# The Case for Cloud Computing

**Robert L. Grossman**
*University of Illinois at Chicago and Open Data Group*

To understand clouds and cloud computing, we must first understand the two different types of clouds. The author distinguishes between clouds that provide on-demand computing instances and those that provide on-demand computing capacity.

Cloud computing doesn't yet have a standard definition, but a good working description of it is to say that *clouds*, or clusters of distributed computers, provide on-demand resources and services over a network, usually the Internet, with the scale and reliability of a data center. This article gives a quick introduction to cloud computing. It covers several different types of clouds, describes what's new about cloud computing, and discusses some of the advantages and disadvantages that clouds offer.

Two different but related types of clouds are those that provide computing *instances* on demand and those that provide computing *capacity* on demand. Both use similar machines, but the first is designed to scale out by providing additional computing instances, whereas the second is designed to support data- or compute-intensive applications via scaling capacity.

Amazon's EC2 services (www.amazon.com/ec2) are an example of the first category. A small EC2 computing instance costs US$0.10 per hour and offers the approximate computing power of a 1.0- to 1.2-GHz 2007 Opteron or 2007 Xeon processor, with 1.7 Gbytes memory, 160 Gbytes of available disk space, and moderate I/O performance. The Eucalyptus system (http://eucalyptus.cs.ucsb.edu) is an open source option that provides on-demand computing instances and shares the same APIs as Amazon's EC2 cloud.

Google's MapReduce is an example of the second category. Recent work gives a glimpse of it in action—in this case, researchers ran a benchmark on a cluster containing approximately 1,800 machines,[1] each of which had two 2-GHz Intel Xeon processors, 4 Gbytes memory, and two 160-Gbyte IDE disks. The researchers used MapReduce on the cluster to run the TeraSort benchmark (http://research.microsoft.com/barc/SortBenchmark), the goal of which is to sort $10^{10}$ 100-byte records (roughly 1 Tbyte of data). The application required approximately 850 seconds to complete on this cluster. The Hadoop system (http://hadoop.apache.org/core) is an open source system that implements MapReduce.

Clouds that provide on-demand computing instances can use these instances to supply software as a services (SaaS), such as Salesforce.com does with its product, or to provide a platform as a service (PaaS), such as Amazon's does with its EC2 product.

1520-9202/09/$25.00 © 2009 IEEE Published by the IEEE Computer Society

## What's New?

Now that we've covered the basics of cloud computing, it's important to understand what's new about it. On-demand services and resources have been available over the Internet for some time, but today's increased focus on cloud computing is due to three important differences:

- *Scale*. Some companies that rely on cloud computing have infrastructures that scale over several (or more) data centers.
- *Simplicity*. Prior to cloud-based computing services, writing code for high-performance and distributed computing was relatively complicated and usually required working with grid-based services, developing code that explicitly passed messages between nodes, and employing other specialized methods. Although simplicity is in the eye of the beholder, most people feel that the cloud-based storage service APIs and MapReduce-style computing APIs are relatively simple compared to previous methods.
- *Pricing*. Cloud computing is often offered with a pricing model that lets you pay as you go and for just the services that you need. For example, if you need an additional 1,000 computing instances for an hour, you pay just for these 1,000 computing instances and just for the hour that you use them. No capital expenditure is required.

The impact has been revolutionary—by using the Google File System (GFS) and MapReduce, or the Hadoop Distributed File System with its implementation of MapReduce, it's relatively easy for a project to perform a computation over 10 Tbytes of data using 1,000 nodes. Until recently, this would have been out of reach for most projects. Several other changes and improvements have raised cloud computing's profile as well.

### Private vs. Hosted Clouds

The management, cost, and security of clouds depend on whether an organization chooses to buy and operate its own cloud or to obtain cloud services from a third party. A *private cloud* is devoted to a single organization's internal use; it might be run by the organization itself or outsourced to a third party to operate. Similarly, a private cloud might be owned by the organization itself or leased by the organization. In contrast, a *public* or *hosted cloud* is managed by another organization that provides cloud services to a variety of third-party clients using the same cloud resources. Google, for example, uses GFS,[2] MapReduce,[3] and BigTable[4] internally as part of its private cloud services; at the time of this writing, these services weren't available to third parties. In contrast, hosted cloud services such as Amazon's EC2, S3, and SimpleDB are open to anyone with a credit card, even at 3 a.m.

It's important to note that Google uses its private cloud to provide hosted-cloud-based applications, such as its email and office-based services, to regular outside users.

### Elastic, Usage-Based Pricing

Cloud computing is usually offered with a usage-based model in which you pay for just the cloud resources that a particular computation requires. Computations that require additional resources simply request them from the cloud (up to the cloud's overall capacity). Sometimes, the terms *elastic* or *utility computing* are used to describe this ability of a cloud to provide additional resources when required. Amazon's S3 and EC2 use this pricing model.

Organizations, therefore, have several options for obtaining cloud services, including running their own private clouds or buying cloud services from a third party using the elastic, usage-based pricing model. This type of pricing offers two other important advantages as well:

- It doesn't require up-front investments; instead, as an on-demand service, users pay for capacity as they need it.
- It lets users access capacity exactly when they need it. For Web 2.0 applications, this means that the model can support 100 users one day and 10,000 the next.

To get a better understanding of utility computing, let's assume that you have a requirement to operate 100 servers over the course of three years. One option is to lease them at $0.40 per instance-hour, which would cost approximately

> 100 servers * $0.40 instance-hour * 3 years * 8,760 hours/year = $1,051,200.

Another option is to buy them. Let's assume the cost to buy each server is $1,500, that you need

two staff members at $100,000 per year to administer the servers, and that the servers require 150 watts each, with the cost of electricity at $0.10 per kilowatt-hour, bringing the yearly cost to operate the 100 servers to $13,140. This option would cost approximately

> 100 servers * $1,500 + 3 years * $13,140 electricity/year + 3 years * 2 staff * $100,000 salary/year = $789,420.

So, if you were to run the servers at 100 percent utilization, buying the 100 servers is less expensive. However, if you were to run them at 75 percent utilization or less, using an on-demand style of cloud would be less expensive.

Of courses, these numbers are only estimates, and I haven't considered all costs, but even from this simple example, it's clear that using a pay-as-you-go utility computing model is preferable for many use cases.

### Some Advantages and Disadvantages

Cloud computing provides several important benefits over today's dominant model in which an enterprise purchases computers a rack at a time and operates them themselves. First, cloud computing's usage-based pricing model offers several advantages, including reduced capital expense, a low barrier to entry, and the ability to scale up as demand requires, as well as to support brief surges in capacity. Second, cloud services enjoy the same economies of scale that data centers provide. By providing services at the scale of a data center, it's possible to provide operations, business continuity, and security more efficiently than can be done when providing these services a rack at a time. For this reason, the unit cost for cloud-based services is often lower than the cost if the services were provided directly by the organization itself. Finally, cloud-computing architectures have proven to be very scalable—for example, cloud-based storage services can easily manage a petabyte of data, whereas managing this much data with a traditional database is problematic.

Of course, cloud computing has some disadvantages as well. First, because cloud services are often remote (at least for hosted cloud services), they can suffer the latency- and bandwidth-related issues associated with any remote application. Second, because hosted cloud services serve multiple customers, various issues related to multiple customers sharing the same piece of hardware can arise. For example, if one user's application compromises the system, it can also compromise applications of other users that share the same system. Also, having data accessible to third parties (such as a cloud service provider) can present security, compliance, and regulatory issues.

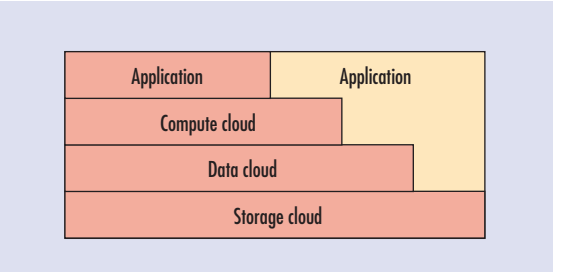


**Figure 1. Layered model. Some clouds that provide on-demand computing capacity use layers of services, forming a stack of cloud services.**

### Layered Services

A *storage cloud* provides storage services (block- or file-based); a *data cloud* provides data management services (record-, column-, or object-based); and a *compute cloud* provides computational services. Often, they're layered (compute services over data services over storage services) to create a stack of cloud services that acts as a computing platform for developing cloud-based applications; see Figure 1.

## Parallel Computing over Clouds

At its core, MapReduce is a style of parallel programming supported by capacity-on-demand clouds. A good illustrating example of how something like MapReduce works is to compute an inverted index in parallel for a large collection of Web pages stored in a cloud.

Let's assume that each node $i$ in the cloud stores Web pages $p_{i,1}$, $p_{i,2}$, $p_{i,3}$, …, and that a Web page $p_j$ contains words (terms) $w_{j,1}$, $w_{j,2}$, $w_{j,3}$, … . A basic but important structure in information retrieval is an inverted index, that is, a list

$$(w_1; p_{1,1}, p_{1,2}, p_{1,3}, \ldots)$$

$$(w_2; p_{2,1}, p_{2,2}, p_{2,3}, \ldots)$$

$$(w_3; p_{3,1}, p_{3,2}, p_{3,3}, \ldots),$$

where the list is sorted by the word $w_j$, and associated with each word $w_j$ is a list of all Web pages $p_i$ containing that word.

MapReduce uses a programming model that processes a list of <key, value> pairs to produce another list of <key', value'> pairs. The initial list of <key, value> pairs is distributed over the nodes in the cloud. In the *map* phase, each Web page $p_i$ is processed independently on its local node to produce an output list of multiple key-value pairs $<w_j, p_i>$, one for word $w_j$ on the page. A partition function $h(w_j)$ then assigns each key (a word $w_j$ in this example) a machine in the cloud for further processing. This is called the *shuffle* phase and, in general, nodes involved in the computation send data to other nodes involved in the computation as determined by the partition function $h(w_j)$. In the next phase—called the *sort* phase—each node in the cloud sorts the key-value pairs $<w_j, p_i>$ according to the key $w_j$. In the final phase—called the *reduce* phase—the key-value pairs with the same key $w_j$ are merged to create the inverted index. So with MapReduce, the programmer defines a Map and a Reduce function, whereas the system supplies the Shuffle and Sort functions.[3]

> Third parties can take advantage of economies of scale to provide a level of security that might not be cost-effective for smaller companies.

Let's consider another example: log files that describe an entity's usage of resources. It's important to analyze log files to identify anomalies that indicate whether a particular resource has been compromised. For small log files, this is easy to do with a database, but, as the size of the log files grows, it's difficult to manage them with just a database. However, clouds can easily manage even very large collections of log files, and MapReduce-style computations can easily identify anomalous patterns indicative of compromises.

## Security

Security is an area of cloud computing that presents some special challenges. For hosted clouds, the first challenge is simply that a third party is responsible both for storing the data and securing it. On the positive side, third parties can take advantage of economies of scale to provide a level of security that might not be cost-effective for smaller companies, but a downside is two or more organizations might share the same physical resource and not be aware of it.

For some cloud applications, security is still somewhat immature. Hadoop, for example, doesn't currently have user-level authentication or access controls, although both are expected in a later version. Fortunately, there's no technical difficulty per se in providing these tools for clouds. Sector,[5] which also provides on-demand computing capacity, offers authentication, authorization, and access controls and, as measured by the TeraSort benchmark, is faster than Hadoop (http://sector.sourceforge.net).

## Standards, Interoperability, and Benchmarks

Organizations that develop cloud-based applications have an interest in frameworks that enable applications to be ported easily from one cloud to another and to interoperate with different cloud-based services. For example, with an appropriate interoperability framework, a cloud application could switch from one provider to another offering lower cost or a greater range of cloud services.

Amazon's APIs (www.aws.amazon.com) have become the de facto standard for clouds that provide on-demand instances. Cloud-based applications that use this API enjoy portability and interoperability—for example, Eucalyptus uses these APIs, and applications that run on Amazon's EC2 service can in turn run on a Eucalyptus cloud. Unfortunately, for clouds that provide on-demand capacity, portability and interoperability are much more problematic. Hadoop is by far the most prevalent system that provides on-demand capacity, but, for instance, it isn't straightforward for a Hadoop MapReduce application to run on another on-demand capacity cloud written in C++.[5]

Although it might be too early yet for standards to fully emerge, several organizations are attempting them, including an effort by the Cloud Computing Interoperability Forum (www.cloudforum.org/) and by the Open Cloud Consortium (www.opencloudconsortium.org). Service-based frameworks for clouds have also recently debuted—for example, Thrift is a software framework for scalable cross-language services development that relies on a code-generation engine (http://incubator.apache.org/thrift). Thrift makes

it easier for cloud-based applications to access different storage clouds, such as Hadoop and Sector. A common language could also help by providing an interoperable way for applications to access compute services across several different clouds; so far, several people have attempted to provide a language for MapReduce-style parallel programming, including some that extend SQL in a way that supports this style of programming, but no single language has emerged as the clear winner yet.

A closely related challenge is creating a standard that would enable different clouds to interoperate. Perhaps the Internet's infancy could guide this type of effort—at that time, any organization that wanted a network set up its own, so sending data between networks was quite difficult. The introduction of TCP and related Internet protocols and standards remedied this situation, but many companies with network products resisted them for some time. Today, we're in a somewhat analogous position: although cloud service providers have pushed back a bit on standardizing, the ability of different clouds to interoperate easily would enable an interesting new class of applications.

As with standards and a common language, cloud computing doesn't yet have well-established benchmarks. The most common method for measuring cloud performance to date is the TeraSort benchmark. For clouds that provide on-demand instances, a recent benchmark called Cloudstone has emerged.[6] Cloudstone is a toolkit consisting of an open source Web 2.0 social application, a set of tools for generating workloads, a set of tools for performance monitoring, and a recommended methodology for computing a metric that quantifies the dollars per user per month that a given cloud requires. For clouds that provide on-demand capacity, a recent benchmark called MalStone (code.google.com/p/malgen/) has emerged as well. MalStone is based on the log file example of a MapReduce computation I described earlier. It includes code to generate synthetic events and a recommended MapReduce computation.

With cloud computing, the "unit of computing" has moved from a single computer or rack of computers to a data center of computers. To say it simply, the unit of computing is now the data center. Not only has cloud computing scaled computing to the data center, but it has also introduced software, systems, and programming models that significantly reduce the complexity of accessing and using these resources. Just as significant, with elastic, usage-based pricing models, an individual or organization pays for just those computing instances or computing capacity that it requires and only when it requires them. Truly, this is revolutionary. IT

## References


1. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Comm. ACM*, vol. 51, no. 1, 2008, pp. 107–113.
2. S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google File System," *Proc. 19th ACM Symp. Operating Systems Principles*, ACM Press, 2003, pp. 29–43.
3. J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Proc. 6th Symp. Operating System Design and Implementation*, Usenix Assoc., 2004, pp. 137–150.
4. F. Chang et al., "Bigtable: A Distributed Storage System for Structured Data," *Proc. 7th Symp. Operating System Design and Implementation*, Usenix Assoc., 2006, pp. 205–218.
5. R.L. Grossman and Y. Gu, "Data Mining Using High-Performance Clouds: Experimental Studies Using Sector and Sphere," *Proc. 14th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining*, ACM Press, 2008, pp. 920–927.
6. W. Sobel et al., "Cloudstone: Multi-Platform Multi-Language Benchmark and Measurement Tools for Web 2.0," *Proc. Cloud Computing and Its Applications*, 2008; www.cca08.org/papers.php.



***Robert L. Grossman*** *is a professor of mathematics, statistics, and computer science at the University of Illinois at Chicago and managing partner at Open Data Group. His research interests include data mining and analytics, distributed computing, and high-performance networking. Grossman has a PhD in applied mathematics from Princeton University. Contact him at grossman at uic.edu.*