DOI:10.1145/1721654.1721672

Clearing the clouds away from the true potential and obstacles posed by this computing capability.

BY MICHAEL ARMBRUST, ARMANDO FOX, REAN GRIFFITH, ANTHONY D. JOSEPH, RANDY KATZ, ANDY KONWINSKI, GUNHO LEE, DAVID PATTERSON, ARIEL RABKIN, ION STOICA, AND MATEI ZAHARIA

A View of Cloud Computing

CLOUD COMPUTING, THE long-held dream of computing as a utility, has the potential to transform a large part of the IT industry, making software even more attractive as a service and shaping the way IT hardware is designed and purchased. Developers with innovative ideas for new Internet services no longer require the large capital outlays in hardware to deploy their service or the human expense to operate it. They need not be concerned about overprovisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or underprovisioning for one that becomes wildly popular, thus missing potential customers and revenue. Moreover, companies with large batch-oriented tasks can get results as quickly as their programs can scale, since using 1,000 servers for one hour costs no more than using one server for 1,000 hours. This elasticity of resources, without paying a premium for large scale, is unprecedented in the history of IT.

As a result, cloud computing is a popular topic for blogging and white papers and has been featured in the title of workshops, conferences, and even magazines. Nevertheless, confusion remains about exactly what it is and when it's useful, causing Oracle's CEO Larry Ellison to vent his frustration: "The interesting thing about cloud computing is that we've redefined cloud computing to include everything that we already do.... I don't understand what we would do differently in the light of cloud computing other than change the wording of some of our ads."

Our goal in this article is to reduce that confusion by clarifying terms, providing simple figures to quantify comparisons between of cloud and conventional computing, and identifying the top technical and non-technical obstacles and opportunities of cloud computing. (Armbrust et al[4] is a more detailed version of this article.)

Defining Cloud Computing

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services. The services themselves have long been referred to as Software as a Service (SaaS).[a] Some vendors use terms such as IaaS (Infrastructure as a Service) and PaaS (Platform as a Service) to describe their products, but we eschew these because accepted definitions for them still vary widely. The line between "low-level" infrastructure and a higher-level "platform" is not crisp. We believe the two are more alike than different, and we consider them together. Similarly, the

a For the purposes of this article, we use the term Software as a Service to mean applications delivered over the Internet. The broadest definition would encompass any on demand software, including those that run software locally but control use via remote software licensing.

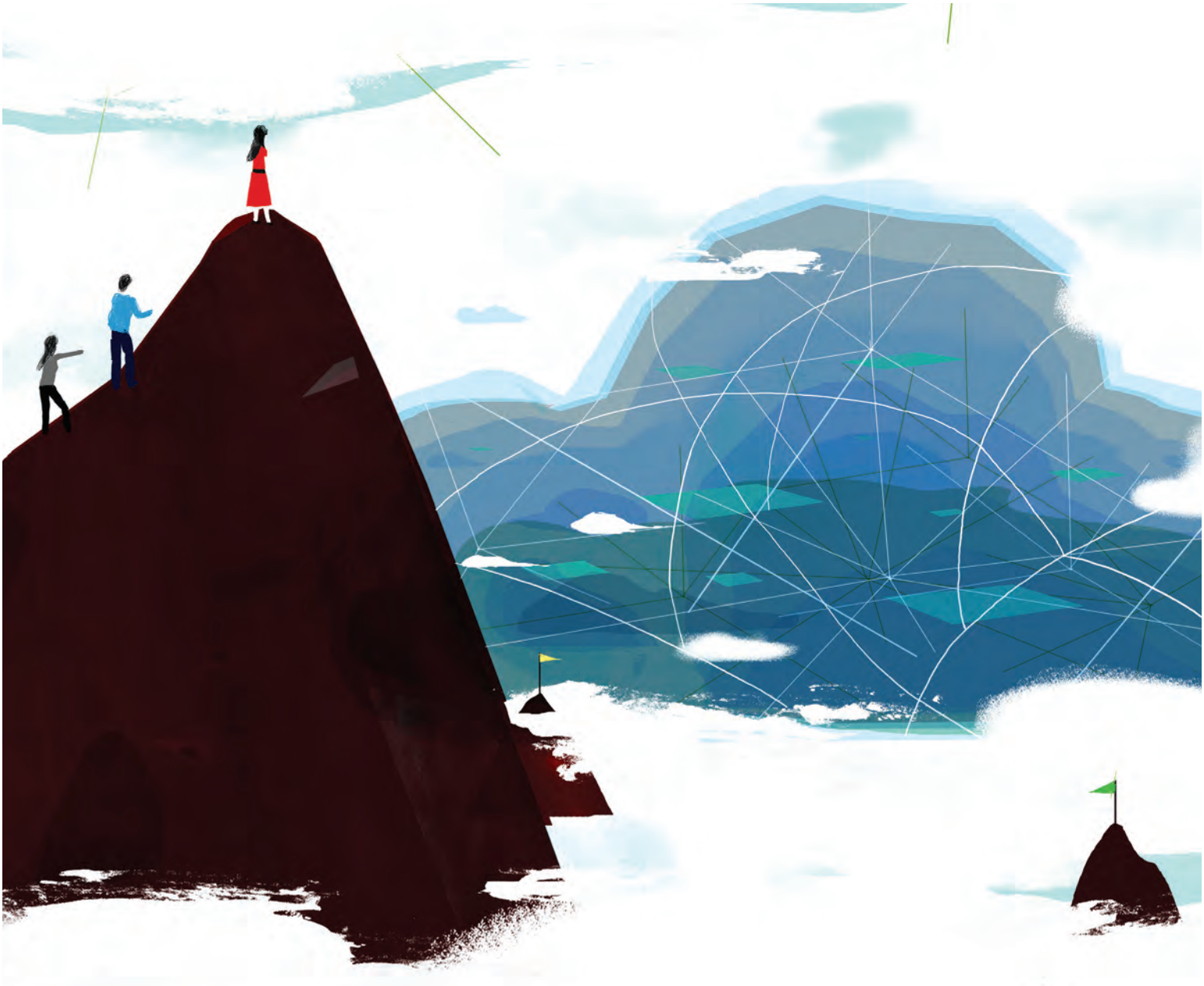


ILLUSTRATION BY JON HAN

related term "grid computing," from the high-performance computing community, suggests protocols to offer shared computation and storage over long distances, but those protocols did not lead to a software environment that grew beyond its community.

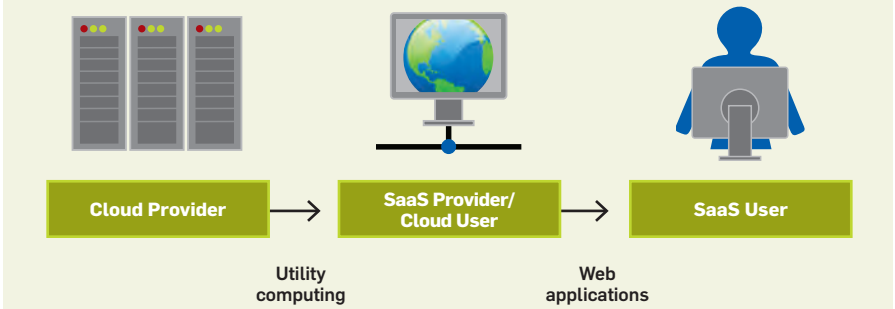
The data center hardware and software is what we will call a *cloud*. When a cloud is made available in a pay-as-you-go manner to the general public, we call it a *public cloud*; the service being sold is *utility computing*. We use the term *private cloud* to refer to internal data centers of a business or other organization, not made available to the general public, when they are large enough to benefit from the advantages of cloud computing that we discuss here. Thus, cloud computing is the sum of SaaS and utility computing, but does not include small or medium-sized data centers, even if these rely on virtualization for management. People can be users or providers of SaaS, or users or providers of utility computing. We focus on SaaS providers (cloud users) and cloud providers, which have received less attention than SaaS users. Figure 1 makes provider-user relationships clear. In some cases, the same actor can play multiple roles. For instance, a cloud provider might also host its own customer-facing services on cloud infrastructure.

From a hardware provisioning and pricing point of view, three aspects are new in cloud computing.

- The appearance of infinite computing resources available on demand, quickly enough to follow load surges, thereby eliminating the need for cloud computing users to plan far ahead for provisioning.
- The elimination of an up-front commitment by cloud users, thereby allowing companies to start small and increase hardware resources only when there is an increase in their needs.[b]
- The ability to pay for use of computing resources on a short-term basis as needed (for example, processors by the hour and storage by the day) and release them as needed, thereby rewarding conservation by letting machines and storage go when they are no longer useful.

b Note, however, that upfront commitments can still be used to reduce per-usage charges. For example, Amazon Web Services also offers long-term rental of servers, which they call reserved instances.

Figure 1. Users and providers of cloud computing. We focus on cloud computing's effects on cloud providers and SaaS providers/cloud users. The top level can be recursive, in that SaaS providers can also be a SaaS users via mashups.



We argue that the construction and operation of extremely large-scale, commodity-computer data centers at low-cost locations was the key necessary enabler of cloud computing, for they uncovered the factors of 5 to 7 decrease in cost of electricity, network bandwidth, operations, software, and hardware available at these very large economies of scale. These factors, combined with statistical multiplexing to increase utilization compared to traditional data centers, meant that cloud computing could offer services below the costs of a medium-sized data center and yet still make a good profit.

Our proposed definition allows us to clearly identify certain installations as examples and non-examples of cloud computing. Consider a public-facing Internet service hosted on an ISP who can allocate more machines to the service given four hours notice. Since load surges on the public Internet can happen much more quickly than that (Animoto saw its load double every 12 hours for nearly three days), this is not cloud computing. In contrast, consider an internal enterprise data center whose applications are modified only with significant advance notice to administrators. In this scenario, large load surges on the scale of minutes are highly unlikely, so as long as allocation can track expected load increases, this scenario fulfills one of the necessary conditions for operating as a cloud. The enterprise data center may still fail to meet other conditions for being a cloud, however, such as the appearance of infinite resources or fine-grained billing. A private data center may also not benefit from the economies of scale that make public clouds financially attractive.

Omitting private clouds from cloud computing has led to considerable debate in the blogosphere. We believe the confusion and skepticism illustrated by Larry Ellison's quote occurs when the advantages of public clouds are also claimed for medium-sized data centers. Except for extremely large data centers of hundreds of thousands of machines, such as those that might be operated by Google or Microsoft, most data centers enjoy only a subset of the potential advantages of public clouds, as Table 1 shows. We therefore believe that including traditional data centers in the definition of cloud computing will lead to exaggerated claims for smaller, so-called private clouds, which is why we exclude them. However, here we describe how so-called private clouds can get more of the benefits of public clouds through *surge computing* or *hybrid cloud computing*.

Classes of Utility Computing

Any application needs a model of computation, a model of storage, and a model of communication. The statistical multiplexing necessary to achieve elasticity and the appearance of infinite capacity available on demand requires automatic allocation and management. In practice, this is done with virtualization of some sort. Our view is that different utility computing offerings will be distinguished based on the cloud system software's level of abstraction and the level of management of the resources.

Amazon EC2 is at one end of the spectrum. An EC2 instance looks much like physical hardware, and users can control nearly the entire software stack, from the kernel upward. This low level makes it inherently difficult for Amazon to offer automatic scalability and failover because the semantics associated with replication and other state management issues are highly application-dependent. At the other extreme of the spectrum are application domain-specific platforms such as Google AppEngine, which is targeted exclusively at traditional Web applications, enforcing an application structure of clean separation between a stateless computation tier and a stateful storage tier. AppEngine's impressive automatic scaling and high-availability mechanisms, and the proprietary MegaStore data storage available to AppEngine applications, all rely on these constraints. Applications for Microsoft's Azure are written using the .NET libraries, and compiled to the Common Language Runtime, a language-independent managed environment. The framework is significantly more flexible than AppEngine's, but still constrains the user's choice of storage model and application structure. Thus, Azure is intermediate between application frameworks like AppEngine and hardware virtual machines like EC2.

Cloud Computing Economics

We see three particularly compelling use cases that favor utility computing over conventional hosting. A first case is when demand for a service varies with time. For example, provisioning a data center for the peak load it must sustain a few days per month leads to underutilization at other times. Instead, cloud computing lets an organization pay by the hour for computing resources, potentially leading to cost savings even if the hourly rate to rent a machine from a cloud provider is higher than the rate to own one. A second case is when demand is unknown in advance. For example, a Web startup will need to support a spike in demand when it becomes popular, followed potentially by a reduction once some visitors turn away. Finally, organizations that perform batch analytics can use the "cost associativity" of cloud computing to finish computations faster: using 1,000 EC2 machines for one hour costs the same as using one machine for 1,000 hours.

Although the economic appeal of

cloud computing is often described as "converting capital expenses to operating expenses" (CapEx to OpEx), we believe the phrase "pay as you go" more directly captures the economic benefit to the buyer. Hours purchased via cloud computing can be distributed non-uniformly in time (for example, use 100 server-hours today and no server-hours tomorrow, and still pay only for 100); in the networking community, this way of selling bandwidth is already known as usage-based pricing.[c] In addition, the absence of up-front capital expense allows capital to be redirected to core business investment.

Therefore, even if Amazon's pay-as-you-go pricing was more expensive than buying and depreciating a comparable server over the same period, we argue that the cost is outweighed by the extremely important cloud computing economic benefits of elasticity and transference of risk, especially the risks of overprovisioning (underutilization) and underprovisioning (saturation).

We start with elasticity. The key observation is that cloud computing's ability to add or remove resources at a fine grain (one server at a time with EC2) and with a lead time of minutes rather than weeks allows matching resources to workload much more closely. Real world estimates of average server utilization in data centers range from 5% to 20%.[15,17] This may sound shockingly low, but it is consistent with the observation that for many services the peak workload exceeds the average by factors of 2 to 10. Since few users deliberately provision for less than the expected peak, resources are idle at nonpeak times. The more pronounced the variation, the more the waste.

For example, Figure 2a assumes our service has a predictable demand where the peak requires 500 servers at noon but the trough requires only 100 servers at midnight. As long as the average utilization over a whole day is 300 servers, the actual cost per day (area under the curve) is 300 × 24 = 7,200 server hours; but since we must provision to the peak of 500 servers, we pay for 500 × 24 = 12,000 server-hours, a factor of 1.7 more. Therefore, as long as the pay-as-you-go cost per server-hour over three years (typical amortization time) is less than 1.7 times the cost of buying the server, utility computing is cheaper.

In fact, this example underestimates the benefits of elasticity, because in addition to simple diurnal patterns, most services also experience seasonal or other periodic demand variation (for example, e-commerce in December and photo sharing sites after holidays) as well as some unexpected demand bursts due to external events (for example, news events). Since it can take weeks to acquire and rack new equipment, to handle such spikes you must provision for them in advance. We already saw that even if service operators predict the spike sizes correctly, capacity is wasted, and if they overestimate the spike they provision for, it's even worse.

They may also underestimate the spike (Figure 2b), however, accidentally turning away excess users. While the cost of overprovisioning is easily measured, the cost of underprovisioning is more difficult to measure yet potentially equally serious: not only do rejected users generate zero revenue, they may never come back. For example, Friendster's decline in popularity relative to competitors Facebook and MySpace is believed to have resulted partly from user dissatisfaction with slow response times (up to 40 seconds).[16] Figure 2c aims to capture this behavior: Users will desert an underprovisioned service until the peak user load equals the data center's usable capacity, at which point users again receive acceptable service.

For a simplified example, assume that users of a hypothetical site fall into two classes: active users (those who use the site regularly) and defectors (those who abandon the site or are turned away from the site due to poor performance). Further, suppose that 10% of active users who receive poor service due to underprovisioning are "permanently lost" opportunities (become defectors), that is, users who would have remained regular visitors with a better experience. The site is initially provisioned to handle an expected peak of 400,000 users (1,000 users per server × 400 servers), but unexpected positive press drives 500,000 users in the first hour. Of the 100,000 who are turned away or receive bad service, by our assumption 10,000 of them are permanently lost, leaving an active user base of 390,000. The next hour sees 250,000 new unique users. The first 10,000 do fine, but the site is still overcapacity by 240,000 users. This results in 24,000 additional defections, leaving 376,000 permanent users. If this pattern continues, after lg(500,000) or 19 hours, the number of new users will approach zero and the site will be at capacity in steady state. Clearly, the service operator has collected less than 400,000 users' worth of steady revenue during those 19 hours, however, again illustrating the underutilization argument—to say nothing of the bad reputation from the disgruntled users.

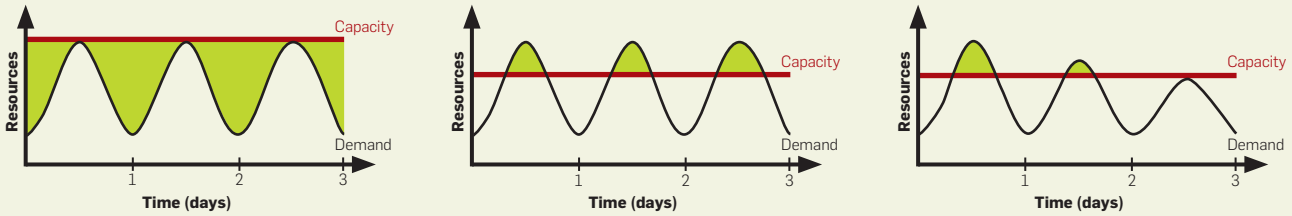
Do such scenarios really occur in practice? When Animoto[3] made its service available via Facebook, it experienced a demand surge that resulted in growing from 50 servers to 3,500 servers in three days. Even if the average

c Usage-based pricing is not renting. Renting a resource involves paying a negotiated cost to have the resource over some time period, whether or not you use the resource. Pay-as-you-go involves metering usage and charging based on actual use, independently of the time period over which the usage occurs.

Table 1. Comparing public clouds and private data centers.

Advantage	Public Cloud	Conventional Data Center
Appearance of infinite computing resources on demand	Yes	No
Elimination of an up-front commitment by Cloud users	Yes	No
Ability to pay for use of computing resources on a short-term basis as needed	Yes	No
Economies of scale due to very large data centers	Yes	Usually not
Higher utilization by multiplexing of workloads from different organizations	Yes	Depends on company size
Simplify operation and increase utilization via resource virtualization	Yes	No

Figure 2. (a) Even if peak load can be correctly anticipated, without elasticity we waste resources (shaded area) during nonpeak times. (b) Underprovisioning case 1: potential revenue from users not served (shaded area) is sacrificed. (c) Underprovisioning case 2: some users desert the site permanently after experiencing poor service; this attrition and possible negative press result in a permanent loss of a portion of the revenue stream.



utilization of each server was low, no one could have foreseen that resource needs would suddenly double every 12 hours for three days. After the peak subsided, traffic fell to a lower level. So in this real-world example, scale-up elasticity was not a cost optimization but an operational requirement, and scale-down elasticity allowed the steady-state expenditure to more closely match the steady-state workload.

Top 10 Obstacles and Opportunities for Cloud Computing

Table 2 summarizes our ranked list of critical obstacles to growth of cloud computing. The first three affect adoption, the next five affect growth, and the last two are policy and business obstacles. Each obstacle is paired with an opportunity to overcome that obstacle, ranging from product development to research projects.

Number 1. Business Continuity and Service Availability

Organizations worry about whether utility computing services will have adequate availability, and this makes some wary of cloud computing. Ironically, existing SaaS products have set a high standard in this regard. Google Search has a reputation for being highly available, to the point that even a small disruption is picked up by major news sources.[11]

Users expect similar availability from new services, which is difficult to do. Table 3 shows recorded outages for Amazon Simple Storage Service (S3), AppEngine and Gmail in 2008, and explanations for the outages. Note that despite the negative publicity due to these outages, few enterprise IT infrastructures are as good. Technical issues of availability aside, a cloud provider could suffer outages for nontechnical reasons, including going out of business or being the target of regulatory action (a recent example of the latter occurred last year, as we describe later).

Although they have not done so, cloud vendors could offer specialized hardware and software techniques in order to deliver higher reliability, presumably at a high price. This reliability could then be sold to users as a service-level agreement. But this approach only goes so far. The high-availability computing community has long followed the mantra "no single point of failure," yet the management of a cloud computing service by a single company is in fact a single point of failure. Even if the company has multiple data centers in different geographic regions using different network providers, it may have common software infrastructure and accounting systems, or the company may even go out of business. Large customers will be reluctant to migrate to cloud computing without a business-continuity strategy for such situations. We believe the best chance for independent software stacks is for them to be provided by different companies, as it has been difficult for one company to justify creating and maintain two stacks in the name of software dependability. Just as large Internet service providers use multiple network providers so that failure by a single company will not take them off the air, we believe the only plausible solution to very high availability is multiple cloud computing providers.

Number 2. Data Lock-In

Software stacks have improved interoperability among platforms, but the storage APIs for cloud computing are still essentially proprietary, or at least have not been the subject of active standardization. Thus, customers cannot easily extract their data and programs from one site to run on another. Con-

Table 2. Top 10 obstacles to and opportunities for growth of cloud computing.

	Obstacle	Opportunity
1	Availability/Business Continuity	Use Multiple Cloud Providers
2	Data Lock-In	Standardize APIs; Compatible SW to enable Surge or Hybird Cloud Computing
3	Data Confidentiality and Auditability	Deploy Encryption, VLANs, Firewalls
4	Data Transfer Bottlenecks	FedExing Disks; Higher BW Switches
5	Performance Unpredictability	Improved VM Support; Flash Memory; Gang Schedule VMs
6	Scalable Storage	Invent Scalable Store
7	Bugs in Large Distributed Systems	Invent Debugger that relies on Distributed VMs
8	Scaling Quickly	Invent Auto-Scaler that relies on ML; Snapshots for Conservation
9	Reputation Fate Sharing	Offer reputation-guarding services like those for email
10	Software Licensing	Pay-for-use licenses

cern about the difficulty of extracting data from the cloud is preventing some organizations from adopting cloud computing. Customer lock-in may be attractive to cloud computing providers, but their users are vulnerable to price increases, to reliability problems, or even to providers going out of business.

For example, an online storage service called The Linkup shut down on Aug. 8, 2008 after losing access as much as 45% of customer data.[6] The Linkup, in turn, had relied on the online storage service Nirvanix to store customer data, which led to finger pointing between the two organizations as to why customer data was lost. Meanwhile, The Linkup's 20,000 users were told the service was no longer available and were urged to try out another storage site.

One solution would be to standardize the APIs[d] in such a way that a SaaS developer could deploy services and data across multiple cloud computing providers so that the failure of a single company would not take all copies of customer data with it. One might worry that this would lead to a "race-to-the-bottom" of cloud pricing and flatten the profits of cloud computing providers. We offer two arguments to allay this fear.

First, the quality of a service matters as well as the price, so customers may not jump to the lowest-cost service. Some Internet service providers today cost a factor of 10 more than others because they are more dependable and offer extra services to improve usability.

Second, in addition to mitigating data lock-in concerns, standardization of APIs enables a new usage model in which the same software infrastructure can be used in an internal data center and in a public cloud. Such an option could enable hybrid cloud computing or surge computing in which the public cloud is used to capture the extra tasks that cannot be easily run in the data center (or private cloud) due to temporarily heavy workloads. This option could significantly expand the cloud computing market. Indeed, open-source reimplementations of proprietary cloud APIs, such as Eucalyptus and HyperTable, are first steps in enabling surge computing.

Number 3. Data Confidentiality/Auditability

Despite most companies outsourcing payroll and many companies using external email services to hold sensitive information, security is one of the most often-cited objections to cloud computing; analysts and skeptical companies ask "who would trust their essential data out there somewhere?" There are also requirements for auditability, in the sense of Sarbanes-Oxley and Health and Human Services Health Insurance Portability and Accountability Act (HIPAA) regulations that must be provided for corporate data to be moved to the cloud.

Cloud users face security threats both from outside and inside the cloud. Many of the security issues involved in protecting clouds from outside threats are similar to those already facing large data centers. In the cloud, however, this responsibility is divided among potentially many parties, including the cloud user, the cloud vendor, and any third-party vendors that users rely on for security-sensitive software or configurations.

The cloud user is responsible for application-level security. The cloud provider is responsible for physical security, and likely for enforcing external firewall policies. Security for intermediate layers of the software stack is shared between the user and the operator; the lower the level of abstraction exposed to the user, the more responsibility goes with it. Amazon EC2 users have more technical responsibility (that is, must implement or procure more of the necessary functionality themselves) for their security than do Azure users, who in turn have more responsibilities than AppEngine customers. This user responsibility, in turn, can be outsourced to third parties who sell specialty security services. The homogeneity and standardized interfaces of platforms like EC2 make it possible for a company to offer, say, configuration management or firewall rule analysis as value-added services.

While cloud computing may make external-facing security easier, it does pose the new problem of internal-facing security. Cloud providers must guard against theft or denial-of-service attacks by users. Users need to be protected from one another.

The primary security mechanism in today's clouds is virtualization. It is a powerful defense, and protects against most attempts by users to attack one another or the underlying cloud infrastructure. However, not all resources are virtualized and not all virtualization environments are bug-free. Virtualization software has been known to contain bugs that allow virtualized code to "break loose" to some extent. Incorrect network virtualization may allow user code access to sensitive portions of the provider's infrastructure, or to the resources of other users. These challenges, though, are similar to those involved in managing large non-cloud data centers, where different applications need to be protected from one another. Any large Internet service will need to ensure that a single security hole doesn't compromise everything else.

One last security concern is protecting the cloud user against the provider. The provider will by definition control the "bottom layer" of the software stack, which effectively circumvents most known security techniques. Absent radical improvements in security technology, we expect that users will use contracts and courts, rather than clever security engineering, to guard against provider malfeasance. The one important exception is the risk of inadvertent data loss. It's difficult to imagine Amazon spying on the contents of virtual machine memory; it's easy to

Table 3. Outages in AWS, AppEngine, and gmail service and outage duration date.

Service and Outage	Duration	Date
S3 outage: authentication service overload leading to unavailability[17]	2 hours	2/15/08
S3 outage: Single bit error leading to gossip protocol blowup[18]	6–8 hours	7/20/08
AppEngine partial outage: programming error[19]	5 hours	6/17/08
Gmail: site unavailable due to outage in contacts system[11]	1.5 hours	8/11/08

d Data Liberation Front; http://dataliberation.org

imagine a hard disk being disposed of without being wiped, or a permissions bug making data visible improperly.

This is a problem in non-cloud contexts as well. The standard defense, user-level encryption, is also effective in the cloud. This is already common for high-value data outside the cloud, and both tools and expertise are readily available. This approach was successfully used by TC3, a health care company with access to sensitive patient records and health care claims, when moving their HIPAA-compliant application to AWS.[3]

Similarly, auditability could be added as an additional layer beyond the reach of the virtualized guest OS, providing facilities arguably more secure than those built into the applications themselves and centralizing the software responsibilities related to confidentiality and auditability into a single logical layer. Such a new feature reinforces the cloud computing perspective of changing our focus from specific hardware to the virtualized capabilities being provided.

Number 4. Data Transfer Bottlenecks

Applications continue to become more data-intensive. If we assume applications may be "pulled apart" across the boundaries of clouds, this may complicate data placement and transport. At $100 to $150 per terabyte transferred, these costs can quickly add up, making data transfer costs an important issue. Cloud users and cloud providers have to think about the implications of placement and traffic at every level of the system if they want to minimize costs. This kind of reasoning can be seen in Amazon's development of its new cloudfront service.

One opportunity to overcome the high cost of Internet transfers is to ship disks. Jim Gray found the cheapest way to send a lot of data is to ship disks or even whole computers.[10] While this does not address every use case, it effectively handles the case of large delay-tolerant point-to-point transfers, such as importing large data sets.

To quantify the argument, assume that we want to ship 10TB from U.C. Berkeley to Amazon in Seattle, WA. Garfinkel[9] measured bandwidth to S3 from three sites and found an average write bandwidth of 5Mbits/sec to 18Mbits/sec. Suppose we get 20Mbits/sec over a WAN link. It would take

$$10 * 1012 \text{ Bytes} / (20 \times 106 \text{ bits/second}) = (8 \times 1013)/(2 \times 107) \text{ seconds} = 4{,}000{,}000 \text{ seconds},$$

which is more than 45 days. If we instead sent 10 1TB disks via overnight shipping, it would take less than a day to transfer 10TB, yielding an effective bandwidth of about 1,500Mbit/sec. For example, AWS[8] recently started offering such a service, called Import/Export.

Number 5. Performance Unpredictability

Our experience is that multiple virtual machines (VMs) can share CPUs and main memory surprisingly well in cloud computing, but that network and disk I/O sharing is more problematic. As a result, different EC2 instances vary more in their I/O performance than in main memory performance. We measured 75 EC2 instances running the STREAM memory benchmark.[14] The mean bandwidth is 1,355Mbytes/sec., with a standard deviation across instances of just 52MBytes/sec, less than or about 4% of the mean. We also measured the average disk bandwidth for 75 EC2 instances each writing 1GB files to local disk. The mean disk write bandwidth is nearly 55Mbytes per second with a standard deviation across instances of a little over 9MBytes/sec, or about 16% of the mean. This demonstrates the problem of I/O interference between virtual machines.

One opportunity is to improve architectures and operating systems to efficiently virtualize interrupts and I/O channels. Note that IBM mainframes and operating systems largely overcame these problems in the 1980s, so we have successful examples from which to learn.

Another possibility is that flash memory will decrease I/O interference. Flash is semiconductor memory that preserves information when powered off like mechanical hard disks, but since it has no moving parts, it is much faster to access (microseconds vs. milliseconds) and uses less energy. Flash memory can sustain many more I/Os per second per gigabyte of storage than disks, so multiple virtual machines

with conflicting random I/O workloads could coexist better on the same physical computer without the interference we see with mechanical disks.

Another unpredictability obstacle concerns the scheduling of virtual machines for some classes of batch processing programs, specifically for high-performance computing. Given that high-performance computing (HPC) is used to justify government purchases of $100M supercomputer centers with 10,000 to 1,000,000 processors, there are many tasks with parallelism that can benefit from elastic computing. Today, many of these tasks are run on small clusters, which are often poorly utilized. There could be a significant savings in running these tasks on large clusters in the cloud instead. Cost associativity means there is no cost penalty for using 20 times as much computing for 1/20th the time. Potential applications that could benefit include those with very high potential financial returns—financial analysis, petroleum exploration, movie animation—that would value a 20x speedup even if there were a cost premium.

The obstacle to attracting HPC is not the use of clusters; most parallel computing today is done in large clusters using the message-passing interface MPI. The problem is that many HPC applications need to ensure that all the threads of a program are running simultaneously, and today's virtual machines and operating systems do not provide a programmer-visible way to ensure this. Thus, the opportunity to overcome this obstacle is to offer something like "gang scheduling" for cloud computing. The relatively tight timing coordination expected in traditional gang scheduling may be challenging to achieve in a cloud computing environment due to the performance unpredictability just described.

Just as large ISPs use multiple network providers so that failure by a single company will not take them off the air, we believe the only plausible solution to very high availability is multiple cloud computing providers.

Number 6: Scalable Storage

Earlier, we identified three properties whose combination gives cloud computing its appeal: short-term usage (which implies scaling down as well as up when demand drops), no upfront cost, and infinite capacity on demand. While it's straightforward what this means when applied to computation, it's less clear how to apply it to persistent storage.

There have been many attempts to answer this question, varying in the richness of the query and storage API's, the performance guarantees offered, and the resulting consistency semantics. The opportunity, which is still an open research problem, is to create a storage system that would not only meet existing programmer expectations in regard to durability, high availability, and the ability to manage and query data, but combine them with the cloud advantages of scaling arbitrarily up and down on demand.

Number 7: Bugs in Large-Scale Distributed Systems

One of the difficult challenges in cloud computing is removing errors in these very large-scale distributed systems. A common occurrence is that these bugs cannot be reproduced in smaller configurations, so the debugging must occur at scale in the production data centers.

One opportunity may be the reliance on virtual machines in cloud computing. Many traditional SaaS providers developed their infrastructure without using VMs, either because they preceded the recent popularity of VMs or because they felt they could not afford the performance hit of VMs. Since VMs are de rigueur in utility computing, that level of virtualization may make it possible to capture valuable information in ways that are implausible without VMs.

Number 8: Scaling Quickly

Pay-as-you-go certainly applies to storage and to network bandwidth, both of which count bytes used. Computation is slightly different, depending on the virtualization level. Google AppEngine automatically scales in response to load increases and decreases, and users are charged by the cycles used. AWS charges by the hour for the number of instances you occupy, even if your machine is idle.

The opportunity is then to automatically scale quickly up and down in response to load in order to save money, but without violating service-level agreements. Indeed, one focus of the UC Berkeley Reliable Adaptive Distributed Systems Laboratory is the pervasive and aggressive use of statistical machine learning as a diagnostic and predictive tool to allow dynamic scaling, automatic reaction to perfor-

mance and correctness problems, and automatically managing many other aspects of these systems.

Another reason for scaling is to conserve resources as well as money. Since an idle computer uses about two-thirds of the power of a busy computer, careful use of resources could reduce the impact of data centers on the environment, which is currently receiving a great deal of negative attention. Cloud computing providers already perform careful and low-overhead accounting of resource consumption. By imposing fine-grained costs, utility computing encourages programmers to pay attention to efficiency (that is, releasing and acquiring resources only when necessary), and allows more direct measurement of operational and development inefficiencies.

Being aware of costs is the first step to conservation, but configuration hassles make it tempting to leave machines idle overnight so that startup time is zero when developers return to work the next day. A fast and easy-to-use snapshot/restart tool might further encourage conservation of computing resources.

Number 9: Reputation Fate Sharing

One customer's bad behavior can affect the reputation of others using the same cloud. For instance, blacklisting of EC2 IP addresses[13] by spam-prevention services may limit which applications can be effectively hosted. An opportunity would be to create reputation-guarding services similar to the "trusted email" services currently offered (for a fee) to services hosted on smaller ISP's, which experience a microcosm of this problem.

Another legal issue is the question of transfer of legal liability—cloud computing providers would want customers to be liable and not them (such as, the company sending the spam should be held liable, not Amazon). In March 2009, the FBI raided a Dallas data center because a company whose services were hosted there was being investigated for possible criminal activity, but a number of "innocent bystander" companies hosted in the same facility suffered days of unexpected downtime, and some went out of business.[7]

Number 10: Software Licensing

Current software licenses commonly restrict the computers on which the software can run. Users pay for the software and then pay an annual maintenance fee. Indeed, SAP announced that it would increase its annual maintenance fee to at least 22% of the purchase price of the software, which is close to Oracle's pricing.[17] Hence, many cloud computing providers originally relied on open source software in part because the licensing model for commercial software is not a good match to utility computing.

The primary opportunity is either for open source to remain popular or simply for commercial software companies to change their licensing structure to better fit cloud computing. For example, Microsoft and Amazon now offer pay-as-you-go software licensing for Windows Server and Windows SQL Server on EC2. An EC2 instance running Microsoft Windows costs $0.15 per hour instead of $0.10 per hour for the open source alternative. IBM also announced pay-as-you-go pricing for hosted IBM software in conjunction with EC2, at prices ranging from $0.38 per hour for DB2 Express to $6.39 per hour for IBM WebSphere with Lotus Web Content Management Server.

Conclusion

We predict cloud computing will grow, so developers should take it into account. Regardless of whether a cloud provider sells services at a low level of abstraction like EC2 or a higher level like AppEngine, we believe computing, storage, and networking must all focus on horizontal scalability of virtualized resources rather than on single node performance. Moreover:

1. Applications software needs to both scale down rapidly as well as scale up, which is a new requirement. Such software also needs a pay-for-use licensing model to match needs of cloud computing.

2. Infrastructure software must be aware that it is no longer running on bare metal but on VMs. Moreover, metering and billing need to be built in from the start.

3. Hardware systems should be designed at the scale of a container (at least a dozen racks), which will be the minimum purchase size. Cost of operation will match performance and cost of purchase in importance, rewarding energy proportionality[5] by putting idle portions of the memory, disk, and network into low-power mode. Processors should work well with VMs and flash memory should be added to the memory hierarchy, and LAN switches and WAN routers must improve in bandwidth and cost.

Acknowledgments

This research is supported in part by gifts from Google, Microsoft, Sun Microsystems, Amazon Web Services, Cisco Systems, Cloudera, eBay, Facebook, Fujitsu, Hewlett-Packard, Intel, Network Appliances, SAP, VMWare, Yahoo! and by matching funds from the University of California Industry/University Cooperative Research Program (UC Discovery) grant COM07-10240 and by the National Science Foundation Grant #CNS-0509559. C

The authors are associated with the UC Berkeley Reliable Adaptive Distributed Systems Laboratory (RAD Lab).

References

1. Amazon.com CEO Jeff Bezos on Animoto (Apr. 2008); http://blog.animoto.com/2008/04/21/amazon-ceo-jeff-bezos-on-animoto/.
2. Amazon S3 Team. Amazon S3 Availability Event (July 20, 2008); http://status.aws.amazon.com/s3-20080720.html.
3. Amazon Web Services. TC3 Health Case Study; http://aws.amazon.com/solutions/case-studies/tc3-health/.
4. Armbrust, M., et al. Above the clouds: A Berkeley view of cloud computing. Tech. Rep. UCB/EECS-2009-28, EECS Department, U.C. Berkeley, Feb 2009.
5. Barroso, L.A., and Holzle, U. The case for energy-proportional computing. *IEEE Computer 40*, 12 (Dec. 2007).
6. Brodkin, J. Loss of customer data spurs closure of online storage service 'The Linkup.' *Network World* (Aug. 2008).
7. Fink, J. FBI agents raid Dallas computer business (Apr. 2009); http://cbs11tv.com/local/Core.IP.Networks.2.974706.html.
8. Freedom OSS. Large data set transfer to the cloud (Apr. 2009); http://freedomoss.com/clouddataingestion.
9. Garfinkel, S. An Evaluation of Amazon's Grid Computing Services: EC2, S3 and SQS. Tech. Rep. TR-08-07, Harvard University, Aug. 2007.
10. Gray, J., and Patterson, D. A conversation with Jim Gray. *ACM Queue 1*, 4 (2003), 8–17.
11. Helft, M. Google confirms problems with reaching its services (May 14, 2009).
12. Jackson, T. We feel your pain, and we're sorry (Aug. 2008); http://gmailblog.blogspot.com/2008/08/we-feel-your-pain-and-were-sorry.html.
13. Krebs, B. Amazon: Hey spammers, Get off my cloud! *Washington Post* (July 1 2008).
14. McCalpin, J. Memory bandwidth and machine balance in current high performance computers. *IEEE Technical Committee on Computer Architecture Newsletter* (1995), 19–25.
15. Rangan, K. The Cloud Wars: $100+ Billion at Stake. Tech. Rep., Merrill Lynch, May 2008.
16. Rivlin, G. Wallflower at the Web Party (Oct. 15, 2006).
17. Siegele, L. Let It Rise: A Special Report on Corporate IT. *The Economist* (Oct. 2008).
18. Stern, A. Update from Amazon Regarding Friday's S3 Downtime. CenterNetworks (Feb. 2008); http://www.centernetworks.com/amazon-s3-downtime-update.
19. Wilson, S. AppEngine Outage. CIO Weblog (June 2008); http://www.cio-weblog.com/50226711/appengine\outage.php.