A View of Cloud Computing

Dr. Sanjay P. Ahuja, Ph.D.

2010-14 FIS Distinguished Professor of Computer Science

School of Computing, CCEC

Cloud Computing: Computing as a Utility

- Developers of new Internet services no longer require large capital outlays in hardware to deploy their service or the human expense to operate it thus avoiding both over-provisioning and under-provisioning.

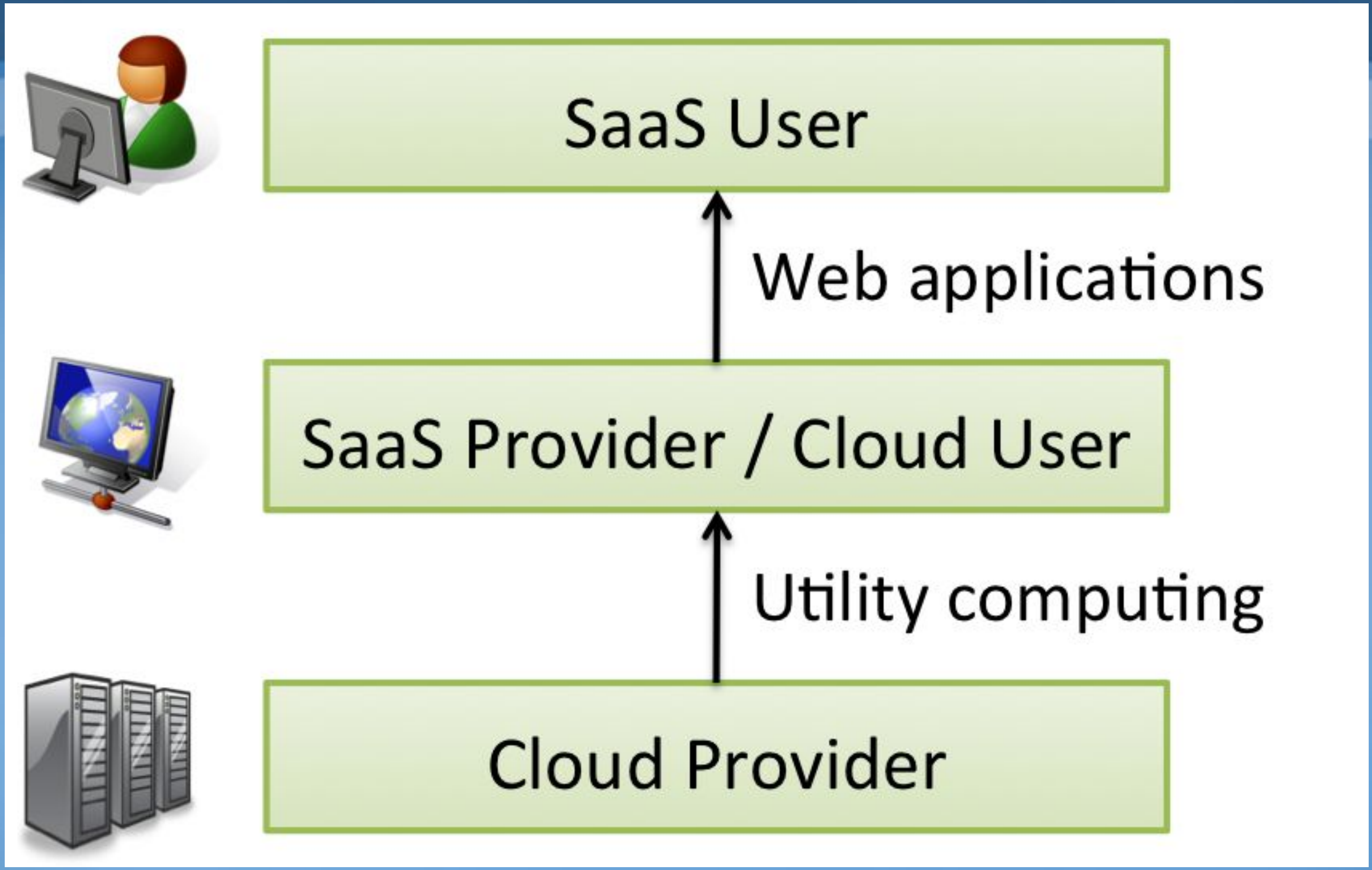
- Cloud Computing refers to both the apps delivered as services over the Internet and the hardware and system software in the datacenter that provide those services. The services themselves are referred to as ***Software-as-a-Service (SaaS).*** The datacenter hardware and system software is what we will call a **Cloud.**

- A cloud that is available in a pay-as-you-go model to the general public is ***Public Cloud***; the service being sold is ***Utility Computing***.

- ***Private Cloud*** refers to internal data centers of a business or other organization not made available to the general public.

- Cloud Computing is the sum of SaaS and Utility Computing.

Users and Providers of Cloud Computing



Users and Providers of Cloud Computing. The top level can be recursive, in that SaaS providers can also be a SaaS users. For example, a provider of rental maps might be a user of the Craigslist and Google maps services.

New Aspects of Cloud Computing from a Hardware Point of View

- **The illusion of infinite computing resources available on demand,** 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 resources only when there is an increase in their needs.

- ***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.

  – All three are important to the technical and economic changes made possible by Cloud Computing. Past efforts at utility computing failed, and in each case one or two of these three critical characteristics were missing. For example, Intel Computing Services in 2000-2001 required negotiating a contract and longer-term use than per hour.

Key Enabler of Cloud Computing

- **The construction and operation of extremely large-scale, commodity-computer datacenters at low cost locations was the key necessary enabler of Cloud Computing.**

- This 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 a good profit.

Abstraction Levels for Utility of 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.

- Utility computing offerings will be distinguished based on the level of abstraction presented to the programmer 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.

Abstraction Levels for Utility of Computing (contd.)

- 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.

- 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.

Cloud Computing will grow!

- All levels of cloud computing should aim at horizontal scalability of virtual machines over the efficiency on a single VM.
- 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.
- Infrastructure Software needs to be aware that it is no longer running on bare metal but on VMs. It needs to have billing built in from the beginning.
- 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, flash memory should be added to the memory hierarchy, and LAN switches and WAN routers must improve in bandwidth and cost.

Advantages of SaaS

- SaaS service providers enjoy greatly simplified software installation and maintenance and centralized control over versioning.
- End users can access the service “anytime, anywhere”, share data and collaborate more easily, and keep their data stored safely in the infrastructure.
- Cloud Computing does not change these arguments, but it does give more application providers the choice of deploying their product as SaaS, scaling on demand, **without building or provisioning a datacenter**.

A Successful Example

- Elastic Compute Cloud (EC2) from Amazon Web Services (AWS) sells 1.0-GHz x86 ISA “slices” for 10 cents per hour, and a new “slice”, or instance, can be added in 2 to 5 minutes.
- Amazon’s Scalable Storage Service (S3) charges $0.12 to $0.15 per gigabyte-month, with additional bandwidth charges of $0.10 to $0.15 per gigabyte to move data in to and out of AWS over the Internet.
- Amazon’s bet is that by statistically multiplexing multiple instances onto a single physical box, that box can be simultaneously rented to many customers who will not in general interfere with each others’ usage.
- A necessary but not sufficient condition for a company to become a Cloud Computing provider is that it must have existing investments not only in very large datacenters (e.g. Amazon, eBay, Google, Microsoft), but also in large-scale software infrastructure (e.g. MapReduce, the Google File System, BigTable, and Dynamo) and operational expertise required to run them.

Factors Influencing Companies to Become Cloud Computing Providers

- **Make a lot of money.**
  - Although 10 cents per server-hour seems low, very large datacenters (tens of thousands of computers) can purchase hardware, network bandwidth, and power for 1/5 to1/7 the prices offered to a medium-sized (hundreds or thousands of computers) datacenter. Further, the fixed costs of software development and deployment can be amortized over many more machines. Thus, a sufficiently large company could leverage these economies of scale to offer a service well below the costs of a medium-sized company and still make a tidy profit.

- **Leverage existing investment.**
  - Adding Cloud Computing services on top of existing infrastructure provides a new revenue stream at (ideally) low incremental cost, helping to amortize the large investments of datacenters. Indeed, many Amazon Web Services technologies were initially developed for Amazon's internal operations.

- **Defend a franchise.**
  - As conventional server and enterprise applications embrace Cloud Computing, vendors with an established franchise in those applications would be motivated to provide a cloud option of their own. For example, Microsoft Azure provides an immediate path for migrating existing customers of Microsoft enterprise applications to a cloud environment.

Factors Influencing Companies to Become Cloud Computing Providers (contd.)

- **Attack an incumbent.**
  - A company with the requisite datacenter and software resources might want to establish a beachhead in this space before a single “800 pound gorilla” emerges. Example, Google AppEngine provides an alternative path to cloud deployment whose appeal lies in its automation of many of the scalability and load balancing features that developers might otherwise have to build for themselves.

- **Leverage customer relationships.**
  - Example, IT service organizations such as IBM Global Services have extensive customer relationships through their service offerings. Providing a branded Cloud Computing offering gives those customers an anxiety-free migration path that preserves both parties’ investments in the customer relationship.

- **Become a platform.**
  - Example, Facebook’s initiative to enable plug-in applications is a great fit for cloud computing and indeed one infrastructure provider for Facebook plug-in applications is Joyent, a cloud provider. Yet Facebook’s motivation was to make their social-networking application a new development platform.

Clouds: Why now, not then?

- While the construction and operation of extremely large scale commodity-computer datacenters was the key enabler of Cloud Computing, new technology trends and business models also played a key role in making it a reality this time around.

1. **New Technology Trends and Business Models**

   With the emergence of Web 2.0, there was a shift from “high-touch, high-margin, high-commitment” provisioning of service “low-touch, low-margin, low-commitment” self-service.

   Example, in Web 1.0, accepting credit card payments from strangers required a contractual arrangement with a payment processing service such as VeriSign making it onerous for an individual or a very small business to accept credit cards online. With the emergence of PayPal, however, any individual can accept credit card payments with no contract, no long-term commitment, and only modest pay-as-you-go transaction fees (low touch = limited customer support and relationship management)

- AmazonWeb Services capitalized on this insight in 2006 by providing pay-as-you-go computing with no contract: all customers need is a credit card. A second innovation was selling hardware-level virtual machines cycles, allowing customers to choose their own software stack without disrupting each other while sharing the same hardware and thereby lowering costs further.

Clouds: Why now, not then? (contd.)

2. **New Application Opportunities**

I. **Mobile interactive applications**

Tim O’Reilly believes that *“the future belongs to services that respond in real time to information provided either by their users or by nonhuman sensors.”* Such services will be attracted to the cloud not only because they must be highly available, but also because these services generally rely on large data sets that are most conveniently hosted in large datacenters. This is especially the case for services that combine two or more data sources or other services, e.g., mashups.

II. **Parallel Data Processing**

Cloud Computing presents a unique opportunity for batch-processing and analytics jobs that analyze terabytes of data and can take hours to finish. If there is enough data parallelism in the application, users can take advantage of the cloud’s new “cost associativity”: using hundreds of computers for a short time costs the same as using a few computers for a long time.

Programming abstractions such as Google’s MapReduce and its open-source counterpart Hadoop allow programmers to express such tasks while hiding the operational complexity of choreographing parallel execution across hundreds of Cloud Computing servers.

The cost/benefit analysis must weigh the cost of moving large datasets into the cloud against the benefit of potential speedup in the data analysis.

Clouds: Why now, not then? (contd.)

II. The rise of analytics

The demand for database transaction processing is leveling off. A growing share of computing resources is now spent on understanding customers, supply chains, buying habits, ranking, and so on. Hence, while online transaction volumes will continue to grow slowly, decision support is growing rapidly, shifting the resource balance in database processing from transactions to business analytics (BI, data mining).

V. Extension of compute-intensive desktop applications

The latest versions of the mathematics software packages Matlab and Mathematica are capable of using Cloud Computing to perform expensive evaluations. Need to compare the cost of computing in the Cloud plus the cost of moving data in and out of the Cloud to the time savings from using the Cloud. An application that involves a great deal of computing per unit of data is worth investigating for a move to a cloud environment. Examples, offline image rendering, 3D animation.

Applications that are not such good candidates for the cloud ("Earthbound apps")

Some applications that would otherwise be good candidates for the cloud's elasticity and parallelism may be thwarted by data movement costs, the fundamental latency limits of getting into and out of the cloud, or both. For example, while the analytics associated with making long-term financial decisions are appropriate for the Cloud, stock trading that requires microsecond precision is not. Until the cost (and possibly latency) of wide area data transfer decrease, such applications may be less obvious candidates for the cloud.

Classes of Utility Computing

- Different utility computing offerings will be distinguished based on the level of abstraction presented to the programmer 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. The API exposed is “thin”: a few dozen API calls to request and configure the virtualized hardware. There is no a priori limit on the kinds of applications that can be hosted; the low level of virtualization—raw CPU cycles, block-device storage, IP-level connectivity- allow developers to code whatever they want. On the other hand, this 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.

Microsoft’s Azure

Is an intermediate point on this spectrum of flexibility vs. programmer convenience. Azure applications are written using the .NET libraries, and compiled to the Common Language Runtime, a language independent managed environment. The system supports general-purpose computing, rather than a single category of application. Users get a choice of language, but cannot control the underlying operating system or runtime. The libraries provide a degree of automatic network configuration and failover/scalability, but require the developer to declaratively specify some application properties in order to do so. Thus, Azure is intermediate between complete application frameworks like AppEngine on the one hand, and hardware virtual machines like EC2 on the other.

Classes of Utility Computing (contd.)

Google AppEngine

At the other extreme of the spectrum are application domain-specific platforms such as Google AppEngine and Force.com, the SalesForce business software development platform. 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.

Furthermore, AppEngine applications are expected to be request-reply based, and as such they are severely rationed in how much CPU time they can use in servicing a particular request. AppEngine's impressive automatic scaling and high-availability mechanisms, and the proprietary MegaStore (based on BigTable) data storage available to AppEngine applications, all rely on these constraints.

Thus, AppEngine is not suitable for general-purpose computing. Similarly, Force.com is designed to support business applications that run against the salesforce.com database, and nothing else.

Will one model beat out the others in the Cloud Computing space?

Different tasks will result in demand for different classes of utility computing. As an analogy, low-level languages such as C and assembly language allow fine control and close communication with the bare metal, but if the developer is writing a Web application, the mechanics of managing sockets, dispatching requests, and so on are cumbersome and tedious to code. high-level frameworks such as Ruby on Rails make these mechanics invisible to the programmer, but are only useful if the application readily fits the request/reply structure and the abstractions provided by Rails.

Classes of Utility Computing (contd.)

Examples of Cloud Computing vendors and how each provides virtualized resources (computation, storage, networking) and ensures scalability and high availability of the resources.

Computation Model (VM)

Amazon EC2

- x86 Instruction Set Architecture (ISA) via Xen VM.
- Computation elasticity allows scalability, but developer must build the machinery, or third party such as RightScale must provide it.

Microsoft's Azure

- Microsoft Common Language Runtime (CLR) VM; common intermediate form executed in managed environment.
- Machines are provisioned based on declarative descriptions (e.g. which "roles" can be replicated); automatic load balancing.

Google AppEngine

- Predefined application structure and framework; programmer-provided "handlers", written in Python, all persistent state stored in MegaStore (outside Python code).
- Automatic scaling up and down of computation and storage; network and server failover; all consistent with 3-tier Web app structure.

Classes of Utility Computing (contd.)

Storage Model

Amazon EC2

- Range of models from block store (EBS) to augmented key/blob store (SimpleDB). Computation elasticity allows scalability, but developer must build the machinery, or third party such as RightScale must provide it.
- Automatic scaling varies from no scaling or sharing (EBS) to fully automatic (SimpleDB, S3), depending on which model used.
- Consistency guarantees vary widely depending on which model used.

Microsoft's Azure

- SQL Data Services (restricted view of SQL Server) .
- Azure storage service

Google AppEngine

- MegaStore/BigTable

Classes of Utility Computing (contd.)

Networking Model

Amazon EC2

- Declarative specification of IP level topology; internal placement details concealed.
- Security Groups enable restricting which nodes may communicate.

Microsoft's Azure

- Automatic based on programmer's declarative descriptions of app components (roles).

Google AppEngine

- Fixed topology to accommodate 3-tier Web app structure.
- Scaling up and down is automatic and programmer invisible.

Cloud Computing Economics

- Economic appeal of Cloud Computing: “converting capital expenses to operating expenses (CapEx to OpEx)” or “pay as you go”. Hours purchased can be distributed non-uniformly in time.

- Even though Amazon’s pay-as-you-go pricing (for example) could be more expensive than buying and depreciating a comparable server over the same period, the cost is outweighed by the extremely important Cloud Computing economic benefits of *elasticity* and transference of risk, especially the risks of over-provisioning (underutilization) and under-provisioning (saturation).

Elasticity: Shifting the Risk

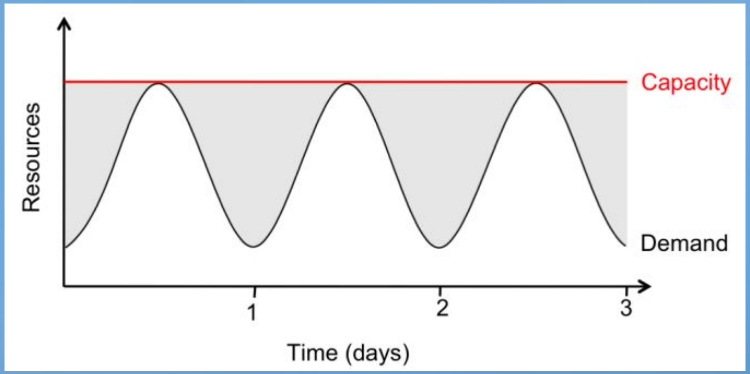
- Cloud Computing's ability to add or remove resources at a fine grain (one server at a time with EC2) and with a lead time of minutes rather than weeks allows matching resources to workload much more closely.
  - Real world estimates of server utilization in datacenters range from 5% to 20%. Users provision for the peak and allow the resources to remain idle at nonpeak times. The more pronounced the variation, the more the waste.
  - Even if peak load can be correctly anticipated, without elasticity we waste resources (shaded area) during nonpeak times.

Elasticity: Shifting the Risk (contd.)

- Example: Elasticity.

Assume our service has a predictable daily demand where the peak requires 500 servers at noon but the trough requires only 100 servers at midnight, as shown in Figure below. As long as the average utilization over a whole day is 300 servers, the actual utilization over the whole day (shaded area under the curve) is 300 * 24 = 7200 server-hours.

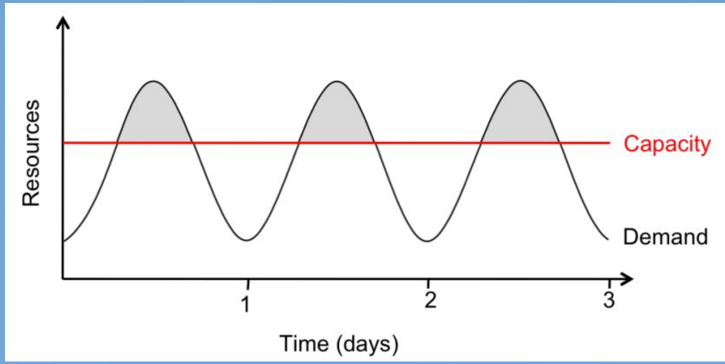
But since we must provision to the peak of 500 servers, we pay for 500 * 24 = 12000 server-hours, a factor of 1.7 more than what is needed. Therefore, as long as the pay-as-you-go cost per server-hour over 3 years is less than 1.7 times the cost of buying the server, we can save money using utility computing.



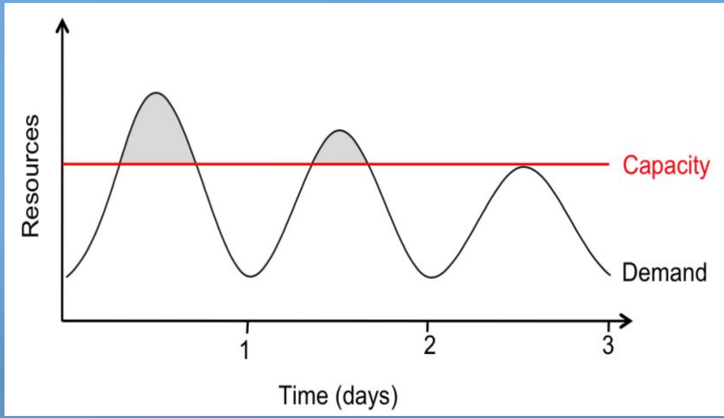
Even if service operators predict the spike sizes correctly, capacity is wasted, and if they overestimate the spike they provision for, it’s even worse.

Elasticity: Shifting the Risk (contd.)

- Service operators also underestimate the spike, however, accidentally turning away excess users. While the monetary effects of over-provisioning are easily measured, those of under-provisioning are harder to measure yet potentially equally serious: not only do rejected users generate zero revenue, they may never come back due to poor service.



- 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.



Elasticity: Shifting the Risk (contd.)

- The following equation generalizes all of the cases discussed and assumes that the Cloud Computing vendor employs usage-based pricing, in which customers pay proportionally to the amount of time and the amount of resources they use.

$$UserHours_{cloud} * (revenue - Cost_{cloud}) >= UserHours_{datacenter} * (revenue - Cost_{datacenter} / Utilization)$$

- The left-hand side multiplies the net revenue per user-hour by the number of user-hours, giving the expected profit from using Cloud Computing. The right-hand side performs the same calculation for a fixed-capacity datacenter by factoring in the average utilization, including nonpeak workloads, of the datacenter. Whichever side is greater represents the opportunity for higher profit.
  - Apparently, if Utilization = 1:0 (the datacenter equipment is 100% utilized), the two sides of the equation look the same. However, basic queuing theory tells us that as utilization approaches 1.0, system response time approaches infinity. In practice, the usable capacity of a datacenter (without compromising service) is typically 0.6 to 0.8.

- The equation makes clear that the common element in all of our examples is the ability to control the cost per user hour of operating the service.

Elasticity: Shifting the Risk (contd.)

- Economic benefits of cloud computing:
- Provides the ability to control the cost per user hour of operating the service.
  - In Example 1, the cost per user-hour without elasticity was high because of resources sitting idle—higher costs but same number of user-hours. The same thing happens when over-estimation of demand results in provisioning for workload that doesn't materialize.
  - In Example 2, the cost per user-hour increased as a result of underestimating a spike and having to turn users away: Since some fraction of those users never return, the fixed costs stay the same but are now amortized over fewer user-hours. This illustrates fundamental limitations of the "buy" model in the face of any nontrivial burstiness in the workload.

- Unexpectedly scaling down (disposing of temporarily underutilized equipment)—for example, due to a business slowdown, or ironically due to improved software efficiency—normally carries a financial penalty. Cloud Computing eliminates this penalty.
  - With 3-year depreciation, a $2,100 server decommissioned after 1 year of operation represents a "penalty" of $1,400. Cloud Computing eliminates this penalty.

- Technology trends suggest that over the useful lifetime of some purchased equipment, hardware costs will fall and new hardware and software technologies will become available. Cloud providers, who already enjoy economy-of-scale buying power, can potentially pass on some of these savings to their customers.
  - Example: Heavy users of AWS saw storage costs fall 20% and networking costs fall 50% over the last 2.5 years, and the addition of nine new services or features to AWS over less than one year. If new technologies or pricing plans become available to a cloud vendor, existing applications and customers can potentially benefit from them immediately, without incurring a capital expense.

Comparing Costs: Should I Move to the Cloud?

- Is it more economical to move an existing datacenter-hosted service to the cloud, or to keep it in a datacenter?
- Table : Gray's costs of computing resources from 2003 to 2008, normalized to what $1 could buy in 2003 vs. 2008, and compared to the cost of paying per use of $1 worth of resources on AWS at 2008 prices.

	WAN bandwidth/mo.	CPU hours (all cores)	disk storage		
Item in 2003	1 Mbps WAN link	2 GHz CPU, 2 GB DRAM	200 GB disk, 50 Mb/s transfer rate		
Cost in 2003	$100/mo.	$2000	$200		
$1 buys in 2003	1 GB	8 CPU hours	1 GB		
Item in 2008	100 Mbps WAN link	2 GHz, 2 sockets, 4 cores/socket, 4 GB DRAM	1 TB disk, 115 MB/s sustained transfer		
Cost in 2008	$3600/mo.	$1000	$100		
$1 buys in 2008	2.7 GB	128 CPU hours	10 GB		
cost/performance improvement	2.7x	16x	16x		
Cost to rent $1 on AWS in 2008	$0.27–$0.40 ($0.10–$0.15/GB * 3 GB)	$2.56 (128 * 2 VM's@$0.10 each)	$1.20–$1.50 ($0.12–$0.15/GB-month * 10 GB)		

Comparing Costs: Should I Move to the Cloud? (contd.)

- To facilitate calculations, Gray calculated what $1 bought in 2003. Table shows his numbers vs. 2008 and compares to EC2/S3 charges. At first glance, it appears that a given dollar will go further if used to purchase hardware in 2008 than to pay for use of that same hardware. However, this simple analysis glosses over several important factors.

1. Pay separately per resource.

   Most applications do not make equal use of computation, storage, and network bandwidth; some are CPU-bound, others network-bound, and so on, and may saturate one resource while underutilizing others. Pay-as-you-go Cloud Computing can charge the application separately for each type of resource, reducing the waste of underutilization.

   – While the exact savings depends on the application, suppose the CPU is only 50% utilized while the network is at capacity; then in a datacenter you are effectively paying for double the number of CPU cycles actually being used. So rather than saying it costs $2.56 to rent only $1 worth of CPU, it would be more accurate to say it costs $2.56 to rent $2 worth of CPU.

Comparing Costs: Should I Move to the Cloud? (contd.)

2. Power, cooling and physical plant costs.

- The costs of power, cooling, and the amortized cost of the building are missing from our simple analyses so far. Hamilton estimates that the costs of CPU, storage and bandwidth roughly double when those costs are amortized over the building's lifetime.
  - Using this estimate, buying 128 hours of CPU in 2008 really costs $2 rather than $1, compared to $2.56 on EC2. Similarly, 10 GB of disk space costs $2 rather than $1, compared to $1.20–$1.50 per month on S3. Lastly, S3 actually replicates the data at least 3 times for durability and performance, and will replicate it further for performance if there is high demand for the data. That means the costs are $6.00 when purchasing vs. $1.20 to $1.50 per month on S3.

3. Operations costs.

- Today, hardware operations costs are very low—rebooting servers is easy and minimally trained staff can replace broken components at the rack or server level. On one hand, since Utility Computing uses virtual machines instead of physical machines from the cloud user's point of view these tasks are shifted to the cloud provider. On the other hand, depending on the level of virtualization, much of the software management costs may remain—upgrades, applying patches, and so on.

- Example: Moving to the cloud. Suppose a biology lab creates 500 GB of new data for every wet lab experiment. A computer the speed of one EC2 instance takes 2 hours per GB to process the new data. The lab has the equivalent 20 instances locally, so the time to evaluate the experiment is 500*2/20 or 50 hours. They could process it in a single hour on 1000 instances at AWS. The cost to process one experiment would be just 1000*$0.10 or $100 in computation and another 500*$0.10 or $50 in network transfer fees. So far, so good. They measure the transfer rate from the lab to AWS at 20 Mbits/second. The transfer time is (500GB *1000MB/GB *8bits/Byte) / 20Mbits/sec = 4,000,000/20 = 200,000 seconds or more than 55 hours. Thus, it takes 50 hours locally vs. 55 + 1 or 56 hours on AWS, so they don't move to the cloud.

Conclusions and Implications for Clouds of Tomorrow

- The long dreamed vision of computing as a utility is finally emerging. The elasticity of a utility matches the need of businesses providing services directly to customers over the Internet, as workloads can grow (and shrink) far faster than 20 years ago.

- From the **cloud provider's view**, the construction of very large datacenters at low cost sites using commodity computing, storage, and networking uncovered the possibility of selling those resources on a pay-as-you-go model below the costs of many medium-sized datacenters, while making a profit by statistically multiplexing among a large group of customers.

- From the **cloud user's view**, it would be as startling for a new software startup to build its own datacenter. In addition to startups, many other established organizations take advantage of the elasticity of Cloud Computing regularly, including newspapers like the Washington Post and movie companies like Pixar.

- Developers would be wise to design their next generation of systems to be deployed into Cloud Computing. In general, the emphasis should be horizontal scalability to hundreds or thousands of virtual machines over the efficiency of the system on a single virtual machine (i.e. horizontal scaling as opposed to vertical scaling).

- When asked to identify the top benefit of deploying applications in a public cloud the most cited by developers were: freedom from maintaining hardware (19.9%), cost savings (19.4%), and scalability (16.4%). Source: http://www.hpcinthecloud.com/features/Cloud-Developments-for-Developers-An-Analyst-View-99840204.html

Conclusions and Implications for Clouds of Tomorrow (contd.)

- There are specific implications as well:

1. **Applications Software** of the future will likely have a piece that runs on clients and a piece that runs in the Cloud. The cloud piece needs to both scale down rapidly as well as scale up, which is a new requirement for software systems. Such software also needs a pay-for-use licensing model to match needs of Cloud Computing.

2. **Infrastructure Software** of the future needs to be cognizant that it is no longer running on bare metal but on virtual machines. Moreover, it needs to have billing built in from the beginning, as it is very difficult to retrofit an accounting system.

3. **Hardware Systems** of the future need to be designed at the scale of a container (at least a dozen racks) rather than at the scale of a single box or single rack, as that is the minimum level at which it will be purchased.

4. **Cost of operation** will match performance and cost of purchase in importance in the acquisition decision. Hence, they need to strive for energy proportionality by making it possible to put into low power mode the idle portions of the memory, storage, and networking, which already happens inside a microprocessor today.

5. **Hardware** should also be designed assuming that the lowest level software will be virtual machines rather than a single native operating system, and it will need to facilitate flash as a new level of the memory hierarchy between DRAM and disk. Finally, we need improvements in bandwidth and costs for both datacenter switches and WAN routers.