Above the Clouds: A View of Cloud Computing

Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz,
Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia
UC Berkeley Reliable Adaptive Distributed systems Laboratory (RAD Lab)

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 over-provisioning for a service whose popularity does not meet their predictions, thus wasting costly resources, or under-provisioning 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 1000 servers for one hour costs no more than using one server for 1000 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 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 paper 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. (A more detailed version of this paper is [4].)

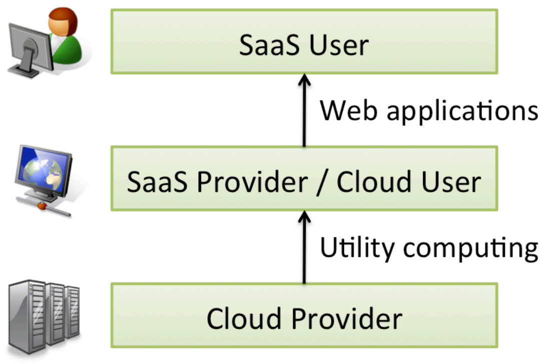


Figure 1: Users and Providers of Cloud Computing. We focus on Cloud Computing's effects on Cloud Providers and SaaS Providers/Cloud users. The top level can be recursive, in that SaaS providers can also be a SaaS users via mashups.

1 Defining Cloud Computing

Cloud Computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services. The services themselves have long been referred to as *Software as a Service (SaaS)*. The datacenter 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 datacenters of a business or other organization, not made available to the general public. Thus, Cloud Computing is the sum of SaaS and Utility Computing, but does not include Private Clouds. 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.

From a hardware point of view, three aspects are new in Cloud Computing.

1. *The illusion of infinite computing resources available on demand*, thereby eliminating the need for Cloud Computing users to plan far ahead for provisioning.
2. *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.
3. *The ability to pay for use of computing resources on a short-term basis as needed* (e.g., 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.

We argue that the construction and operation of extremely large-scale, commodity-computer datacenters 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 a private cloud, meant that cloud computing could offer services below the costs of a medium-sized datacenter and yet still make

Table 1: Comparing Public Clouds and Private Clouds.

Advantage	Public Cloud	Private Cloud
Illusion of infinite computing resources available 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 datacenters	Yes	No
Higher utilization by multiplexing of workloads from different organizations	Yes	Depends on company size
Simplify operation and increase utilization via resource virtualization	Yes	Yes

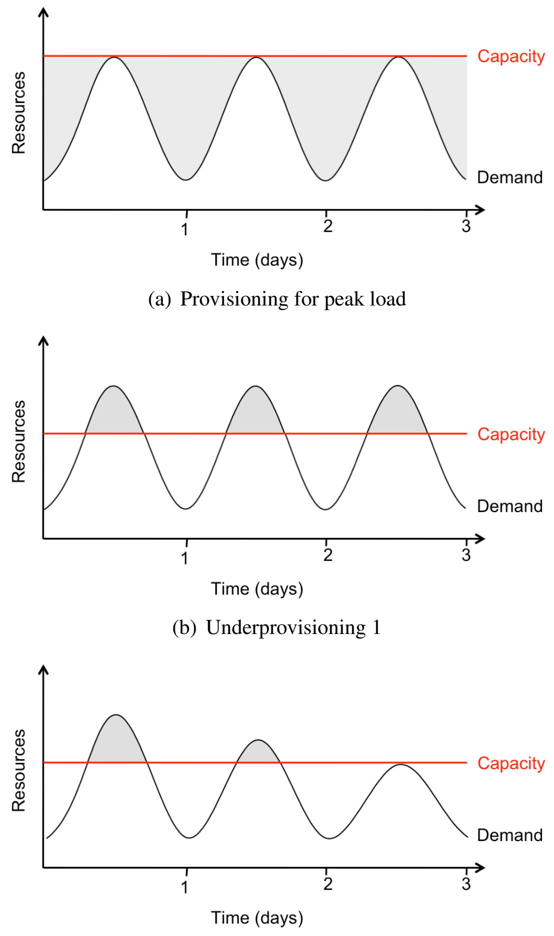
a good profit.

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 Private Clouds. Except for extremely large Private Clouds of hundreds of thousands of machines, such as those operated by Google or Microsoft, Private Clouds enjoy only a subset of the potential advantages of Public Clouds, as shown in Table 1. We therefore believe that including Private Clouds in the definition of Cloud Computing will lead to exaggerated claims for Private Clouds, which is why we exclude them. However, below we describe how Private Clouds can get more of the benefits of Private Clouds through *Surge Computing* or *Hybrid Cloud Computing*.

2 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 illusion of infinite capacity requires each of these resources to be virtualized to hide the implementation of how they are multiplexed and shared. Our view is that different utility computing offerings will be distinguished based on the programmer'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 upwards. 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. AppEngine 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. Thus, Azure is intermediate between application frameworks like AppEngine and hardware virtual machines like EC2.



(a) Provisioning for peak load

(b) Underprovisioning 1

(c) Underprovisioning 2

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.

3 Cloud Computing Economics

We see three particularly compelling use cases that favor Utility Computing over Private Clouds. A first case is when demand for a service varies with time. Provisioning a data center for the peak load it must sustain a few days per month leads to underutilization at other times, for example. 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 1000 EC2 machines for 1 hour costs the same as using 1 machine for 1000 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 (e.g., 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*. [1] 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 server utilization in datacenters range from 5% to 20% [12, 13]. 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 2(a) 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 (shaded area under the curve) is $300 \times 24 = 7200$ server-hours; but since we must provision to the peak of 500 servers, we pay for $500 \times 24 = 12000$ server-hours, a factor of 1.7 more. Therefore, as long as the pay-as-you-go cost per server-hour over 3 years (typical amoritization time) is less than 1.7 times the cost of buying the server, utility computing is cheaper.

[1] 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.

In fact, the above example *underestimates* the benefits of elasticity, because in addition to simple diurnal patterns, most services also experience seasonal or other periodic demand variation (e.g., e-commerce in December and photo sharing sites after holidays) as well as some *unexpected* demand bursts due to external events (e.g., 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 2(b)), however, accidentally turning away excess users. While the cost of overprovisioning is easily measured, the cost of underprovisioning is harder to measure yet potentially equally serious: not only do rejected users generate zero revenue, they may never come back. Figure 2(c) aims to capture this behavior: users will desert an underprovisioned service until the peak user load equals the datacenter's usable capacity, at which point users again receive acceptable service, but with fewer potential users.

For example, suppose but 10% of users who receive poor service due to underprovisioning are "permanently lost" opportunities, i.e. 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 (1000 users per server $\times$ 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 over capacity by 240,000 users. This results in 24,000 additional defections, leaving 376,000 permanent users. If this pattern continues, after $\lg 500000$ 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 [2] made its service available via Facebook, it experienced a demand surge that resulted in growing from 50 servers to 3500 servers in three days. Even if the average utilization of each server was low, no one could have foreseen that resource needs would suddenly double every 12 hours for 3 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.

4 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 concern *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.

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

Number 1 Obstacle: 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 is effectively the dial tone of the Internet: if people went to Google for search and it wasn't available, they would think the Internet was down. Users expect similar availability from new services, which is hard 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 early this year, as we describe in section 4).

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. The high-availability computing community has long followed the mantra "no single source 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 datacenters 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.

Number 2: Data Lock-In

Software stacks have improved interoperability among platforms, but the storage API's 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. Concern about the difficult 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 August 8, 2008 after losing access as much as 45% of customer data [5]. The Linkup, in turn, had relied on the online storage service Nirvanix to store customer data, and now there is 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.

The obvious solution is to standardize the APIs so 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. The obvious fear is 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 ten 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 a Private Cloud and in a Public Cloud. Such an option could enable *Surge Computing* or or *Hybrid Cloud Computing* in which the public Cloud is used to capture the extra tasks that cannot be easily run in the datacenter (or private cloud) due to temporarily heavy workloads. This option could significantly expand the Cloud Computing market. Indeed, open-source reimplementations of proprietary cloud API's, such as Eucalyptus and HyperTable, are first steps in enabling surge computing.

Number 3: Data Confidentiality/Auditability

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

Table 3: Outages in AWS, AppEngine, and Gmail

Service and Outage	Duration	Date
S3 outage: authentication service overload leading to unavailability [14]	2 hours	2/15/08
S3 outage: Single bit error leading to gossip protocol blowup. [15]	6-8 hours	7/20/08
AppEngine partial outage: programming error [16]	5 hours	6/17/08
Gmail: site unavailable due to outage in contacts system [9]	1.5 hours	8/11/08

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.

The security issues involved in protecting clouds from outside threats are similar to those already facing large datacenters, except that responsibility is divided among potentially many parties, including the cloud user, the cloud vendor, and any third-party vendors whose value-added services have been bundled into the cloud offering. For example, RightScale offers a value-added service for automatic scaling on Amazon EC2.

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 a 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 (i.e. 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 makes 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 need to guard against theft or denial of service attacks by users. Users need to be protected against one another.

The primary security mechanism in today's clouds is virtualization. This 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 virtualizion 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 mangaging large non-cloud datacenters, 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. 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 hard to imagine Amazon spying on the contents of virtual machine memory; it's easy to imagine a hard disk being disposed of without being wiped, or a permissions bug making data visible improperly.

There's an obvious defense, namely user-level encryption of storage. 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 healthcare company with access to sensitive patient records and healthcare claims, when moving their HIPAA-compliant application to AWS [1].

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 their new Cloudfront service.

One opportunity to overcome the high cost of Internet transfers is to ship disks. Jim Gray found that the cheapest way to send a lot of data is to ship disks or even whole computers [8].

To quantify the argument, assume that we want to ship 10 TB from U.C. Berkeley to Amazon in Seattle, Washington. Garfinkel measured bandwidth to S3 from three sites and found an average write bandwidth of 5 to 18 Mbits/second. [7] Suppose we get 20 Mbit/sec over a WAN link. It would take
$10 * 10^{12}$ Bytes / $(20 \times 10^6$ bits/second$) = (8 \times 10^{13})/(2 \times 10^7)$ seconds $=$ 4,000,000 seconds,
which is more than 45 days. If we instead sent ten 1 TB disks via overnight shipping, it would take less than a day to transfer 10 TB, yielding an effective bandwidth of about 1500 Mbit/sec. A new commercial service offers this approach for AWS. [3]

Number 5: Performance Unpredictability

Our experience is that multiple Virtual Machines can share CPUs and main memory surprisingly well in Cloud Computing, but that I/O sharing is more problematic. We measured 75 EC2 instances running the STREAM memory benchmark [11]. The mean bandwidth is 1355 MBytes per second, with a standard deviation of just 52 MBytes/sec, less than 4% of the mean. We also measured the average disk bandwidth for 75 EC2 instances each writing 1 GB files to local disk. The mean disk write bandwidth is nearly 55 MBytes per second with a standard deviation of a little over 9 MBytes/sec, more than 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 is used to justify Government purchases of $100M supercomputer centers with 10,000 to 1,000,000 processors, there certainly are many tasks with parallelism that can benefit from elastic computing. Cost associativity means that there is no cost penalty for using 20 times as much computing for $1/20^{th}$ the time. Potential applications that could benefit include those with very high potential financial returns—financial analysis, petroleum exploration, movie animation—and could easily justify paying a modest premium for a 20x speedup.

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. As these applications exchange data more frequently, they may also need higher bandwidth switches than are deployed today.

Number 6: Scalable Storage

Early in this paper, 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 up-front cost, and infinite capacity on-demand. While it's straightforward what this means when applied to computation, it's less obvious 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 complexity of data structures that are directly supported by the storage system (e.g., schema-less blobs vs. column-oriented storage). The opportunity, which is still an open research problem, is to create a storage system would not only meet these needs but combine them with the cloud advantages of scaling arbitrarily up and down on-demand, as well as meeting programmer expectations in regard to resource management for scalability, data durability, and high availability.

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 datacenters.

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 RAD Lab focus is the pervasive and aggressive use of statistical machine learning as a diagnostic and predictive tool to allow dynamic scaling, automatic reaction to performance 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 datacenters 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 per-hour and per-byte costs, utility computing encourages programmers to pay attention to efficiency (i.e., 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

Reputations do not virtualize well. One customer's bad behavior can affect the reputation of the cloud as a whole. For instance, blacklisting of EC2 IP addresses [10] 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 (i.e., the company sending the spam should be held liable, not Amazon). In March 2009, the FBI raided a Dallas datacenter 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 [6].

Number 10 Obstacle: 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 [13]. 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 version. 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.

5 Conclusions and Implications

We predict Cloud Computing *will* grow, so developers should take it into account. Regardless whether a cloud provider sells services at a low level of abstraction like EC2 or a higher level like AppEngine, we believe that 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* needs to be aware that it is no longer running on bare metal but on VMs. Moreover, billing needs to 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 is the minimum purchase size. Cost of operation will match performance and cost of purchase in importance, rewarding *energy proportionality* such as 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.

References

[1] TC3 Health Case Study: Amazon Web Services [online]. Available from: `http://aws.amazon.com/solutions/case-studies/tc3-health/`.

[2] Amazon.com CEO Jeff Bezos on Animoto [online]. April 2008. Available from: `http://blog.animoto.com/2008/04/21/amazon-ceo-jeff-bezos-on-animoto/`.

[3] Large data set transfer to the cloud [online]. April 2009. Available from: `http://freedomoss.com/clouddataingestion`.

[4] ARMBRUST, M., AND 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] BRODKIN, J. Loss of customer data spurs closure of online storage service 'The Linkup'. *Network World* (August 2008).

[6] FINK, J. FBI agents raid dallas computer business [online]. April 2009. Available from: `http://cbs11tv.com/local/Core.IP.Networks.2.974706.html`.

[7] GARFINKEL, S. An Evaluation of Amazon's Grid Computing Services: EC2, S3 and SQS . Tech. Rep. TR-08-07, Harvard University, August 2007.

[8] GRAY, J., AND PATTERSON, D. A conversation with Jim Gray. *ACM Queue 1*, 4 (2003), 8–17.

[9] JACKSON, T. We feel your pain, and we're sorry [online]. August 2008. Available from: `http://gmailblog.blogspot.com/2008/08/we-feel-your-pain-and-were-sorry.html`.

[10] KREBS, B. Amazon: Hey Spammers, Get Off My Cloud! *Washington Post* (July 2008).

[11] MCCALPIN, J. Memory bandwidth and machine balance in current high performance computers. *IEEE Technical Committee on Computer Architecture Newsletter* (1995), 19–25.

[12] RANGAN, K. The Cloud Wars: \$100+ billion at stake. Tech. rep., Merrill Lynch, May 2008.

[13] SIEGELE, L. Let It Rise: A Special Report on Corporate IT. *The Economist* (October 2008).

[14] STERN, A. Update From Amazon Regarding Friday's S3 Downtime. *CenterNetworks* (February 2008). Available from: `http://www.centernetworks.com/amazon-s3-downtime-update`.

[15] THE AMAZON S3 TEAM. Amazon S3 Availability Event: July 20, 2008 [online]. July 2008. Available from: `http://status.aws.amazon.com/s3-20080720.html`.

[16] WILSON, S. AppEngine Outage. *CIO Weblog* (June 2008). Available from: `http://www.cio-weblog.com/50226711/appengine\_outage.php`.