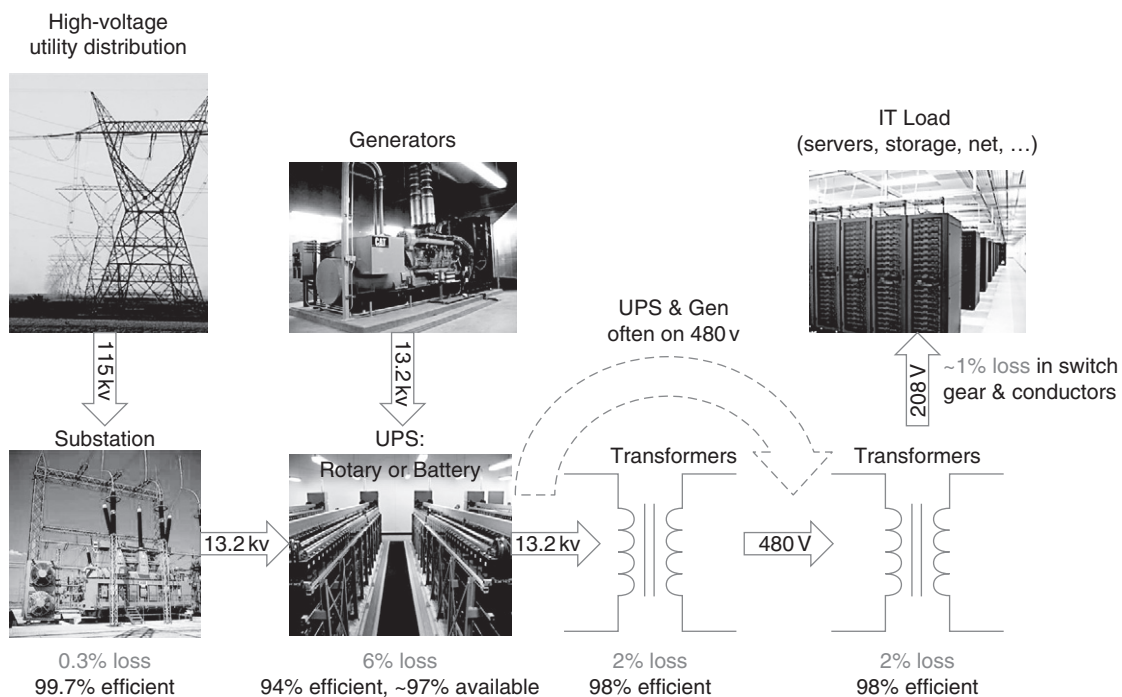
### Physical Infrastructure and Costs of Warehouse-Scale Computers

To build a WSC, you first need to build a warehouse. One of the first questions is where? Real estate agents emphasize location, but location for a WSC means proximity to Internet backbone optical fibers, low cost of electricity, and low risk from environmental disasters, such as earthquakes, floods, and hurricanes. For a company with many WSCs, another concern is finding a place geographically near a current or future population of Internet users, so as to reduce latency over the Internet. There are also many more mundane concerns, such as property tax rates.

Infrastructure costs for power distribution and cooling dwarf the construction costs of a WSC, so we concentrate on the former. [Figures 6.9](#) and [6.10](#) show the power distribution and cooling infrastructure within a WSC.

Although there are many variations deployed, in North America electrical power typically goes through about five steps and four voltage changes on the way to the server, starting with the high-voltage lines at the utility tower of 115,000 volts:

1. The substation switches from 115,000 volts to medium-voltage lines of 13,200 volts, with an efficiency of 99.7%.



**Figure 6.9** Power distribution and where losses occur. Note that the best improvement is 11%. (From Hamilton [2010].)

2. To prevent the whole WSC from going offline if power is lost, a WSC has an uninterruptible power supply (UPS), just as some servers do. In this case, it involves large diesel engines that can take over from the utility company in an emergency and batteries or flywheels to maintain power after the service is lost but before the diesel engines are ready. The generators and batteries can take up so much space that they are typically located in a separate room from the IT equipment. The UPS plays three roles: power conditioning (maintain proper voltage levels and other characteristics), holding the electrical load while the generators start and come on line, and holding the electrical load when switching back from the generators to the electrical utility. The efficiency of this very large UPS is 94%, so the facility loses 6% of the power by having a UPS. The WSC UPS can account for 7% to 12% of the cost of all the IT equipment.
3. Next in the system is a power distribution unit (PDU) that converts to low-voltage, internal, three-phase power at 480 volts. The conversion efficiency is 98%. A typical PDU handles 75 to 225 kilowatts of load, or about 10 racks.
4. There is yet another down step to two-phase power at 208 volts that servers can use, once again at 98% efficiency. (Inside the server, there are more steps to bring the voltage down to what chips can use; see [Section 6.7.](#))

- The connectors, breakers, and electrical wiring to the server have a collective efficiency of 99%.

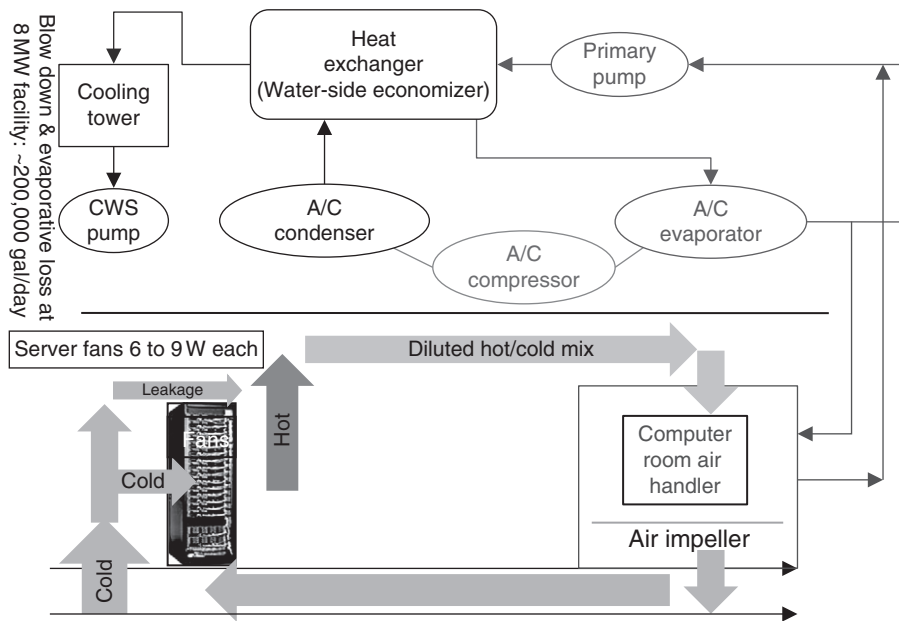
WSCs outside North America use different conversion values, but the overall design is similar.

Putting it all together, the efficiency of turning 115,000-volt power from the utility into 208-volt power that servers can use is 89%:

$$99.7\% \times 94\% \times 98\% \times 98\% \times 99\% = 89\%$$

This overall efficiency leaves only a little over 10% room for improvement, but as we shall see, engineers still try to make it better.

There is considerably more opportunity for improvement in the cooling infrastructure. The computer room air-conditioning (CRAC) unit cools the air in the server room using chilled water, similar to how a refrigerator removes heat by releasing it outside of the refrigerator. As a liquid absorbs heat, it evaporates. Conversely, when a liquid releases heat, it condenses. Air conditioners pump the liquid into coils under low pressure to evaporate and absorb heat, which is then sent to an external condenser where it is released. Thus, in a CRAC unit, fans push warm air past a set of coils filled with cold water and a pump moves the warmed water to the external chillers to be cooled down. The cool air for servers is typically between 64°F and 71°F (18°C and 22°C). Figure 6.10 shows the large collection of fans and water pumps that move air and water throughout the system.



**Figure 6.10** Mechanical design for cooling systems. CWS stands for circulating water system. (From Hamilton [2010].)

Clearly, one of the simplest ways to improve energy efficiency is simply to run the IT equipment at higher temperatures so that the air need not be cooled as much. Some WSCs run their equipment considerably above 71°F (22°C).

In addition to chillers, cooling towers are used in some datacenters to leverage the colder outside air to cool the water before it is sent to the chillers. The temperature that matters is called the *wet-bulb temperature*. The wet-bulb temperature is measured by blowing air on the bulb end of a thermometer that has water on it. It is the lowest temperature that can be achieved by evaporating water with air.

Warm water flows over a large surface in the tower, transferring heat to the outside air via evaporation and thereby cooling the water. This technique is called *airside economization*. An alternative is use cold water instead of cold air. Google's WSC in Belgium uses a water-to-water intercooler that takes cold water from an industrial canal to chill the warm water from inside the WSC.

Airflow is carefully planned for the IT equipment itself, with some designs even using airflow simulators. Efficient designs preserve the temperature of the cool air by reducing the chances of it mixing with hot air. For example, a WSC can have alternating aisles of hot air and cold air by orienting servers in opposite directions in alternating rows of racks so that hot exhaust blows in alternating directions.

In addition to energy losses, the cooling system also uses up a lot of water due to evaporation or to spills down sewer lines. For example, an 8 MW facility might use 70,000 to 200,000 gallons of water per day.

The relative power costs of cooling equipment to IT equipment in a typical datacenter [Barroso and Hölzle 2009] are as follows:

- Chillers account for 30% to 50% of the IT equipment power.
- CRAC accounts for 10% to 20% of the IT equipment power, due mostly to fans.

Surprisingly, it's not obvious to figure out how many servers a WSC can support after you subtract the overheads for power distribution and cooling. The so-called *nameplate power rating* from the server manufacturer is always conservative; it's the maximum power a server can draw. The first step then is to measure a single server under a variety of workloads to be deployed in the WSC. (Networking is typically about 5% of power consumption, so it can be ignored to start.)

To determine the number of servers for a WSC, the available power for IT could just be divided by the measured server power; however, this would again be too conservative according to Fan, Weber, and Barroso [2007]. They found that there is a significant gap between what thousands of servers could theoretically do in the worst case and what they will do in practice, since no real workloads will keep thousands of servers all simultaneously at their peaks. They found that they could safely oversubscribe the number of servers by as much as 40% based on the power of a single server. They recommended that WSC architects should do that to increase the average utilization of power within a WSC; however, they also suggested using extensive monitoring software along

with a safety mechanism that deschedules lower priority tasks in case the workload shifts.

Breaking down power usage inside the IT equipment itself, Barroso and Hölzle [2009] reported the following for a Google WSC deployed in 2007:

- 33% of power for processors
- 30% for DRAM
- 10% for disks
- 5% for networking
- 22% for other reasons (inside the server)

### Measuring Efficiency of a WSC

A widely used, simple metric to evaluate the efficiency of a datacenter or a WSC is called *power utilization effectiveness* (or *PUE*):

$$\text{PUE} = (\text{Total facility power})/(\text{IT equipment power})$$

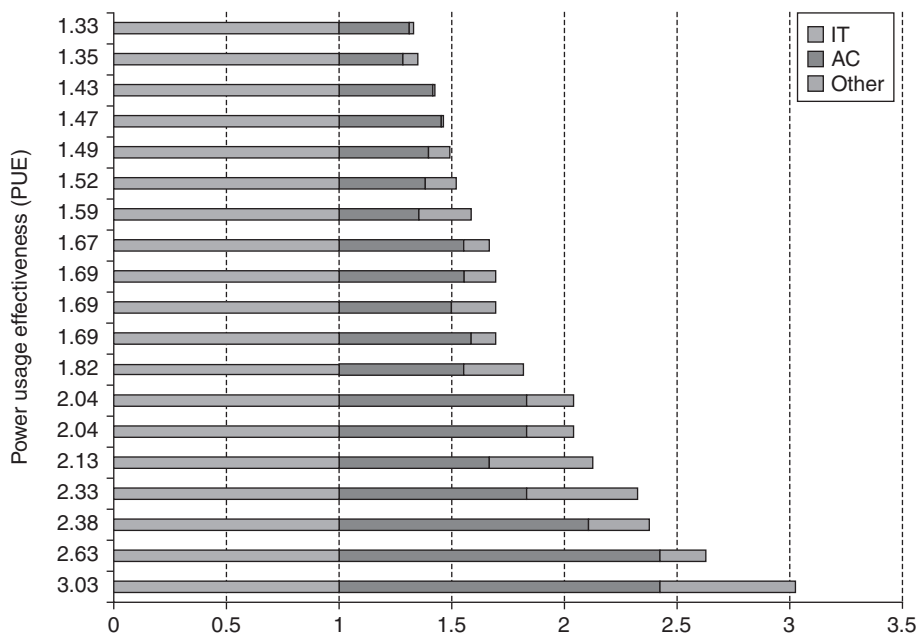
Thus, PUE must be greater than or equal to 1, and the bigger the PUE the less efficient the WSC.

Greenberg et al. [2006] reported on the PUE of 19 datacenters and the portion of the overhead that went into the cooling infrastructure. Figure 6.11 shows what they found, sorted by PUE from most to least efficient. The median PUE is 1.69, with the cooling infrastructure using more than half as much power as the servers themselves—on average, 0.55 of the 1.69 is for cooling. Note that these are average PUEs, which can vary daily depending on workload and even external air temperature, as we shall see.

Since performance per dollar is the ultimate metric, we still need to measure performance. As Figure 6.7 above shows, bandwidth drops and latency increases depending on the distance to the data. In a WSC, the DRAM bandwidth within a server is 200 times larger than within a rack, which in turn is 10 times larger than within an array. Thus, there is another kind of locality to consider in the placement of data and programs within a WSC.

While designers of a WSC often focus on bandwidth, programmers developing applications on a WSC are also concerned with latency, since latency is visible to users. Users' satisfaction and productivity are tied to response time of a service. Several studies from the timesharing days report that user productivity is inversely proportional to time for an interaction, which was typically broken down into human entry time, system response time, and time for the person to think about the response before entering the next entry. The results of experiments showed that cutting system response time 30% shaved the time of an interaction by 70%. This implausible result is explained by human nature: People need less time to think when given a faster response, as they are less likely to get distracted and remain "on a roll."

Figure 6.12 shows the results of such an experiment for the Bing search engine, where delays of 50 ms to 2000 ms were inserted at the search server. As expected



**Figure 6.11** Power utilization efficiency of 19 datacenters in 2006 [Greenberg et al. 2006]. The power for air conditioning (AC) and other uses (such as power distribution) is normalized to the power for the IT equipment in calculating the PUE. Thus, power for IT equipment must be 1.0 and AC varies from about 0.30 to 1.40 times the power of the IT equipment. Power for “other” varies from about 0.05 to 0.60 of the IT equipment.

from previous studies, time to next click roughly doubled the delay; that is, a 200 ms delay at the server led to a 500 ms increase in time to next click. Revenue dropped linearly with increasing delay, as did user satisfaction. A separate study on the Google search engine found that these effects lingered long after the 4-week experiment ended. Five weeks later, there were 0.1% fewer searchers per day for users who experienced 200 ms delays, and there were 0.2% fewer searches from users who experienced 400 ms delays. Given the amount of money made in search, even such small changes are disconcerting. In fact, the results were so negative that they ended the experiment prematurely [Schurman and Brutlag 2009].

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].

Because of this extreme concern with satisfaction of all users of an Internet service, performance goals are typically specified that a high percentage of requests be below a latency threshold rather than just offer a target for the average latency. Such threshold goals are called *service level objectives (SLOs)* or *service level agreements (SLAs)*. An SLO might be that 99% of requests must be below 100 milliseconds. Thus, the designers of Amazon's Dynamo key-value storage system decided that, for services to offer good latency on top of Dynamo, their storage system had to deliver on its latency goal 99.9% of the time [DeCandia et al. 2007]. For example, one improvement of Dynamo helped the 99.9th percentile much more than the average case, which reflects their priorities.

### Cost of a WSC

As mentioned in the introduction, unlike most architects, designers of WSCs worry about operational costs as well as the cost to build the WSC. Accounting labels the former costs as *operational expenditures (OPEX)* and the latter costs as *capital expenditures (CAPEX)*.

To put the cost of energy into perspective, Hamilton [2010] did a case study to estimate the costs of a WSC. He determined that the CAPEX of this 8 MW facility was \$88M, and that the roughly 46,000 servers and corresponding networking equipment added another \$79M to the CAPEX for the WSC. Figure 6.13 shows the rest of the assumptions for the case study.

We can now price the total cost of energy, since U.S. accounting rules allow us to convert CAPEX into OPEX. We can just amortize CAPEX as a fixed amount each month for the effective life of the equipment. Figure 6.14 breaks down the monthly OPEX for this case study. Note that the amortization rates differ significantly, from 10 years for the facility to 4 years for the networking equipment and 3 years for the servers. Hence, the WSC facility lasts a decade, but you need to replace the servers every 3 years and the networking equipment every 4 years. By amortizing the CAPEX, Hamilton came up with a monthly OPEX, including accounting for the cost of borrowing money (5% annually) to pay for the WSC. At \$3.8M, the monthly OPEX is about 2% of the CAPEX.

This figure allows us to calculate a handy guideline to keep in mind when making decisions about which components to use when being concerned about energy. The fully burdened cost of a watt per year in a WSC, including the cost of amortizing the power and cooling infrastructure, is

$$\frac{\text{Monthly cost of infrastructure} + \text{monthly cost of power}}{\text{Facility size in watts}} \times 12 = \frac{\$765\text{K} + \$475\text{K}}{8\text{M}} \times 12 = \$1.86$$

The cost is roughly \$2 per watt-year. Thus, to reduce costs by saving energy you shouldn't spend more than \$2 per watt-year (see Section 6.8).

Note that more than a third of OPEX is related to power, with that category trending up while server costs are trending down over time. The networking



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%
<b>CAPEX for facility (not including IT equipment)</b>	<b>\$88,000,000</b>
Number of servers	45,978
Cost/server	\$1450
<b>CAPEX for servers</b>	<b>\$66,700,000</b>
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
<b>CAPEX for networking gear</b>	<b>\$12,810,000</b>
<b>Total CAPEX for WSC</b>	<b>\$167,510,000</b>
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.

equipment is significant at 8% of total OPEX and 19% of the server CAPEX, and networking equipment is not trending down as quickly as servers are. This difference is especially true for the switches in the networking hierarchy above the rack, which represent most of the networking costs (see Section 6.6). People costs for security and facilities management are just 2% of OPEX. Dividing the OPEX in Figure 6.14 by the number of servers and hours per month, the cost is about \$0.11 per server per hour.

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.

**Example** The cost of electricity varies by region in the United States from \$0.03 to \$0.15 per kilowatt-hour. What is the impact on hourly server costs of these two extreme rates?

**Answer** We multiply the critical load of 8 MW by the PUE and by the average power usage from Figure 6.13 to calculate the average power usage:

$$8 \times 1.45 \times 80\% = 9.28 \text{ Megawatts}$$

The monthly cost for power then goes from \$475,000 in Figure 6.14 to \$205,000 at \$0.03 per kilowatt-hour and to \$1,015,000 at \$0.15 per kilowatt-hour. These changes in electricity cost change the hourly server costs from \$0.11 to \$0.10 and \$0.13, respectively.

**Example** What would happen to monthly costs if the amortization times were all made to be the same—say, 5 years? How does that change the hourly cost per server?

**Answer** The spreadsheet is available online at <http://mvdirona.com/jrh/TalksAndPapers/PerspectivesDataCenterCostAndPower.xls>. Changing the amortization time to 5 years changes the first four rows of Figure 6.14 to

Servers	\$1,260,000	37%
Networking equipment	\$242,000	7%
Power and cooling infrastructure	\$1,115,000	33%
Other infrastructure	\$245,000	7%

and the total monthly OPEX is \$3,422,000. If we replaced everything every 5 years, the cost would be \$0.103 per server hour, with more of the amortized costs now being for the facility rather than the servers, as in [Figure 6.14](#).

The rate of \$0.11 per server per hour can be much less than the cost for many companies that own and operate their own (smaller) conventional datacenters. The cost advantage of WSCs led large Internet companies to offer computing as a utility where, like electricity, you pay only for what you use. Today, utility computing is better known as cloud computing.

## 6.5

### Cloud Computing: The Return of Utility Computing

If computers of the kind I have advocated become the computers of the future, then computing may someday be organized as a public utility just as the telephone system is a public utility. . . . The computer utility could become the basis of a new and important industry.

**John McCarthy**

*MIT centennial celebration (1961)*

Driven by the demand of an increasing number of users, Internet companies such as Amazon, Google, and Microsoft built increasingly larger warehouse-scale computers from commodity components. This demand led to innovations in systems software to support operating at this scale, including Bigtable, Dynamo, GFS, and MapReduce. It also demanded improvement in operational techniques to deliver a service available at least 99.99% of the time despite component failures and security attacks. Examples of these techniques include failover, firewalls, virtual machines, and protection against distributed denial-of-service attacks. With the software and expertise providing the ability to scale and increasing customer demand that justified the investment, WSCs with 50,000 to 100,000 servers have become commonplace in 2011.

With increasing scale came increasing economies of scale. Based on a study in 2006 that compared a WSC with a datacenter with only 1000 servers, Hamilton [2010] reported the following advantages:

- *5.7 times reduction in storage costs*—It cost the WSC \$4.6 per GByte per year for disk storage versus \$26 per GByte for the datacenter.
- *7.1 times reduction in administrative costs*—The ratio of servers per administrator was over 1000 for the WSC versus just 140 for the datacenter.
- *7.3 times reduction in networking costs*—Internet bandwidth cost the WSC \$13 per Mbit/sec/month versus \$95 for the datacenter. Unsurprisingly, you can negotiate a much better price per Mbit/sec if you order 1000 Mbit/sec than if you order 10 Mbit/sec.

Another economy of scale comes during purchasing. The high level of purchasing leads to volume discount prices on the servers and networking gear. It also allows optimization of the supply chain. Dell, IBM, and SGI will deliver on new orders in a week to a WSC instead of 4 to 6 months. Short delivery time makes it much easier to grow the utility to match the demand.

Economies of scale also apply to operational costs. From the prior section, we saw that many datacenters operate with a PUE of 2.0. Large firms can justify hiring mechanical and power engineers to develop WSCs with lower PUEs, in the range of 1.2 (see [Section 6.7](#)).

Internet services need to be distributed to multiple WSCs for both dependability and to reduce latency, especially for international markets. All large firms use multiple WSCs for that reason. It's much more expensive for individual firms to create multiple, small datacenters around the world than a single datacenter in the corporate headquarters.

Finally, for the reasons presented in [Section 6.1](#), servers in datacenters tend to be utilized only 10% to 20% of the time. By making WSCs available to the public, uncorrelated peaks between different customers can raise average utilization above 50%.

Thus, economies of scale for a WSC offer factors of 5 to 7 for several components of a WSC plus a few factors of 1.5 to 2 for the entire WSC.

While there are many cloud computing providers, we feature Amazon Web Services (AWS) in part because of its popularity and in part because of the low level and hence more flexible abstraction of their service. Google App Engine and Microsoft Azure raise the level of abstraction to managed runtimes and to offer automatic scaling services, which are a better match to some customers, but not as good a match as AWS to the material in this book.

## Amazon Web Services

Utility computing goes back to commercial timesharing systems and even batch processing systems of the 1960s and 1970s, where companies only paid for a terminal and a phone line and then were billed based on how much computing they used. Many efforts since the end of timesharing then have tried to offer such pay as you go services, but they were often met with failure.

When Amazon started offering utility computing via the Amazon Simple Storage Service (Amazon S3) and then Amazon Elastic Computer Cloud (Amazon EC2) in 2006, it made some novel technical and business decisions:

- *Virtual Machines*. Building the WSC using x86-commodity computers running the Linux operating system and the Xen virtual machine solved several problems. First, it allowed Amazon to protect users from each other. Second, it simplified software distribution within a WSC, in that customers only need install an image and then AWS will automatically distribute it to all the instances being used. Third, the ability to kill a virtual machine reliably

makes it easy for Amazon and customers to control resource usage. Fourth, since Virtual Machines can limit the rate at which they use the physical processors, disks, and the network as well as the amount of main memory, that gave AWS multiple price points: the lowest price option by packing multiple virtual cores on a single server, the highest price option of exclusive access to all the machine resources, as well as several intermediary points. Fifth, Virtual Machines hide the identity of older hardware, allowing AWS to continue to sell time on older machines that might otherwise be unattractive to customers if they knew their age. Finally, Virtual Machines allow AWS to introduce new and faster hardware by either packing even more virtual cores per server or simply by offering instances that have higher performance per virtual core; virtualization means that offered performance need not be an integer multiple of the performance of the hardware.

- *Very low cost.* When AWS announced a rate of \$0.10 per hour per instance in 2006, it was a startlingly low amount. An instance is one Virtual Machine, and at \$0.10 per hour AWS allocated two instances per core on a multicore server. Hence, one EC2 computer unit is equivalent to a 1.0 to 1.2 GHz AMD Opteron or Intel Xeon of that era.
- *(Initial) reliance on open source software.* The availability of good-quality software that had no licensing problems or costs associated with running on hundreds or thousands of servers made utility computing much more economical for both Amazon and its customers. More recently, AWS started offering instances including commercial third-party software at higher prices.
- *No (initial) guarantee of service.* Amazon originally promised only best effort. The low cost was so attractive that many could live without a service guarantee. Today, AWS provides availability SLAs of up to 99.95% on services such as Amazon EC2 and Amazon S3. Additionally, Amazon S3 was designed for 99.999999999% durability by saving multiple replicas of each object across multiple locations. That is, the chances of permanently losing an object are one in 100 billion. AWS also provides a Service Health Dashboard that shows the current operational status of each of the AWS services in real time, so that AWS uptime and performance are fully transparent.
- *No contract required.* In part because the costs are so low, all that is necessary to start using EC2 is a credit card.

Figure 6.15 shows the hourly price of the many types of EC2 instances in 2011. In addition to computation, EC2 charges for long-term storage and for Internet traffic. (There is no cost for network traffic inside AWS regions.) Elastic Block Storage costs \$0.10 per GByte per month and \$0.10 per million I/O requests. Internet traffic costs \$0.10 per GByte going to EC2 and \$0.08 to \$0.15 per GByte leaving from EC2, depending on the volume. Putting this into historical perspective, for \$100 per month you can use the equivalent capacity of the sum of the capacities of all magnetic disks produced in 1960!

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

**Figure 6.15** Price and characteristics of on-demand EC2 instances in the United States in the Virginia region in January 2011. Micro Instances are the newest and cheapest category, and they offer short bursts of up to 2.0 compute units for just \$0.02 per hour. Customers report that Micro Instances average about 0.5 compute units. Cluster-Compute Instances in the last row, which AWS identifies as dedicated dual-socket Intel Xeon X5570 servers with four cores per socket running at 2.93 GHz, offer 10 Gigabit/sec networks. They are intended for HPC applications. AWS also offers Spot Instances at much less cost, where you set the price you are willing to pay and the number of instances you are willing to run, and then AWS will run them when the spot price drops below your level. They run until you stop them or the spot price exceeds your limit. One sample during the daytime in January 2011 found that the spot price was a factor of 2.3 to 3.1 lower, depending on the instance type. AWS also offers Reserved Instances for cases where customers know they will use most of the instance for a year. You pay a yearly fee per instance and then an hourly rate that is about 30% of column 1 to use it. If you used a Reserved Instance 100% for a whole year, the average cost per hour including amortization of the annual fee would be about 65% of the rate in the first column. The server equivalent to those in Figures 6.13 and 6.14 would be a Standard Extra Large or High-CPU Extra Large Instance, which we calculated to cost \$0.11 per hour.

**Example** Calculate the cost of running the average MapReduce jobs in Figure 6.2 on page 437 on EC2. Assume there are plenty of jobs, so there is no significant extra cost to round up so as to get an integer number of hours. Ignore the monthly storage costs, but include the cost of disk I/Os for AWS’s Elastic Block Storage (EBS). Next calculate the cost per year to run all the MapReduce jobs.

**Answer** The first question is what is the right size instance to match the typical server at Google? Figure 6.21 on page 467 in Section 6.7 shows that in 2007 a typical Google server had four cores running at 2.2 GHz with 8 GB of memory. Since a single instance is one virtual core that is equivalent to a 1 to 1.2 GHz AMD Opteron, the closest match in Figure 6.15 is a High-CPU Extra Large with eight virtual cores and 7.0 GB of memory. For simplicity, we’ll assume the average EBS storage access is 64 KB in order to calculate the number of I/Os.

	Aug-04	Mar-06	Sep-07	Sep-09
Average completion time (hours)	0.15	0.21	0.10	0.11
Average number of servers per job	157	268	394	488
Cost per hour of EC2 High-CPU XL instance	\$0.68	\$0.68	\$0.68	\$0.68
Average EC2 cost per MapReduce job	\$16.35	\$38.47	\$25.56	\$38.07
Average number of EBS I/O requests (millions)	2.34	5.80	3.26	3.19
EBS cost per million I/O requests	\$0.10	\$0.10	\$0.10	\$0.10
Average EBS I/O cost per MapReduce job	\$0.23	\$0.58	\$0.33	\$0.32
Average total cost per MapReduce job	\$16.58	\$39.05	\$25.89	\$38.39
Annual number of MapReduce jobs	29,000	171,000	2,217,000	3,467,000
Total cost of MapReduce jobs on EC2/EBS	\$480,910	\$6,678,011	\$57,394,985	\$133,107,414

**Figure 6.16** Estimated cost if you ran the Google MapReduce workload (Figure 6.2) using 2011 prices for AWS ECS and EBS (Figure 6.15). Since we are using 2011 prices, these estimates are less accurate for earlier years than for the more recent ones.

Figure 6.16 calculates the average and total cost per year of running the Google MapReduce workload on EC2. The average 2009 MapReduce job would cost a little under \$40 on EC2, and the total workload for 2009 would cost \$133M on AWS. Note that EBS accesses are about 1% of total costs for these jobs.

**Example** Given that the costs of MapReduce jobs are growing and already exceed \$100M per year, imagine that your boss wants you to investigate ways to lower costs. Two potentially lower cost options are either AWS Reserved Instances or AWS Spot Instances. Which would you recommend?

**Answer** AWS Reserved Instances charge a fixed annual rate plus an hourly per-use rate. In 2011, the annual cost for the High-CPU Extra Large Instance is \$1820 and the hourly rate is \$0.24. Since we pay for the instances whether they are used or not, let's assume that the average utilization of Reserved Instances is 80%. Then the average price per hour becomes:

$$\frac{\text{Annual price}}{\text{Hours per year} \times \text{Utilization}} + \text{Hourly price} = \frac{\$1820}{8760 \times 80\%} + \$0.24 = (0.21 + 0.24) \times 1.25 = \$0.56$$

Thus, the savings using Reserved Instances would be roughly 17% or \$23M for the 2009 MapReduce workload.

Sampling a few days in January 2011, the hourly cost of a High-CPU Extra Large Spot Instance averages \$0.235. Since that is the minimum price to bid to get one server, that cannot be the average cost since you usually want to run tasks to completion without being bumped. Let's assume you need to pay double the minimum price to run large MapReduce jobs to completion. The cost savings for Spot Instances for the 2009 workload would be roughly 31% or \$41M.



Thus, you tentatively recommend Spot Instances to your boss since there is less of an up-front commitment and they may potentially save more money. However, you tell your boss you need to try to run MapReduce jobs on Spot Instances to see what you actually end up paying to ensure that jobs run to completion and that there really are hundreds of High-CPU Extra Large Instances available to run these jobs daily.

---

In addition to the low cost and a pay-for-use model of utility computing, another strong attractor for cloud computing users is that the cloud computing providers take on the risks of over-provisioning or under-provisioning. Risk avoidance is a godsend for startup companies, as either mistake could be fatal. If too much of the precious investment is spent on servers before the product is ready for heavy use, the company could run out of money. If the service suddenly became popular, but there weren't enough servers to match the demand, the company could make a very bad impression with the potential new customers it desperately needs to grow.

The poster child for this scenario is FarmVille from Zynga, a social networking game on Facebook. Before FarmVille was announced, the largest social game was about 5 million daily players. FarmVille had 1 million players 4 days after launch and 10 million players after 60 days. After 270 days, it had 28 million daily players and 75 million monthly players. Because they were deployed on AWS, they were able to grow seamlessly with the number of users. Moreover, it sheds load based on customer demand.

More established companies are taking advantage of the scalability of the cloud, as well. In 2011, Netflix migrated its Web site and streaming video service from a conventional datacenter to AWS. Netflix's goal was to let users watch a movie on, say, their cell phone while commuting home and then seamlessly switch to their television when they arrive home to continue watching their movie where they left off. This effort involves batch processing to convert new movies to the myriad formats they need to deliver movies on cell phones, tablets, laptops, game consoles, and digital video recorders. These batch AWS jobs can take thousands of machines several weeks to complete the conversions. The transactional backend for streaming is done in AWS and the delivery of encoded files is done via Content Delivery Networks such as Akamai and Level 3. The online service is much less expensive than mailing DVDs, and the resulting low cost has made the new service popular. One study put Netflix as 30% of Internet download traffic in the United States during peak evening periods. (In contrast, YouTube was just 10% in the same 8 p.m. to 10 p.m. period.) In fact, the overall average is 22% of Internet traffic, making Netflix alone responsible for the largest portion of Internet traffic in North America. Despite accelerating growth rates in Netflix subscriber accounts, the growth rate of Netflix's datacenter has been halted, and all capacity expansion going forward has been done via AWS.

Cloud computing has made the benefits of WSC available to everyone. Cloud computing offers cost associativity with the illusion of infinite scalability at no extra cost to the user: 1000 servers for 1 hour cost no more than 1 server for

1000 hours. It is up to the cloud computing provider to ensure that there are enough servers, storage, and Internet bandwidth available to meet the demand. The optimized supply chain mentioned above, which drops time-to-delivery to a week for new computers, is a considerable aid in providing that illusion without bankrupting the provider. This transfer of risks, cost associativity, and pay-as-you-go pricing is a powerful argument for companies of varying sizes to use cloud computing.

Two crosscutting issues that shape the cost-performance of WSCs and hence cloud computing are the WSC network and the efficiency of the server hardware and software.

## 6.6

### Crosscutting Issues

Net gear is the SUV of the datacenter.

James Hamilton (2009)

#### WSC Network as a Bottleneck

Section 6.4 showed that the networking gear above the rack switch is a significant fraction of the cost of a WSC. Fully configured, the list price of a 128-port 1 Gbit datacenter switch from Juniper (EX8216) is \$716,000 without optical interfaces and \$908,000 with them. (These list prices are heavily discounted, but they still cost more than 50 times as much as a rack switch did.) These switches also tend to be power hungry. For example, the EX8216 consumes about 19,200 watts, which is 500 to 1000 times more than a server in a WSC. Moreover, these large switches are manually configured and fragile at a large scale. Because of their price, it is difficult to afford more than dual redundancy in a WSC using these large switches, which limits the options for fault tolerance [Hamilton 2009].

However, the real impact on switches is how oversubscription affects the design of software and the placement of services and data within the WSC. The ideal WSC network would be a black box whose topology and bandwidth are uninteresting because there are no restrictions: You could run any workload in any place and optimize for server utilization rather than network traffic locality. The WSC network bottlenecks today constrain data placement, which in turn complicates WSC software. As this software is one of the most valuable assets of a WSC company, the cost of this added complexity can be significant.

For readers interested in learning more about switch design, Appendix F describes the issues involved in the design of interconnection networks. In addition, Thacker [2007] proposed borrowing networking technology from supercomputing to overcome the price and performance problems. Vahdat et al. [2010] did as well, and proposed a networking infrastructure that can scale to 100,000 ports and 1 petabit/sec of bisection bandwidth. A major benefit of these novel datacenter switches is to simplify the software challenges due to oversubscription.

## Using Energy Efficiently Inside the Server

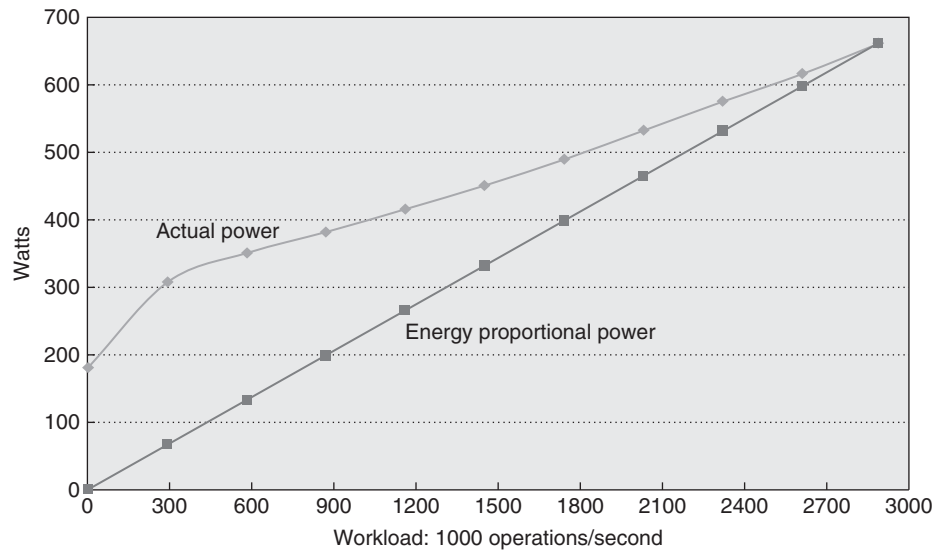
While PUE measures the efficiency of a WSC, it has nothing to say about what goes on inside the IT equipment itself. Thus, another source of electrical inefficiency not covered in Figure 6.9 is the power supply *inside* the server, which converts input of 208 volts or 110 volts to the voltages that chips and disks use, typically 3.3, 5, and 12 volts. The 12 volts are further stepped down to 1.2 to 1.8 volts on the board, depending on what the microprocessor and memory require. In 2007, many power supplies were 60% to 80% efficient, which meant there were greater losses inside the server than there were going through the many steps and voltage changes from the high-voltage lines at the utility tower to supply the low-voltage lines at the server. One reason is that they have to supply a range of voltages to the chips and the disks, since they have no idea what is on the motherboard. A second reason is that the power supply is often oversized in watts for what is on the board. Moreover, such power supplies are often at their worst efficiency at 25% load or less, even though as Figure 6.3 on page 440 shows, many WSC servers operate in that range. Computer motherboards also have voltage regulator modules (VRMs), and they can have relatively low efficiency as well.

To improve the state of the art, Figure 6.17 shows the Climate Savers Computing Initiative standards [2007] for rating power supplies and their goals over time. Note that the standard specifies requirements at 20% and 50% loading in addition to 100% loading.

In addition to the power supply, Barroso and Hölzle [2007] said the goal for the whole server should be *energy proportionality*; that is, servers should consume energy in proportion to the amount of work performed. Figure 6.18 shows how far we are from achieving that ideal goal using SPECpower, a server benchmark that measures energy used at different performance levels (Chapter 1). The energy proportional line is added to the actual power usage of the most efficient server for SPECpower as of July 2010. Most servers will not be that efficient; it was up to 2.5 times better than other systems benchmarked that year, and late in a benchmark competition systems are often configured in ways to win the benchmark that are not typical of systems in the field. For example, the best-rated SPECpower servers use solid-state disks whose capacity is smaller than main memory! Even so, this very efficient system still uses almost 30% of the full

Loading conditioning	Base	Bronze (June 2008)	Silver (June 2009)	Gold (June 2010)
20%	80%	82%	85%	87%
50%	80%	85%	88%	90%
100%	80%	82%	85%	87%

**Figure 6.17** Efficiency ratings and goals for power supplies over time of the Climate Savers Computing Initiative. These ratings are for Multi-Output Power Supply Units, which refer to desktop and server power supplies in nonredundant systems. There is a slightly higher standard for single-output PSUs, which are typically used in redundant configurations (1U/2U single-, dual-, and four-socket and blade servers).



**Figure 6.18** The best SPECpower results as of July 2010 versus the ideal energy proportional behavior. The system was the HP ProLiant SL2x170z G6, which uses a cluster of four dual-socket Intel Xeon L5640s with each socket having six cores running at 2.27 GHz. The system had 64 GB of DRAM and a tiny 60 GB SSD for secondary storage. (The fact that main memory is larger than disk capacity suggests that this system was tailored to this benchmark.) The software used was IBM Java Virtual Machine version 9 and Windows Server 2008, Enterprise Edition.

power when idle and almost 50% of full power at just 10% load. Thus, energy proportionality remains a lofty goal instead of a proud achievement.

Systems software is designed to use all of an available resource if it potentially improves performance, without concern for the energy implications. For example, operating systems use all of memory for program data or for file caches, despite the fact that much of the data will likely never be used. Software architects need to consider energy as well as performance in future designs [Carter and Rajamani 2010].

---

**Example** Using the data of the kind in Figure 6.18, what is the saving in power going from five servers at 10% utilization versus one server at 50% utilization?

**Answer** A single server at 10% load is 308 watts and at 50% load is 451 watts. The savings is then

$$5 \times 308 / 451 = (1540 / 451) \approx 3.4$$

or about a factor of 3.4. If we want to be good environmental stewards in our WSC, we must consolidate servers when utilizations drop, purchase servers that are more energy proportional, or find something else that is useful to run in periods of low activity.

---

Given the background from these six sections, we are now ready to appreciate the work of the Google WSC architects.

## 6.7

## Putting It All Together: A Google Warehouse-Scale Computer

Since many companies with WSCs are competing vigorously in the marketplace, up until recently, they have been reluctant to share their latest innovations with the public (and each other). In 2009, Google described a state-of-the-art WSC as of 2005. Google graciously provided an update of the 2007 status of their WSC, making this section the most up-to-date description of a Google WSC [Clidaras, Johnson, and Felderman 2010]. Even more recently, Facebook described their latest datacenter as part of <http://opencompute.org>.

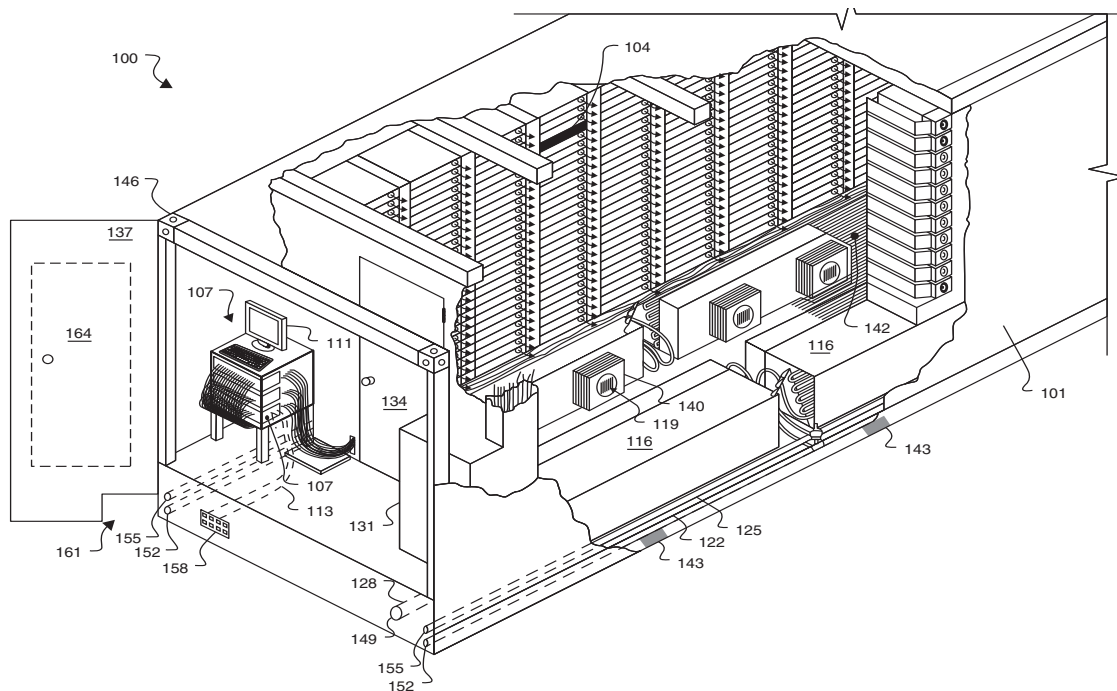
### Containers

Both Google and Microsoft have built WSCs using shipping containers. The idea of building a WSC from containers is to make WSC design modular. Each container is independent, and the only external connections are networking, power, and water. The containers in turn supply networking, power, and cooling to the servers placed inside them, so the job of the WSC is to supply networking, power, and cold water to the containers and to pump the resulting warm water to external cooling towers and chillers.

The Google WSC that we are looking at contains 45 40-foot-long containers in a 300-foot by 250-foot space, or 75,000 square feet (about 7000 square meters). To fit in the warehouse, 30 of the containers are stacked two high, or 15 pairs of stacked containers. Although the location was not revealed, it was built at the time that Google developed WSCs in The Dalles, Oregon, which provides a moderate climate and is near cheap hydroelectric power and Internet backbone fiber. This WSC offers 10 megawatts with a PUE of 1.23 over the prior 12 months. Of that 0.230 of PUE overhead, 85% goes to cooling losses (0.195 PUE) and 15% (0.035) goes to power losses. The system went live in November 2005, and this section describes its state as of 2007.

A Google container can handle up to 250 kilowatts. That means the container can handle 780 watts per square foot (0.09 square meters), or 133 watts per square foot across the entire 75,000-square-foot space with 40 containers. However, the containers in this WSC average just 222 kilowatts

Figure 6.19 is a cutaway drawing of a Google container. A container holds up to 1160 servers, so 45 containers have space for 52,200 servers. (This WSC has about 40,000 servers.) The servers are stacked 20 high in racks that form two long rows of 29 racks (also called *bays*) each, with one row on each side of the container. The rack switches are 48-port, 1 Gbit/sec Ethernet switches, which are placed in every other rack.

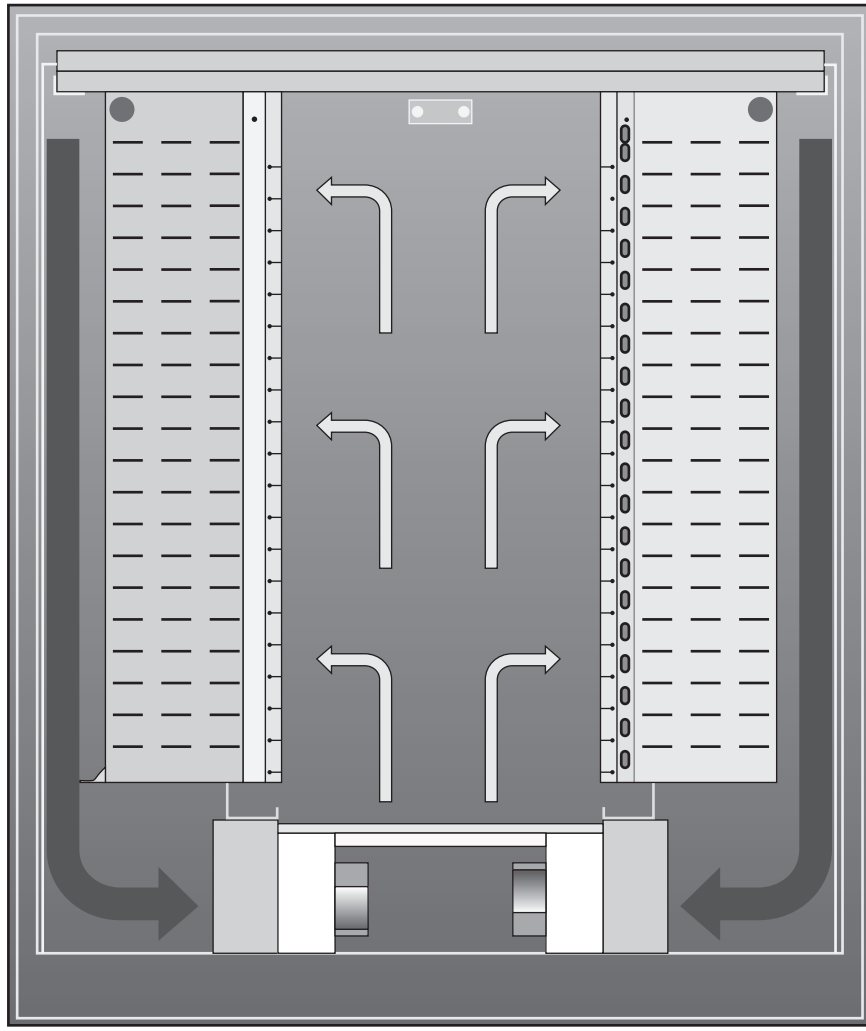


**Figure 6.19** Google customizes a standard 1AAA container: 40 x 8 x 9.5 feet (12.2 x 2.4 x 2.9 meters). The servers are stacked up to 20 high in racks that form two long rows of 29 racks each, with one row on each side of the container. The cool aisle goes down the middle of the container, with the 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.

## Cooling and Power in the Google WSC

Figure 6.20 is a cross-section of the container that shows the airflow. The computer racks are attached to the ceiling of the container. The cooling is below a raised floor that blows into the aisle between the racks. Hot air is returned from behind the racks. The restricted space of the container prevents the mixing of hot and cold air, which improves cooling efficiency. Variable-speed fans are run at the lowest speed needed to cool the rack as opposed to a constant speed.

The “cold” air is kept 81°F (27°C), which is balmy compared to the temperatures in many conventional datacenters. One reason datacenters traditionally run so cold is not for the IT equipment, but so that hot spots within the datacenter don’t cause isolated problems. By carefully controlling airflow to prevent hot spots, the container can run at a much higher temperature.

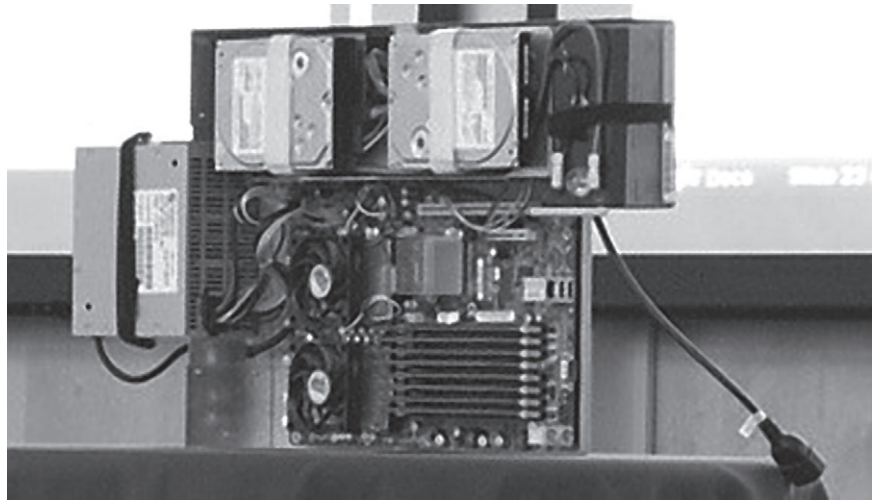


**Figure 6.20** Airflow within the container shown in Figure 6.19. This cross-section diagram shows two racks on each side of the container. Cold air blows into the aisle in the middle of the container and is then sucked into the servers. Warm air returns at the edges of the container. This design isolates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

External chillers have cutouts so that, if the weather is right, only the outdoor cooling towers need cool the water. The chillers are skipped if the temperature of the water leaving the cooling tower is 70°F (21°C) or lower.

Note that if it's too cold outside, the cooling towers need heaters to prevent ice from forming. One of the advantages of placing a WSC in The Dalles is that the annual wet-bulb temperature ranges from 15°F to 66°F (−9°C to 19°C) with an average of 41°F (5°C), so the chillers can often be turned off. In contrast,





**Figure 6.21** Server for Google WSC. The power supply is on the left and the two disks are on the top. The two fans below the left disk cover the two sockets of the AMD Barcelona microprocessor, each with two cores, running at 2.2 GHz. The eight DIMMs in the lower right each hold 1 GB, giving a total of 8 GB. There is no extra sheet metal, as the servers are plugged into the battery and a separate plenum is in the rack for each server to help control the airflow. In part because of the height of the batteries, 20 servers fit in a rack.

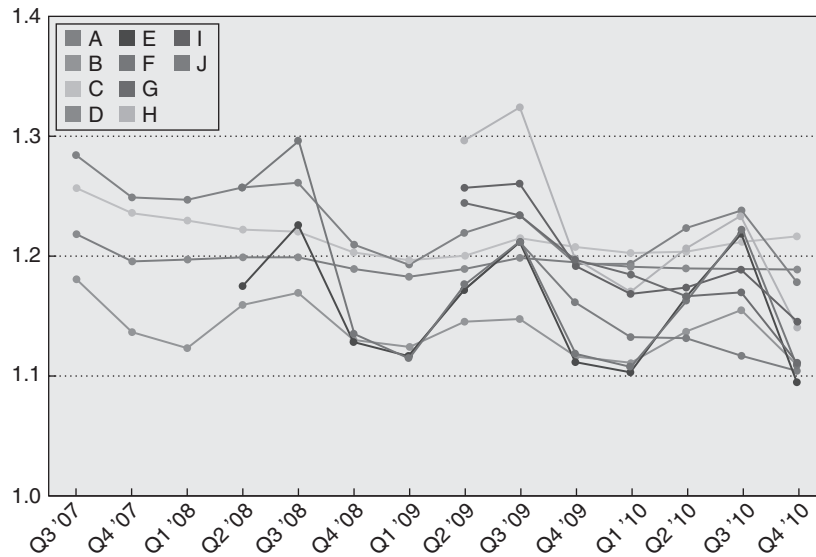
Las Vegas, Nevada, ranges from  $-42^{\circ}\text{F}$  to  $62^{\circ}\text{F}$  ( $-41^{\circ}\text{C}$  to  $17^{\circ}\text{C}$ ) with an average of  $29^{\circ}\text{F}$  ( $-2^{\circ}\text{C}$ ). In addition, having to cool only to  $81^{\circ}\text{F}$  ( $27^{\circ}\text{C}$ ) inside the container makes it much more likely that Mother Nature will be able to cool the water.

Figure 6.21 shows the server designed by Google for this WSC. To improve efficiency of the power supply, it only supplies 12 volts to the motherboard and the motherboard supplies just enough for the number of disks it has on the board. (Laptops power their disks similarly.) The server norm is to supply the many voltage levels needed by the disks and chips directly. This simplification means the 2007 power supply can run at 92% efficiency, going far above the Gold rating for power supplies in 2010 (Figure 6.17).

Google engineers realized that 12 volts meant that the UPS could simply be a standard battery on each shelf. Hence, rather than have a separate battery room, which Figure 6.9 shows as 94% efficient, each server has its own lead acid battery that is 99.99% efficient. This “distributed UPS” is deployed incrementally with each machine, which means there is no money or power spent on overcapacity. They use standard off-the-shelf UPS units to protect network switches.

What about saving power by using dynamic voltage-frequency scaling (DVFS), which Chapter 1 describes? DVFS was not deployed in this family of machines since the impact on latency was such that it was only feasible in very low activity regions for online workloads, and even in those cases the system-wide savings were very small. The complex management control loop needed to deploy it therefore could not be justified.





**Figure 6.22** Power usage effectiveness (PUE) of 10 Google WSCs over time. Google A is the WSC described in this section. It is the highest line in Q3 '07 and Q2 '10. (From [www.google.com/corporate/green/datacenters/measuring.htm](http://www.google.com/corporate/green/datacenters/measuring.htm).) Facebook recently announced a new datacenter that should deliver an impressive PUE of 1.07 (see <http://opencompute.org/>). The Prineville Oregon Facility has no air conditioning and no chilled water. It relies strictly on outside air, which is brought in one side of the building, filtered, cooled via misters, pumped across the IT equipment, and then sent out the building by exhaust fans. In addition, the servers use a custom power supply that allows the power distribution system to skip one of the voltage conversion steps in Figure 6.9.

One of the keys to achieving the PUE of 1.23 was to put measurement devices (called *current transformers*) in all circuits throughout the containers and elsewhere in the WSC to measure the actual power usage. These measurements allowed Google to tune the design of the WSC over time.

Google publishes the PUE of its WSCs each quarter. Figure 6.22 plots the PUE for 10 Google WSCs from the third quarter in 2007 to the second quarter in 2010; this section describes the WSC labeled Google A. Google E operates with a PUE of 1.16 with cooling being only 0.105, due to the higher operational temperatures and chiller cutouts. Power distribution is just 0.039, due to the distributed UPS and single voltage power supply. The best WSC result was 1.12, with Google A at 1.23. In April 2009, the trailing 12-month average weighted by usage across all datacenters was 1.19.

## Servers in a Google WSC

The server in Figure 6.21 has two sockets, each containing a dual-core AMD Opteron processor running at 2.2 GHz. The photo shows eight DIMMS, and

these servers are typically deployed with 8 GB of DDR2 DRAM. A novel feature is that the memory bus is downclocked to 533 MHz from the standard 666 MHz since the slower bus has little impact on performance but a significant impact on power.

The baseline design has a single network interface card (NIC) for a 1 Gbit/sec Ethernet link. Although the photo in [Figure 6.21](#) shows two SATA disk drives, the baseline server has just one. The peak power of the baseline is about 160 watts, and idle power is 85 watts.

This baseline node is supplemented to offer a storage (or “diskfull”) node. First, a second tray containing 10 SATA disks is connected to the server. To get one more disk, a second disk is placed into the empty spot on the motherboard, giving the storage node 12 SATA disks. Finally, since a storage node could saturate a single 1 Gbit/sec Ethernet link, a second Ethernet NIC was added. Peak power for a storage node is about 300 watts, and it idles at 198 watts.

Note that the storage node takes up two slots in the rack, which is one reason why Google deployed 40,000 instead of 52,200 servers in the 45 containers. In this facility, the ratio was about two compute nodes for every storage node, but that ratio varied widely across Google’s WSCs. Hence, Google A had about 190,000 disks in 2007, or an average of almost 5 disks per server.

## Networking in a Google WSC

The 40,000 servers are divided into three arrays of more than 10,000 servers each. (Arrays are called *clusters* in Google terminology.) The 48-port rack switch uses 40 ports to connect to servers, leaving 8 for uplinks to the array switches.

Array switches are configured to support up to 480 1 Gbit/sec Ethernet links and a few 10 Gbit/sec ports. The 1 Gigabit ports are used to connect to the rack switches, as each rack switch has a single link to each of the array switches. The 10 Gbit/sec ports connect to each of two datacenter routers, which aggregate all array routers and provide connectivity to the outside world. The WSC uses two datacenter routers for dependability, so a single datacenter router failure does not take out the whole WSC.

The number of uplink ports used per rack switch varies from a minimum of 2 to a maximum of 8. In the dual-port case, rack switches operate at an oversubscription rate of 20:1. That is, there is 20 times the network bandwidth inside the switch as there was exiting the switch. Applications with significant traffic demands beyond a rack tended to suffer from poor network performance. Hence, the 8-port uplink design, which provided a lower oversubscription rate of just 5:1, was used for arrays with more demanding traffic requirements.

## Monitoring and Repair in a Google WSC

For a single operator to be responsible for more than 1000 servers, you need an extensive monitoring infrastructure and some automation to help with routine events.

Google deploys monitoring software to track the health of all servers and networking gear. Diagnostics are running all the time. When a system fails, many of the possible problems have simple automated solutions. In this case, the next step is to reboot the system and then to try to reinstall software components. Thus, the procedure handles the majority of the failures.

Machines that fail these first steps are added to a queue of machines to be repaired. The diagnosis of the problem is placed into the queue along with the ID of the failed machine.

To amortize the cost of repair, failed machines are addressed in batches by repair technicians. When the diagnosis software is confident in its assessment, the part is immediately replaced without going through the manual diagnosis process. For example, if the diagnostic says disk 3 of a storage node is bad, the disk is replaced immediately. Failed machines with no diagnostic or with low-confidence diagnostics are examined manually.

The goal is to have less than 1% of all nodes in the manual repair queue at any one time. The average time in the repair queue is a week, even though it takes much less time for repair technician to fix it. The longer latency suggests the importance of repair throughput, which affects cost of operations. Note that the automated repairs of the first step take minutes for a reboot/reinstall to hours for running directed stress tests to make sure the machine is indeed operational.

These latencies do not take into account the time to idle the broken servers. The reason is that a big variable is the amount of state in the node. A stateless node takes much less time than a storage node whose data may need to be evacuated before it can be replaced.

## Summary

As of 2007, Google had already demonstrated several innovations to improve the energy efficiency of its WSCs to deliver a PUE of 1.23 in Google A:

- In addition to providing an inexpensive shell to enclose servers, the modified shipping containers separate hot and cold air plenums, which helps reduce the variation in intake air temperature for servers. With less severe worst-case hot spots, cold air can be delivered at warmer temperatures.
- These containers also shrink the distance of the air circulation loop, which reduces energy to move air.
- Operating servers at higher temperatures means that air only has to be chilled to 81°F (27°C) instead of the traditional 64°F to 71°F (18°C to 22°C).
- A higher target cold air temperature helps put the facility more often within the range that can be sustained by evaporative cooling solutions (cooling towers), which are more energy efficient than traditional chillers.
- Deploying WSCs in temperate climates to allow use of evaporative cooling exclusively for portions of the year.
- Deploying extensive monitoring hardware and software to measure actual PUE versus designed PUE improves operational efficiency.

- Operating more servers than the worst-case scenario for the power distribution system would suggest, since it's statistically unlikely that thousands of servers would all be highly busy simultaneously, yet rely on the monitoring system to off-load work in the unlikely case that they did [Fan, Weber, and Barroso 2007] [Ranganathan et al. 2006]. PUE improves because the facility is operating closer to its fully designed capacity, where it is at its most efficient because the servers and cooling systems are not energy proportional. Such increased utilization reduces demand for new servers and new WSCs.
- Designing motherboards that only need a single 12-volt supply so that the UPS function could be supplied by standard batteries associated with each server instead of a battery room, thereby lowering costs and reducing one source of inefficiency of power distribution within a WSC.
- Carefully designing the server board itself to improve its energy efficiency. For example, underclocking the front-side bus on these microprocessors reduces energy usage with negligible performance impact. (Note that such optimizations do not impact PUE but do reduce overall WSC energy consumption.)

WSC design must have improved in the intervening years, as Google's best WSC has dropped the PUE from 1.23 for Google A to 1.12. Facebook announced in 2011 that they had driven PUE down to 1.07 in their new datacenter (see <http://opencompute.org/>). It will be interesting to see what innovations remain to improve further the WSC efficiency so that we are good guardians of our environment. Perhaps in the future we will even consider the energy cost to *manufacture* the equipment within a WSC [Chang et al. 2010].

## 6.8

## Fallacies and Pitfalls

Despite WSC being less than a decade old, WSC architects like those at Google have already uncovered many pitfalls and fallacies about WSCs, often learned the hard way. As we said in the introduction, WSC architects are today's Seymour Crays.

**Fallacy** *Cloud computing providers are losing money.*

A popular question about cloud computing is whether it's profitable at these low prices.

Based on AWS pricing from [Figure 6.15](#), we could charge \$0.68 per hour per server for computation. (The \$0.085 per hour price is for a Virtual Machine equivalent to one EC2 compute unit, not a full server.) If we could sell 50% of the server hours, that would generate \$0.34 of income per hour per server. (Note that customers pay no matter how little they use the servers they occupy, so selling 50% of the server hours doesn't necessarily mean that average server utilization is 50%.)

Another way to calculate income would be to use AWS *Reserved Instances*, where customers pay a yearly fee to reserve an instance and then a lower rate per

hour to use it. Combining the charges together, AWS would receive \$0.45 of income per hour per server for a full year.

If we could sell 750 GB per server for storage using AWS pricing, in addition to the computation income, that would generate another \$75 per month per server, or another \$0.10 per hour.

These numbers suggest an average income of \$0.44 per hour per server (via On-Demand Instances) to \$0.55 per hour (via Reserved Instances). From Figure 6.13, we calculated the cost per server as \$0.11 per hour for the WSC in Section 6.4. Although the costs in Figure 6.13 are estimates that are *not* based on actual AWS costs and the 50% sales for server processing and 750 GB utilization of per server storage are just examples, these assumptions suggest a gross margin of 75% to 80%. Assuming these calculations are reasonable, they suggest that cloud computing is profitable, especially for a service business.

**Fallacy** *Capital costs of the WSC facility are higher than for the servers that it houses.*

While a quick look at Figure 6.13 on page 453 might lead you to that conclusion, that glimpse ignores the length of amortization for each part of the full WSC. However, the facility lasts 10 to 15 years while the servers need to be repurchased every 3 or 4 years. Using the amortization times in Figure 6.13 of 10 years and 3 years, respectively, the capital expenditures over a decade are \$72M for the facility and  $3.3 \times \$67M$ , or \$221M, for servers. Thus, the capital costs for servers in a WSC over a decade are a factor of three higher than for the WSC facility.

**Pitfall** *Trying to save power with inactive low power modes versus active low power modes.*

Figure 6.3 on page 440 shows that the average utilization of servers is between 10% and 50%. Given the concern on operational costs of a WSC from Section 6.4, you would think low power modes would be a huge help.

As Chapter 1 mentions, you cannot access DRAMs or disks in these *inactive low power modes*, so you must return to fully active mode to read or write, no matter how low the rate. The pitfall is that the time and energy required to return to fully active mode make inactive low power modes less attractive. Figure 6.3 shows that almost all servers average at least 10% utilization, so you might expect long periods of low activity but not long periods of inactivity.

In contrast, processors still run in lower power modes at a small multiple of the regular rate, so *active low power modes* are much easier to use. Note that the time to move to fully active mode for processors is also measured in microseconds, so active low power modes also address the latency concerns about low power modes.

**Pitfall** *Using too wimpy a processor when trying to improve WSC cost-performance.*

Amdahl's law still applies to WSC, as there will be some serial work for each request, and that can increase request latency if it runs on a slow server [Hölzle 2010] [Lim et al. 2008]. If the serial work increases latency, then the cost of using a wimpy processor must include the software development costs to optimize the

code to return it to the lower latency. The larger number of threads of many slow servers can also be more difficult to schedule and load balance, and thus the variability in thread performance can lead to longer latencies. A 1 in 1000 chance of bad scheduling is probably not an issue with 10 tasks, but it is with 1000 tasks when you have to wait for the longest task. Many smaller servers can also lead to lower utilization, as it's clearly easier to schedule when there are fewer things to schedule. Finally, even some parallel algorithms get less efficient when the problem is partitioned too finely. The Google rule of thumb is currently to use the low-end range of server class computers [Barroso and Hölzle 2009].

As a concrete example, Reddi et al. [2010] compared embedded microprocessors (Atom) and server microprocessors (Nehalem Xeon) running the Bing search engine. They found that the latency of a query was about three times longer on Atom than on Xeon. Moreover, the Xeon was more robust. As load increases on Xeon, quality of service degrades gradually and modestly. Atom quickly violates its quality-of-service target as it tries to absorb additional load.

This behavior translates directly into search quality. Given the importance of latency to the user, as Figure 6.12 suggests, the Bing search engine uses multiple strategies to refine search results if the query latency has not yet exceeded a cut-off latency. The lower latency of the larger Xeon nodes means they can spend more time refining search results. Hence, even when the Atom had almost no load, it gave worse answers in 1% of the queries than Xeon. At normal loads, 2% of the answers were worse.

**Fallacy** *Given improvements in DRAM dependability and the fault tolerance of WSC systems software, you don't need to spend extra for ECC memory in a WSC.*

Since ECC adds 8 bits to every 64 bits of DRAM, potentially you could save a ninth of the DRAM costs by eliminating error-correcting code (ECC), especially since measurements of DRAM had claimed failure rates of 1000 to 5000 FIT (failures per billion hours of operation) per megabit [Tezzaron Semiconductor 2004].

Schroeder, Pinheiro, and Weber [2009] studied measurements of the DRAMs with ECC protection at the majority of Google's WSCs, which was surely many hundreds of thousands of servers, over a 2.5-year period. They found 15 to 25 times higher FIT rates than had been published, or 25,000 to 70,000 failures per megabit. Failures affected more than 8% of DIMMs, and the average DIMM had 4000 correctable errors and 0.2 uncorrectable errors per year. Measured at the server, about a third experienced DRAM errors each year, with an average of 22,000 correctable errors and 1 uncorrectable error per year. That is, for one-third of the servers, one memory error is corrected every 2.5 hours. Note that these systems used the more powerful chipkill codes rather than the simpler SECDED codes. If the simpler scheme had been used, the uncorrectable error rates would have been 4 to 10 times higher.

In a WSC that only had parity error protection, the servers would have to reboot for each memory parity error. If the reboot time were 5 minutes, one-third of the machines would spend 20% of their time rebooting! Such behavior would

lower the performance of the \$150M facility by about 6%. Moreover, these systems would suffer many uncorrectable errors without operators being notified that they occurred.

In the early years, Google used DRAM that didn't even have parity protection. In 2000, during testing before shipping the next release of the search index, it started suggesting random documents in response to test queries [Barroso and Hölzle 2009]. The reason was a stuck-at-zero fault in some DRAMs, which corrupted the new index. Google added consistency checks to detect such errors in the future. As WSC grew in size and as ECC DIMMs became more affordable, ECC became the standard in Google WSCs. ECC has the added benefit of making it much easier to find broken DIMMs during repair.

Such data suggest why the Fermi GPU (Chapter 4) adds ECC to its memory where its predecessors didn't even have parity protection. Moreover, these FIT rates cast doubts on efforts to use the Intel Atom processor in a WSC—due to its improved power efficiency—since the 2011 chip set does not support ECC DRAM.

**Fallacy** *Turning off hardware during periods of low activity improves cost-performance of a WSC.*

Figure 6.14 on page 454 shows that the cost of amortizing the power distribution and cooling infrastructure is 50% higher than the entire monthly power bill. Hence, while it certainly would save some money to compact workloads and turn off idle machines, even if you could save half the power it would only reduce the monthly operational bill by 7%. There would also be practical problems to overcome, since the extensive WSC monitoring infrastructure depends on being able to poke equipment and see it respond. Another advantage of energy proportionality and active low power modes is that they are compatible with the WSC monitoring infrastructure, which allows a single operator to be responsible for more than 1000 servers.

The conventional WSC wisdom is to run other valuable tasks during periods of low activity so as to recoup the investment in power distribution and cooling. A prime example is the batch MapReduce jobs that create indices for search. Another example of getting value from low utilization is spot pricing on AWS, which the caption in Figure 6.15 on page 458 describes. AWS users who are flexible about when their tasks are run can save a factor of 2.7 to 3 for computation by letting AWS schedule the tasks more flexibly using Spot Instances, such as when the WSC would otherwise have low utilization.

**Fallacy** *Replacing all disks with Flash memory will improve cost-performance of a WSC.*

Flash memory is much faster than disk for some WSC workloads, such as those doing many random reads and writes. For example, Facebook deployed Flash memory packaged as solid-state disks (SSDs) as a write-back cache called Flash-cache as part of its file system in its WSC, so that hot files stay in Flash and cold files stay on disk. However, since all performance improvements in a WSC must



be judged on cost-performance, before replacing all the disks with SSD the question is really I/Os per second per dollar and storage capacity per dollar. As we saw in [Chapter 2](#), Flash memory costs at least 20 times more per GByte than magnetic disks: \$2.00/GByte versus \$0.09/Gbyte.

Narayanan et al. [2009] looked at migrating workloads from disk to SSD by simulating workload traces from small and large datacenters. Their conclusion was that SSDs were not cost effective for any of their workloads due to the low storage capacity per dollar. To reach the break-even point, Flash memory storage devices need to improve capacity per dollar by a factor of 3 to 3000, depending on the workload.

Even when you factor power into the equation, it's hard to justify replacing disk with Flash for data that are infrequently accessed. A one-terabyte disk uses about 10 watts of power, so, using the \$2 per watt-year rule of thumb from [Section 6.4](#), the most you could save from reduced energy is \$20 a year per disk. However, the CAPEX cost in 2011 for a terabyte of storage is \$2000 for Flash and only \$90 for disk.

---

## 6.9

## Concluding Remarks

Inheriting the title of building the world's biggest computers, computer architects of WSCs are designing the large part of the future IT that completes the mobile client. Many of us use WSCs many times a day, and the number of times per day and the number of people using WSCs will surely increase in the next decade. Already more than half of the nearly seven billion people on the planet have cell phones. As these devices become Internet ready, many more people from around the world will be able to benefit from WSCs.

Moreover, the economies of scale uncovered by WSC have realized the long dreamed of goal of computing as a utility. Cloud computing means anyone anywhere with good ideas and business models can tap thousands of servers to deliver their vision almost instantly. Of course, there are important obstacles that could limit the growth of cloud computing around standards, privacy, and the rate of growth of Internet bandwidth, but we foresee them being addressed so that cloud computing can flourish.

Given the increasing number of cores per chip (see [Chapter 5](#)), clusters will increase to include thousands of cores. We believe the technologies developed to run WSC will prove useful and trickle down to clusters, so that clusters will run the same virtual machines and systems software developed for WSC. One advantage would be easy support of "hybrid" datacenters, where the workload could easily be shipped to the cloud in a crunch and then shrink back afterwards to relying only on local computing.

Among the many attractive features of cloud computing is that it offers economic incentives for conservation. Whereas it is hard to convince cloud computing *providers* to turn off unused equipment to save energy given the



cost of the infrastructure investment, it is easy to convince cloud computing *users* to give up idle instances since they are paying for them whether or not they are doing anything useful. Similarly, charging by use encourages programmers to use computation, communication, and storage efficiently, which can be difficult to encourage without an understandable pricing scheme. The explicit pricing also makes it possible for researchers to evaluate innovations in cost-performance instead of just performance, since costs are now easily measured and believable. Finally, cloud computing means that researchers can evaluate their ideas at the scale of thousands of computers, which in the past only large companies could afford.

We believe that WSCs are changing the goals and principles of server design, just as the needs of mobile clients are changing the goals and principles of micro-processor design. Both are revolutionizing the software industry, as well. Performance per dollar and performance per joule drive both mobile client hardware and the WSC hardware, and parallelism is the key to delivering on those sets of goals.

Architects will play a vital role in both halves of this exciting future world. We look forward to seeing—and to using—what will come.

---

## 6.10

### Historical Perspectives and References

Section L.8 (available online) covers the development of clusters that were the foundation of WSC and of utility computing. (Readers interested in learning more should start with Barroso and Hölzle [2009] and the blog postings and talks of James Hamilton at <http://perspectives.mvdirona.com>.)

---

### Case Studies and Exercises by Parthasarathy Ranganathan

#### Case Study 1: Total Cost of Ownership Influencing Warehouse-Scale Computer Design Decisions

*Concepts illustrated by this case study*

- Total Cost of Ownership (TCO)
- Influence of Server Cost and Power on the Entire WSC
- Benefits and Drawbacks of Low-Power Servers

Total cost of ownership is an important metric for measuring the effectiveness of a warehouse-scale computer (WSC). TCO includes both the CAPEX and OPEX described in [Section 6.4](#) and reflects the ownership cost of the entire datacenter to achieve a certain level of performance. In considering different servers, networks, and storage architectures, TCO is often the important comparison metric