Eucalyptus (`http://open.eucalyptus.com/`) is a Linux-based software platform for creating cloud computing IaaS systems based on computer clusters. The project has an interface that can connect to Amazon's compute and storage cloud systems (EC2 and S3), and it maintains a private cloud as a sandbox for developers to work in. Eucalyptus works with a number of technologies for system virtualization, including VMware, Xen, and KVM. Eucalyptus is an acronym taken from the expression "Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems." Most of the major Linux vendors support this project, which is based on the original work of Rich Wolski at the University of California at Santa Barbara. The company Eucalyptus Systems was formed in 2009 to support the commercialization of the Eucalyptus Cloud Computing Platform.

OpenStack and Eucalyptus are by no means unique; several other projects are underway to create open-source cloud platforms. There also are numerous research projects in the area. The IEEE Technical Committee on Services Computing (`http://tab.computer.org/tcsc/`) sponsors a conference in this area called CLOUD and has some working groups and publications in this area. Figure 1.10 shows the home page of TCSC.

FIGURE 1.10

The home page of the IEEE Technical Committee on Services Computing (`http://tab.computer.org/tcsc/`), which works with the cloud computing community to create standards

Summary

In this chapter, you learned what is meant by the term "cloud computing." Cloud computing defines systems that are virtualized and where resources are pooled so that customers can provision computing resources as needed.

To understand cloud computing and fully appreciate its subtleties, you need to categorize the different cloud types. The two types of cloud models are those based on where the cloud is deployed and those based on the types of services that clouds offer. In this chapter, you were introduced to these cloud types.

Cloud computing has a number of benefits and has attracted great industry and general interest. Cloud computing allows systems to be created cheaply with little upfront costs and to be scaled to massive sizes, when needed. Not all applications and services benefit from cloud computing, and I presented some of the factors that help you differentiate between successful deployments and those that are not.

In Chapter 2, "Assessing the Value Proposition," the economics of cloud computing are discussed.

CHAPTER 2

Assessing the Value Proposition

In this chapter, the various attributes of cloud computing that make it a unique service are described. These attributes—scalability, elasticity, low barrier to entry, and a utility type of delivery—completely change how applications are created, priced, and delivered. I describe the factors that have led to this new model of computing. Early adopters of these services are those enterprises that can best make use of these characteristics.

To get a sense for the value of cloud computing, this chapter compares it to on-premises systems. From this perspective, a number of benefits for cloud computing emerge, along with many obstacles. I describe these factors in some detail. Aside from technological reasons, behavior considerations associated with cloud adoption are discussed.

Cloud computing is particularly valuable because it shifts capital expenditures into operating expenditures. This has the benefit of decoupling growth from cash on hand or from requiring access to capital. It also shifts risk away from an organization and onto the cloud provider.

This chapter describes how to begin to measure the costs of cloud computing and some of the tools that you can use to do so. The concept of optimization known as right-sizing is described, and cloud computing has some unique new capabilities in this area.

Service Level Agreements (SLAs) are an important aspect of cloud computing. They are essentially your working contract with any provider. Cloud computing is having impact on software licensing, which although not entirely settled is also described in this chapter.

IN THIS CHAPTER

- Discovering the attributes that make cloud computing unique
- Applying cloud computing when it is the best option
- Measuring the costs associated with cloud computing systems
- Learning about Service Level Agreements and Licensing

Measuring the Cloud's Value

Cloud computing presents new opportunities to users and developers because it is based on the paradigm of a shared multitenant utility. The ability to access pooled resources on a pay-as-you-go basis provides a number of system characteristics that completely alter the economics of information technology infrastructures and allows new types of access and business models for user applications.

Any application or process that benefits from economies of scale, commoditization of assets, and conformance to programming standards benefits from the application of cloud computing. Any application or process that requires a completely customized solution, imposes a high degree of specialization, and requires access to proprietary technology is going to expose the limits of cloud computing rather quickly. Applications that work with cloud computing are ones that I refer to as "low touch" applications; they tend to be applications that have low margins and usually low risk. The "high touch" applications that come with high margins require committed resources and pose more of a risk; those applications are best done on-premises.

A cloud is defined as the combination of the infrastructure of a datacenter with the ability to provision hardware and software. A service that concentrates on hardware follows the Infrastructure as a Service (IaaS) model, which is a good description for the Amazon Web Service described in Chapter 9. When you add a software stack, such as an operating system and applications to the service, the model shifts to the Software as a Service (SaaS) model. Microsoft's Windows Azure Platform, discussed in Chapter 10, is best described as currently using SaaS model. When the service requires the client to use a complete hardware/software/application stack, it is using the most refined and restrictive service model, called the Platform as a Service (PaaS) model. The best example of a PaaS offering is probably SalesForce.com. The Google App Engine discussed in Chapter 11 is another PaaS. As the Windows Azure Platform matures adding more access to Microsoft servers, it is developing into a PaaS model rather quickly.

Cross-Ref

Chapter 4, "Understanding Services and Applications by Type," describes a number of these XaaS service models. Cloud computing is in its wild and wooly frontier days, so it's best to take a few of the lesser known acronyms with a grain of salt. ■

A cloud is an infrastructure that can be partitioned and provisioned, and resources are pooled and virtualized. If the cloud is available to the public on a pay-as-you-go basis, then the cloud is a public cloud, and the service is described as a utility. If the cloud is captive in an organization's infrastructure (network), it is referred to as a private cloud. When you mix public and private clouds together, you have a hybrid cloud. Any analysis of the potential of cloud computing must account for all these possibilities.

These are the unique characteristics of an ideal cloud computing model:

- **Scalability:** You have access to unlimited computer resources as needed.

  This feature obviates the need for planning and provisioning. It also enables batch processing, which greatly speeds up high-processing applications.

- **Elasticity:** You have the ability to right-size resources as required.

  This feature allows you to optimize your system and capture all possible transactions.

- **Low barrier to entry:** You can gain access to systems for a small investment.

  This feature offers access to global resources to small ventures and provides the ability to experiment with little risk.

- **Utility:** A pay-as-you-go model matches resources to need on an ongoing basis.

  This eliminates waste and has the added benefit of shifting risk from the client.

It is the construction of large datacenters running commodity hardware that has enabled cloud computing to gain traction. These datacenters gain access to low-cost electricity, high-network bandwidth pipes, and low-cost commodity hardware and software, which, taken together, represents an economy of scale that allows cloud providers to amortize their investment and retain a profit. It has been estimated that it costs around $100 million to create a datacenter with sufficient scale to be a cloud provider. At this scale, the resources for a large datacenter have been estimated to be between 35 percent and 20 percent lower than the pricing that is offered to medium-sized datacenters.

Note

Members of the UC Berkeley Reliable Adaptive Distribute System Laboratory have published a white paper summarizing the benefits of Cloud Computing called "Above the Clouds: A Berkeley View of Cloud Computing," which can be found at `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.pdf`**. This lab was funded by contributions from the major cloud providers and from the NSF, but it's a useful source of analytical data.** ■

The virtualization of pooled resources—processors or compute engines, storage, and network connectivity—optimizes these investments and allows the cloud provider to pass along these economies to customers. Pooling also blurs the differences between a small deployment and a large one because scale becomes tied only to demand.

Companies become cloud computing providers for several reasons:

- **Profit:** The economies of scale can make this a profitable business.
- **Optimization:** The infrastructure already exists and isn't fully utilized.

  This was certainly the case for Amazon Web Services.

- **Strategic:** A cloud computing platform extends the company's products and defends their franchise.

  This is the case for Microsoft's Windows Azure Platform.

- **Extension:** A branded cloud computing platform can extend customer relationships by offering additional service options.

  This is the case with IBM Global Services and the various IBM cloud services.

- **Presence:** Establish a presence in a market before a large competitor can emerge.

  Google App Engine allows a developer to scale an application immediately. For Google, its office applications can be rolled out quickly and to large audiences.

- **Platform:** A cloud computing provider can become a hub master at the center of many ISV's (Independent Software Vendor) offerings.

  The customer relationship management provider SalesForce.com has a development platform called Force.com that is a PaaS offering.

The development of cloud computing has been likened to the situation that has faced hardware companies that rely on proprietary silicon to produce their products: the AMDs, nVidias, eVGAs, and Apples of the world. Because a semiconductor fabrication facility costs several billion dollars to create, these companies were at a severe disadvantage to companies such as Intel or NEC, which could build their own fabs or fabrication facilities. (A *fab* is a facility that is a self-contained semiconductor assembly line.) Companies such as TSMC (Taiwan Semiconductor Manufacturing Company) have come along that provide fabrication based on customer designs, spreading their risk and optimizing their operation. Cloud computing is much the same.

Early adopters and new applications

Cloud computing is still in its infancy, but trends in adoption are already evident. In his white paper "Realizing the Value Proposition of Cloud Computing: CIO's Enterprise IT Strategy for the Cloud," Jitendra Pal Thethi, a Principle Architect for Infosys' Microsoft Technology Group, lists the following business types as the Top 10 adopters of cloud computing:

1. Messaging and team collaboration applications
2. Cross enterprise integration projects
3. Infrastructure consolidation, server, and desktop virtualization efforts
4. Web 2.0 and social strategy companies
5. Web content delivery services
6. Data analytics and computation
7. Mobility applications for the enterprise
8. CRM applications
9. Experimental deployments, test bed labs, and development efforts
10. Backup and archival storage

You can download Thethi's paper from: `http://www.infosys.com/cloud-computing/white-papers/Documents/realizing-value-proposition.pdf`.

As a group, early adopters are categorized by their need for ubiquity and access to large data sets.

Around 2000-2001, some companies began using the Internet to stage various types of user-facing applications such as office suites, accounting packages, games, and so forth. The first attempts by large ISPs to create utility computing date to that period. By 2005-2006, several Internet sites had become sufficiently large that they had developed extensive infrastructure for their own sites. The excess capacity in these sites began to be offered to partners and eventually to the general public.

The infrastructure cloud computing market was established as profitable so that by 2007-2008 many more vendors became cloud providers.

The nature of cloud computing should provide us with new classes of applications, some of which are currently emerging. Because Wide Area Network (WAN) bandwidth provides one of the current bottlenecks for distributed computing, one of the major areas of interest in cloud computing is in establishing content delivery networks (CDN). These solutions are also called edge networks because they cache content geographically.

Due to its scalability, cloud computing provides a means to do high-performance parallel batch processing that wasn't available to many organizations before. If a company must perform a complex data analysis that might take a server a month to do, then with cloud computing you might launch 100 virtual machine instances and complete the analysis in around 8 hours for the same cost. Processor-intensive applications that users currently perform on their desktops such as mathematical simulations in Mathematica and Matlab, graphic rendering in Renderman, and long encoding/decoding tasks are other examples of applications that could benefit from parallel batch processing and be done directly from the desktop. The economics must work out, but this approach is a completely new one for most people and is a game changer.

The relative ubiquity of cloud computing systems also enables emerging classes of interactive mobile applications. The large array of sensors, diagnostic, and mobile devices, all of which both generate data and consume data, require the use of large data sets and on-demand processing that are a good fit for the cloud computing model. Cloud computing also can provide access to multiple data sets that can support layered forms of information, the types of information you get when you view a mashup, such as the layers of information like Panoramio provided in the application Google Earth.

Note

A mashup is an application or Web page that combines data from two or more sources. Ajax (Asynchronous JavaScript and XML) is often used to create mashups. ■

The laws of cloudonomics

Joe Wienman of AT&T Global Services has concisely stated the advantages that cloud computing offers over a private or captured system. His article appeared on Gigaom.com at: `http://gigaom.com/2008/09/07/the-10-laws-of-cloudonomics/`. A summary of Wienman's "10 Laws of Cloudonomics" follows and his interpretation:

1. **Utility services cost less even though they cost more.**

   Utilities charge a premium for their services, but customers save money by not paying for services that they aren't using.

2. **On-demand trumps forecasting.**

   The ability to provision and tear down resources (de-provision) captures revenue and lowers costs.

3. **The peak of the sum is never greater than the sum of the peaks.**

   A cloud can deploy less capacity because the peaks of individual tenants in a shared system are averaged over time by the group of tenants.

4. **Aggregate demand is smoother than individual.**

   Multi-tenancy also tends to average the variability intrinsic in individual demand because the "coefficient of random variables" is always less than or equal to that of any of the individual variables. With a more predictable demand and less variation, clouds can run at higher utilization rates than captive systems. This allows cloud systems to operate at higher efficiencies and lower costs.

5. **Average unit costs are reduced by distributing fixed costs over more units of output.**

   Cloud vendors have a size that allows them to purchase resources at significantly reduced prices. (This feature was described in the previous section.)

6. **Superiority in numbers is the most important factor in the result of a combat (Clausewitz).**

   Weinman argues that a large cloud's size has the ability to repel botnets and DDoS attacks better than smaller systems do.

7. **Space-time is a continuum (Einstein/Minkowski).**

   The ability of a task to be accomplished in the cloud using parallel processing allows real-time business to respond quicker to business conditions and accelerates decision making providing a measurable advantage.

8. **Dispersion is the inverse square of latency.**

   Latency, or the delay in getting a response to a request, requires both large-scale and multi-site deployments that are a characteristic of cloud providers. Cutting latency in half requires four times the number of nodes in a system.

9. **Don't put all your eggs in one basket.**

   The reliability of a system with n redundant components and a reliability of r is $1-(1-r)^n$. Therefore, when a datacenter achieves a reliability of 99 percent, two redundant datacenters have a reliability of 99.99 percent (four nines) and three redundant datacenters can achieve a reliability of 99.9999 percent (six nines). Large cloud providers with geographically dispersed sites worldwide therefore achieve reliability rates that are hard for private systems to achieve.

10. **An object at rest tends to stay at rest (Newton).**

    Private datacenters tend to be located in places where the company or unit was founded or acquired. Cloud providers can site their datacenters in what are called "greenfield sites." A *greenfield site* is one that is environmentally friendly: locations that are on a network backbone, have cheap access to power and cooling, where land is inexpensive, and the environmental impact is low. A network backbone is a very high-capacity network connection. On the Internet, an Internet backbone consists of the high-capacity routes and routers that are typically operated by an individual service provider such as a government or commercial

entity. You can access a jump page of Internet backbone maps at: `http://www.nthelp.com/maps.htm`.

Cloud computing obstacles

Cloud computing isn't a panacea; nor is it either practical or economically sensible for many computer applications that you encounter. In practice, cloud computing can deviate from the ideals described in the previous list in many significant ways. The illusion of scalability is bounded by the limitations cloud providers place on their clients. Resource limits are exposed at peak conditions of the utility itself. As we all know, power utilities suffer brownouts and outages when the temperature soars, and cloud computing providers are no different. You see these outages on peak computing days such as Black Monday, which is the Monday after Thanksgiving in the United States when Internet Christmas sales traditionally start.

The illusion of low barrier to entry may be pierced by an inconsistent pricing scheme that makes scaling more expensive than it should be. You can see this limit in the nonlinearity of pricing associated with "extra large" machine instances versus their "standard" size counterparts. Additionally, the low barrier to entry also can be accompanied by a low barrier to provisioning. If you make a provisioning error, it can lead to vast costs.

Cloud computing vendors run very reliable networks. Often, cloud data is load-balanced between virtual systems and replicated between sites. However, even cloud providers experience outages. In the cloud, it is common to have various resources, such as machine instances, fail. Except for tightly managed PaaS cloud providers, the burden of resource management is still in the hands of the user, but the user is often provided with limited or immature management tools to address these issues.

Table 2.1 summarizes the various obstacles and challenges that cloud computing faces. These issues are described in various chapters in this book.

TABLE 2.1

Challenges and Obstacles to Cloud Computing

Subject Area	Captive	Cloud	Challenge
Accounting Management	Chargeback or Licensed	Usage	In private systems, costs associated with operations are fixed due to licenses and must be charged back to accounts based on some formula or usage model. For cloud computing, the pay-as-you-go usage model allows for costs to be applied to individual accounts directly.

(continued)

TABLE 2.1 *(continued)*

Subject Area	Captive	Cloud	Challenge
Compliance	Policy-based	Proprietary	Compliance to laws and policies varies by geographical area. This requires that the cloud accommodate multiple compliance regimes.
Data Privacy	Bounded	Shared with cloud	To ensure data privacy in the cloud, additional security methods such as private encryption, VLANs, firewalls, and local storage of sensitive data is necessary.
Monitoring	Variable but under control	Limited	For private systems, any monitoring system the organization wishes to deploy can be brought to bear. Cloud computing models often have limited monitoring because it is vendor-defined.
Network Bottlenecks	Low	High	Network bottlenecks occur when large data sets must be transferred. This is the case for staging, replication, and other operations. On-premise operations use LANs that are better able to accommodate transfers than the WAN connections used in cloud computing.
Reputation	Individual	Shared	The reputation for cloud computing services for the quality of those services is shared by tenants. An outage of the cloud provider impacts individuals. Clouds often have higher reliability than private systems.
Security	Restricted	Federated	The different trust mechanisms require that applications be structured differently and that operations be modified to account for these differences.
Service Level Agreements (SLAs)	Customized	Cloud specific	Cloud SLAs are standardized in order to appeal to the majority of its audience. Custom SLAs that allow for multiple data sources are difficult to obtain or enforce. Cloud SLAs do not generally offer industry standard chargeback rates, and negotiations with large cloud providers can be difficult for small users. Business risks that aren't covered by a cloud SLA must be taken into account.
Software Stack	Customized	Commoditized	The cloud enforces standardization and lowers the ability of a system to be customized for need.

Subject Area	Captive	Cloud	Challenge
Storage	Scalable and high performance	Scalable but low performance	Enterprise class storage is under the control of an on-premise system and can support high speed queries. In cloud computing large data stores are possible but they have low bandwidth connection. High speed local storage in the cloud tends to be expensive.
Vendor Lock-in	Varies by company	Varies by platform	Vendor lock-in is a function of the particular enterprise and application in an on-premises deployment. For cloud providers, vendor lock-in increases going from the IaaS to SaaS to PaaS model. Vendor lock-in for a cloud computing solution in a PaaS model is very high.

Behavioral factors relating to cloud adoption

The issues described in Table 2.1 are real substantive issues that can be measured and quantified. However, a number of intrinsic properties of cloud computing create cognitive biases in people that are obstacles to cloud adoption and are worth mentioning. This goes for users as well as organizations. Duke University economist Dan Ariely, in his book *Predictably Irrational: The Hidden Forces that Shape Our Decisions* (Harper Collins, 2008), explores how people often make choices that are inconsistent based on expediency or human nature. Joe Weinman has expanded on these ideas and some others to formulate ten more "laws" for cloud computing adoption based on human behavior. You can read the original article at `http://gigaom.com/2010/06/06/lazy-hazy-crazy-the-10-laws-of-behavioral-cloudonomics/`.

The "10 Laws of Behavioral Cloudonomics" are summarized below:

1. **People are risk averse and loss averse.**

   Ariely argues that losses are more painful than gains are pleasurable. Cloud initiatives may cause the concerns of adoption to be weighed more heavily than the benefits accrued to improving total costs and achieving greater agility.

2. **People have a flat-rate bias.**

   Loss aversion expresses itself by preferences to flat-rate plans where risk is psychologically minimized to usage-based plans where costs are actually less.

   Weiman cites the work of Anja Lambrecht and Bernd Skiera, "Paying Too Much and Being Happy About It: Existence, Causes, and Consequences of Tariff-Choice Biases" (`http://www.test2.marketing.wiwi.uni-frankfurt.de/fileadmin/Publikationen/Lambrecht_Skiera_Tariff-Choice-Biases-JMR.pdf`) for this point.

3. **People have the need to control their environment and remain anonymous.**

   The need for environmental control is a primal directive. Loss of control leads to "learned helplessness" and shorter life spans.

   You can read about the research in this area in David Rock's Oxford Leadership Journal article "Managing with the Brain in Mind," found at `http://www.oxfordleadership.com/journal/vol1_issue1/rock.pdf`. The point about shorter life spans comes from the work of Judith Rodin and Ellen Langer, "Long-Term Effects of a Control-Relevant Intervention with the Institutionalized Aged" (`http://capital2.capital.edu/faculty/jfournie/documents/Rodin_Judith.pdf`), which appeared in the Journal of Personality and Social Psychology in 1977.

4. **People fear change.**

   Uncertainty leads to fear, and fear leads to inertia. This is as true for cloud computing as it is for investing in the stock market.

5. **People value what they own more than what they are given.**

   This is called the endowment effect. It is a predilection for existing assets that is out of line with their value to others. The cognitive science behind this principle is referred to as the choice-supportive bias (`http://en.wikipedia.org/wiki/Choice-supportive_bias`).

6. **People favor the status quo and invest accordingly.**

   There is a bias toward the way things have been and a willingness to invest in the status quo that is out of line with their current value. In cognitive science, the former attribute is referred to as the status quo bias (`http://en.wikipedia.org/wiki/Status_quo_bias`), while the latter attribute is referred to as an escalation of commitment (`http://en.wikipedia.org/wiki/Escalation_of_commitment`).

7. **People discount future risk and favor instant gratification.**

   Weinman argues that because cloud computing is an on-demand service, the instant gratification factor should favor cloud computing.

8. **People favor things that are free.**

   When offered an item that is free or another that costs money but offers a greater gain, people opt for the free item. Weinman argues that this factor also favors the cloud computing model because upfront costs are eliminated.

9. **People have the need for status.**

   A large IT organization with substantial assets is a visual display of your status; a cloud deployment is not. This is expressed as a pride of ownership.

10. **People are incapacitated by choice.**

    The Internet enables commerce to shift to a large inventory where profit can be maintained by many sales of a few items each, the so-called long tail. When this model is applied to cloud computing, people tend to be overwhelmed by the choice and delay adoption.

Measuring cloud computing costs

As you see, cloud computing has many advantages and disadvantages, and you can't always measure them. You can measure costs though, and that's a valuable exercise. Usually a commodity is cheaper than a specialized item, but not always. Depending upon your situation, you can pay more for public cloud computing than you would for owning and managing your private cloud, or for owning and using software as well. That's why it's important to analyze the costs and benefits of your own cloud computing scenario carefully and quantitatively. You will want to compare the costs of cloud computing to private systems.

The cost of a cloud computing deployment is roughly estimated to be

$$Cost_{CLOUD} = \Sigma(UnitCost_{CLOUD} \times (Revenue - Cost_{CLOUD}))$$

where the unit cost is usually defined as the cost of a machine instance per hour or another resource.

Depending upon the deployment type, other resources add additional unit costs: storage quantity consumed, number of transactions, incoming or outgoing amounts of data, and so forth. Different cloud providers charge different amounts for these resources, some resources are free for one provider and charged for another, and there are almost always variable charges based on resource sizing. Cloud resource pricing doesn't always scale linearly based on performance.

To compare your cost benefit with a private cloud, you will want to compare the value you determine in the equation above with the same calculation:

$$Cost_{DATACENTER} = \Sigma(UnitCost_{DATACENTER} \times (Revenue - (Cost_{DATACENTER}/Utilization))$$

Notice the additional term for Utilization added as a divisor to the term for $Cost_{DATACENTER}$. This term appears because it is assumed that a private cloud has capacity that can't be captured, and it is further assumed that a private cloud doesn't employ the same level of virtualization or pooling of resources that a cloud computing provider can achieve. Indeed, no system can work at 100 percent utilization because queuing theory states that as the system approaches 100 percent, the latency and response times go to infinity. Typical efficiencies in datacenters are between 60 and 85 percent. It is also further assumed that the datacenter is operating under averaged loads (not at peak capacity) and that the capacity of the datacenter is fixed by the assets it has.

There is another interesting aspect to the calculated costs associated with $Cost_{CLOUD}$ vs. $Cost_{DATACENTER}$: The costs associated with resources in the cloud computing model $Cost_{CLOUD}$ can be unbundled to a greater extent than the costs associated with $Cost_{DATACENTER}$. The $Cost_{DATACENTER}$ consists of the summation of the cost of each of the individual systems with all the associated resources, as follows:

$$Cost_{DATACENTER} = {}_{1}{}^{n}\Sigma(UnitCost_{DATACENTER} \times (Revenue - (Cost_{DATACENTER}/Utilization))_{SYSTEM_n},$$

where the sum includes terms for System 1, System 2, System 3, and so on.

The costs of a system in a datacenter must also include the overhead associated with power, cooling, and the physical plant. Estimates of these additional overheads indicate that over the lifetime of a system, overhead roughly doubles the cost of any system. For a server with a four-year lifetime, you would therefore need to include an overhead roughly equal to 25 percent of the system's acquisition cost.

The overhead associated with IT staff is also a major cost, but it's highly variable from organization to organization. It is not uncommon for the burden cost of a system in a datacenter to be 150 percent of the cost of the system itself.

The costs associated with the cloud model are calculated rather differently. Each resource has its own specific cost and many resources can be provisioned independently of one another. In theory, therefore, the $Cost_{CLOUD}$ is better represented by the equation:

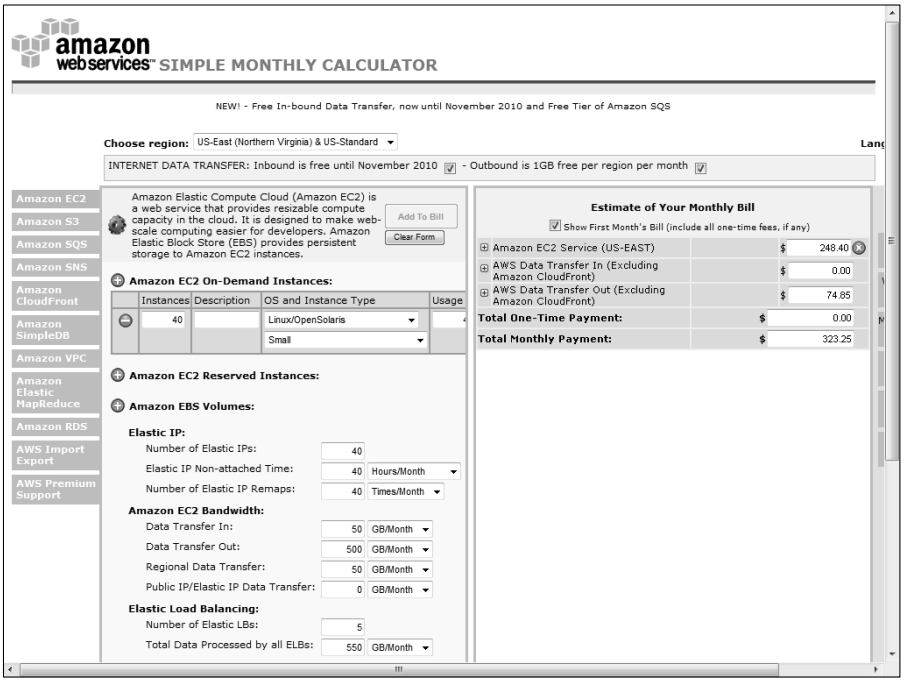
$$Cost_{CLOUD} = {}_{1}{}^{n}\Sigma(UnitCost_{CLOUD} \times (Revenue - Cost_{CLOUD})) + {}_{1}{}^{n}\Sigma(UnitCost^{CLOUD} \times (Revenue - Cost^{CLOUD}))^{INSTANCE_n} + . {}_{1}{}^{n}\Sigma(UnitCost_{CLOUD} \times (Revenue - Cost_{CLOUD}))_{STORAGE\_UNIT_n} + {}_{1}{}^{n}\Sigma(UnitCost_{CLOUD} \times (Revenue - Cost_{CLOUD}))_{NETWORK\_UNIT_n} + \ldots$$

In practice, cloud providers offer packages of machine instances with a fixed relationship between a machine instances, memory allocation (RAM), and network bandwidth. Storage and transactions are unbundled and variable.

Many cloud computing providers have created their own cost calculators to support their customers. Amazon lets you create a simulated billing based on the machine instances, storage, transactions, and other resources that you provision. An example is the Amazon Simple Monthly Calculator (`http://calculator.s3.amazonaws.com/calc5.html`) shown in Figure 2.1. You can find similar calculators elsewhere or download a spreadsheet with the calculations built into it from the various sites.

FIGURE 2.1

The Amazon Web Service Simple Monthly Calculator



Avoiding Capital Expenditures

A major part of cloud computing's value proposition and its appeal is its ability to convert capital expenses (CapEx) to operating expenses (OpEx) through a usage pricing scheme that is elastic and can be right-sized. The conversion of real assets to virtual ones provides a measure of protection against too much or too little infrastructure. Essentially, moving expenses onto the OpEx side of a budget allows an organization to transfer risk to their cloud computing provider.

Capitalization may be the single largest reason that new businesses fail, and it is surely an impediment to established businesses starting new enterprises. Growth itself can be difficult when revenues don't cover the expansion and obtaining financing is difficult. A company wishing to grow would normally be faced with the following options:

- Buy the new equipment, and deploy it in-house
- Lease the equipment for a set period of time
- Outsource the operation to a managed-services organization

Capital expenditures must create the infrastructure necessary to capture the transactions that the business needs. However, if demand is variable, then it is an open question as to how much infrastructure is needed to support demand.

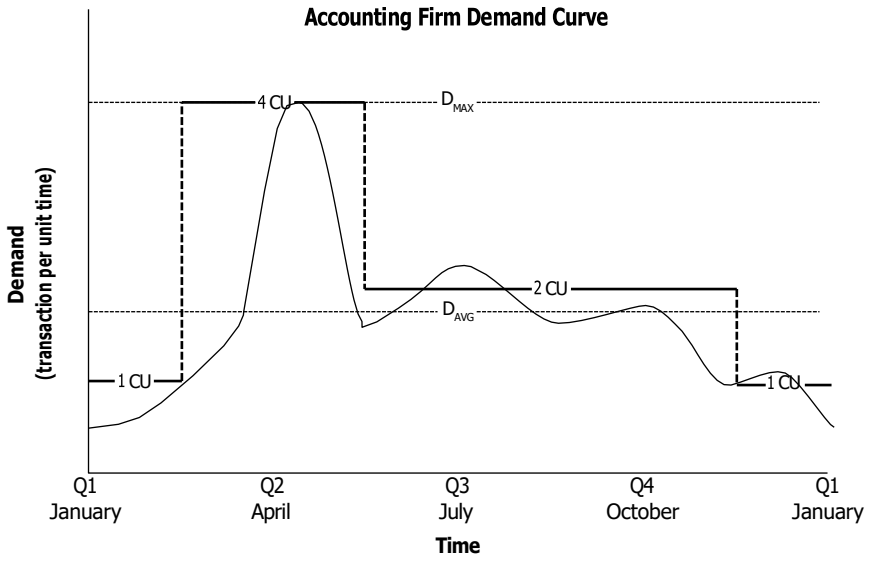
Cloud computing is also a good option when the cost of infrastructure and management is high. This is often the case with legacy applications and systems where maintaining the system capabilities is a significant cost.

Right-sizing

Consider an accounting firm with a variable demand load, as shown in Figure 2.2. For each of the four quarters of the tax year, clients file their quarterly taxes on the service's Web site. Demand for three of those quarters rises broadly as the quarterly filing deadline arrives. The fourth quarter that represents the year-end tax filing on April 15 shows a much larger and more pronounced spike for the two weeks approaching and just following that quarter's end. Clearly, this accounting business can't ignore the demand spike for its year-end accounting, because this is the single most important portion of the firm's business, but it needs to match demand to resources to maximize its profits.

FIGURE 2.2

Right-sizing demand to infrastructure



Buying and leasing infrastructure to accommodate the peak demand (or alternatively load) shown in the figure as D_{MAX} means that nearly half of that infrastructure remains idle for most of the time. Fitting the infrastructure to meet the average demand, D_{AVG}, means that half of the transactions in the

Q2 spike are not captured, which is the mission critical portion of this enterprise. More accurately using D_{AVG} means that during maximum demand the service is slowed to a crawl and the system may not be responsive enough to satisfy any of the users.

These limits can be a serious constraint on profit and revenue. Outsourcing the demand may provide a solution to the problem. But outsourcing essentially shifts the burden of capital expenditures onto the service provider. A service contract that doesn't match infrastructure to demand suffers from the same inefficiencies that captive infrastructure does.

The cloud computing model addresses this problem by allowing you to right-size your infrastructure. In Figure 2.2, the demand is satisfied by an infrastructure that is labeled in terms of a CU or "Compute Unit." The rule for this particular cloud provider is that infrastructure may be modified at the beginning of any month. For the low-demand Q1/Q4 time period, a 1 CU infrastructure is applied. On February 1st, the size is changed to a 4 CU infrastructure, which captures the entire spike of Q2 demand. Finally, on June 1st, a 2 CU size is applied to accommodate the typical demand DAVG that is experienced in the last half of Q2 through the middle of Q4. This curve-fitting exercise captures the demand nearly all the time with little idle capacity left unused.

Note

In reality, the major cloud providers provide machine instances in small slices that can be added within five minutes or less. A standard machine instance (virtual computer) might cost $0.10 or less an hour; a typical storage charge might be $0.10 per GB/month. It is this flexibility that has made cloud computing viable. Past efforts to push cloud computing such as the Intel Computing Services (circa 2000) approach required negotiated contracts and long commitments, which is why this service didn't gain traction. ■

If this deployment represented a single server, then 1 CU might represent a single dual-core processor, 2 CU might represent a quad-core processor, and 4 CU might represent a dual quad-core processor virtual system. Most cloud providers size their systems small, medium, and large in just this manner.

Right-sizing is possible when the system load is cyclical or in some cases when there are predictable bursts or spikes in the load. You encounter cyclical loads in many public facing commercial ventures with seasonal demands, when the load is affected by time zones, and at times that new products launch. Burst loads are less predictable. You can encounter bursts in systems that are gateways or hubs for traffic. In situations where demand is unpredictable and change can be rapid, right-sizing a cloud computing solution demands automated solutions. Amazon Web Services' Auto Scaling feature (`http://aws.amazon.com/autoscaling/`) for its EC2 service described in Chapter 9 is an example of such an automated solution. Shared systems with multiple tenants that need to scale are another example where right-sizing can be applied.

Computing the Total Cost of Ownership

The Total Cost of Ownership or TCO is a financial estimate for the costs of the use of a product or service over its lifetime, often broken out on a yearly or quarterly basis. In pitching cloud

computing projects, it is common to create spreadsheets that predict the costs of using the cloud computing model versus performing the same functions in-house or on-premises.

To be really useful, a TCO must account for the real costs of items, but frequently they do not. For example, the cost of a system deployed in-house is not only the cost of acquisition and the cost of maintenance. A system consumes resources such as space in a datacenter or portion of your site, power, cooling, and management. All these resources represent an overhead that is often overlooked, either in error or for political reasons. When you account for monitoring and management of systems, you must account for the burdened cost of an IT employee, the cost of the hardware and software that is used for management, and other hidden costs.

The Wikipedia page on Total Cost of Ownership (`http://en.wikipedia.org/wiki/Total_cost_of_ownership`) contains a list of computer and software industries TCO elements that are a good place to start to build your own worksheet.

Note

A really thorough and meaningful TCO study should probably be done by an accountant or consultant with a financial background in this area to obtain meaningful results. ■

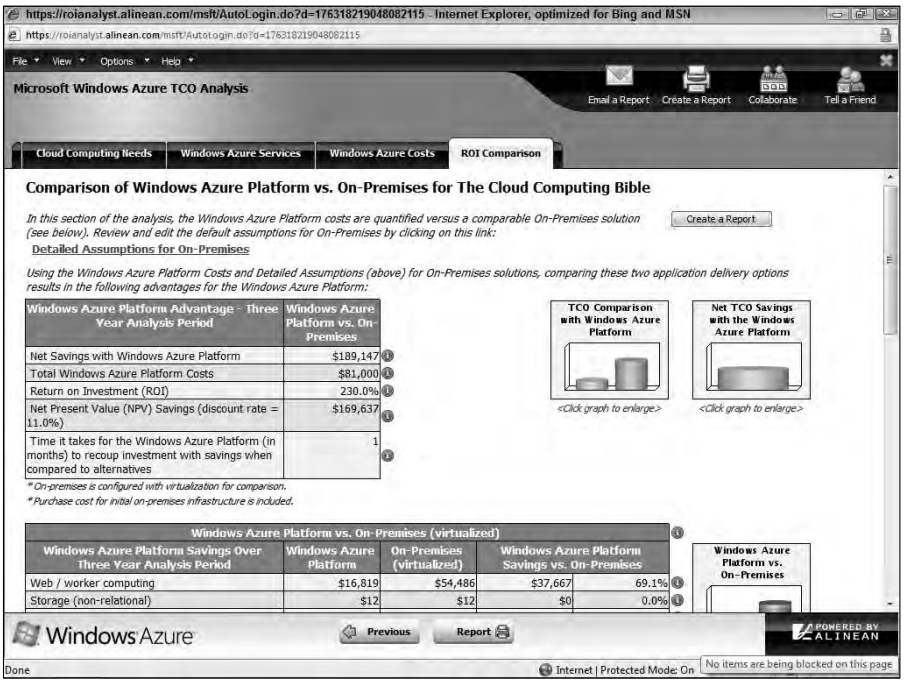
You can find many examples of TCO wizards and worksheets for cloud systems. Microsoft maintains an economics page (`http://www.microsoft.com/windowsazure/economics/#tco_content`) for its Windows Azure Platform. One of the features on the page is a link to a TCO calculator that is based on an engine created by Alinean. In this calculator, you describe your business, its location and industry, and the level of activity (logins and connections) in the first step of the wizard. The Azure TCO Calculator shows you its recommendations for deployment, the Azure costs, and a report of its analysis as the last step of the wizard.

Figure 2.3 shows you a report for a hypothetical company with 10 servers on Azure. You can use this report to show graphs and print the report, and the factors it presents in this report may be useful to you.

Any discussion of Total Cost of Ownership provides an operational look at infrastructure deployment. A better metric for enterprises is captured by a Return on Investment or ROI calculation. To accurately measure an ROI, you need to capture the opportunities that a business has been able to avail itself of (or not), something that is accurate only in hindsight. The flexibility and agility of cloud computing allows a company to focus on its core business and create more opportunities.

FIGURE 2.3

The Microsoft Azure Platform ROI wizard provides a quick and dirty analysis of your TCO for a cloud deployment on Windows Azure in an attractive report format.



Specifying Service Level Agreements

A Service Level Agreement (SLA) is the contract for performance negotiated between you and a service provider. In the early days of cloud computing, all SLAs were negotiated between a client and the provider. Today with the advent of large utility-like cloud computing providers, most SLAs are standardized until a client becomes a large consumer of services.

Caution

Some SLAs are enforceable as contracts, but many are really agreements that are more along the lines of an Operating Level Agreement (OLA) and may not have the force of law. It's good to have an attorney review these documents before you make a major commitment to a cloud provider. ■

SLAs usually specify these parameters:

- Availability of the service (uptime)
- Response times or latency

- Reliability of the service components
- Responsibilities of each party
- Warranties

If a vendor fails to meet the stated targets or minimums, it is punished by having to offer the client a credit or pay a penalty. In this regard, an SLA should be like buying insurance, and like buying insurance, getting the insurer to pay up when disaster strikes can be the devil's work.

Microsoft publishes the SLAs associated with the Windows Azure Platform components at `http://www.microsoft.com/windowsazure/sla/`, which is illustrative of industry practice for cloud providers. Each individual component has its own SLA. The summary versions of these SLAs from Microsoft are reproduced here:

- **Windows Azure SLA:** "Windows Azure has separate SLA's for compute and storage. For compute, we guarantee that when you deploy two or more role instances in different fault and upgrade domains, your Internet facing roles will have external connectivity at least 99.95% of the time. Additionally, we will monitor all of your individual role instances and guarantee that 99.9% of the time we will detect when a role instance's process is not running and initiate corrective action."

Cross-Ref

The different components of the Windows Azure Platform are discussed in detail in Chapter 10. ■

- **SQL Azure SLA:** "SQL Azure customers will have connectivity between the database and our Internet gateway. SQL Azure will maintain a "Monthly Availability" of 99.9% during a calendar month. "Monthly Availability Percentage" for a specific customer database is the ratio of the time the database was available to customers to the total time in a month. Time is measured in 5-minute intervals in a 30-day monthly cycle. Availability is always calculated for a full month. An interval is marked as unavailable if the customer's attempts to connect to a database are rejected by the SQL Azure gateway."
- **AppFabric SLA:** "Uptime percentage commitments and SLA credits for Service Bus and Access Control are similar to those specified above in the Windows Azure SLA. Due to inherent differences between the technologies, underlying SLA definitions and terms differ for the Service Bus and Access Control services. Using the Service Bus, customers will have connectivity between a customer's service endpoint and our Internet gateway; when our service fails to establish a connection from the gateway to a customer's service endpoint, then the service is assumed to be unavailable. Using Access Control, customers will have connectivity between the Access Control endpoints and our Internet gateway. In addition, for both Service Bus and Access Control, the service will process correctly formatted requests for the handling of messages and tokens; when our service fails to process a request properly, then the service is assumed to be unavailable. SLA calculations will be based on an average over a 30-day monthly cycle, with 5-minute time intervals. Failures seen by a customer in the form of service unavailability will be counted for the purpose of availability calculations for that customer."

You can find Google's App Engine for Business SLA at `http://code.google.com/appengine/business/sla.html`. The SLA for Amazon Web Service Elastic Computer Cloud (EC2) is published at `http://aws.amazon.com/ec2-sla/`, and the SLA for Amazon Simple Storage Service (S3) may be found at `http://aws.amazon.com/s3-sla/`.

Some cloud providers allow for service credits based on their ability to meet their contractual levels of uptime. For example, Amazon applies a service credit of 10 percent off the charge for Amazon S3 if the monthly uptime is equal to or greater than 99 percent but less than 99.9 percent. When the uptime drops below 99 percent, the service credit percentage rises to 25 percent and this credit is applied to usage in the next billing period. Amazon Web Services uses an algorithm that calculates uptime based on the following formula:

Uptime = Error Rate/Requests

as measured for each 5-minute interval during a billing period. The error rate is based on internal server counters such as "InternalError" or "ServiceUnavailable." There are exclusions that limit Amazon's exposure.

Service Level Agreements are based on the usage model. Most cloud providers price their pay-as-you-go resources at a premium and issue standard SLAs only for that purpose. You can also purchase subscriptions at various levels that guarantee you access to a certain amount of purchased resources. The SLAs attached to a subscription often offer different terms. If your organization requires access to a certain level of resources, then you need a subscription to a service. A usage model may not provide that level of access under peak load conditions.

Defining Licensing Models

When you purchase shrink-wrapped software, you are using that software based on a licensing agreement called a EULA or End User License Agreement. The EULA may specify that the software meets the following criteria:

- It is yours to own.
- It can be installed on a single or multiple machines.
- It allows for one or more connections.
- It has whatever limit the ISV has placed on its software.

In most instances, the purchase price of the software is directly tied to the EULA.

For a long time now, the computer industry has known that the use of distributed applications over the Internet was going to impact the way in which companies license their software, and indeed it has. The problem is that there is no uniform description of how applications accessed over cloud networks will be priced. There are several different licensing models in play at the moment—and no clear winners.

It can certainly be argued that the use of free software is a successful model. The free use of software in the cloud is something that the open-source community can support, it can be supported as a line extension of a commercial product, or it can be paid for out of money obtained from other sources such as advertising. Google's advertising juggernaut has allowed it to create a portfolio of applications that users can access based on their Google accounts. Microsoft's free software in Windows Live is supported by its sales of the Windows operating system and by sales of the Microsoft Office suite.

In practice, cloud-based providers tend to license their applications or services based on user or machine accounts, but they do so in ways that are different than you might expect based on your experience with physical hardware and software. Many applications and services use a subscription or usage model (or both) and tie it to a user account. Lots of experimentation is going on in the publishing industry on how to price Internet offerings, and you can find the same to be true in all kinds of computer applications at the moment. Some services tie their licenses into a machine account when it makes sense. An example is the backup service Carbonite, where the service is for backing up a licensed computer. However, cloud computing applications rarely use machine licenses when the application is meant to be ubiquitous. If you need to access an application, service, or Web site from any location, then a machine license isn't going to be practical.

The impact of cloud computing on bulk software purchases and enterprise licensing schemes is even harder to gauge. Several analysts have remarked that the advent of cloud computing could lead to the end of enterprise licensing and could cause difficulties for software vendors going forward. It isn't clear what the impact on licensing will be in the future, but it is certainly an area to keep your eyes on.

Summary

In this chapter, you learned about the features that make cloud computing unique. It is both a new model and a new platform for computing. The idea of computing as a utility is as old as the computer industry itself, but it is the advent of low-cost datacenters that have really enabled this platform to thrive.

A cloud's unique features are scalability, elasticity, low barrier to entry, and a utility delivery of services. These features completely change the way in which applications and services are used. Cloud computing will enable the development of new types of applications, several of which were described in this chapter. Undoubtedly many new application types will arise that will be a complete surprise to us all. However, cloud computing has some limitations that were discussed as well.

To help you get a handle on cloud computing, this chapter stresses measurements of costs in comparison to private or on-premises systems. Cloud computing shifts capital expenditures into operating expenditures.

Cloud computing changes the nature of the service provider and its relationship to its client. You see this expressed in the Service Level Agreements (SLAs) and software licensing that are part of this new developing industry. There are many changes to come in these areas.

Chapter 3, "Understanding Cloud Architecture," is an in-depth look at the superstructure and plumbing that makes cloud computing possible. In that chapter, you learn about the standards and protocols used by the cloud computing industry.

CHAPTER 3

Understanding Cloud Architecture

Cloud computing is a natural extension of many of the design principles, protocols, plumbing, and systems that have been developed over the past 20 years. However, cloud computing describes some new capabilities that are architected into an application stack and are responsible for the programmability, scalability, and virtualization of resources. One property that differentiates cloud computing is referred to as composability, which is the ability to build applications from component parts.

A platform is a cloud computing service that is both hardware and software. Platforms are used to create more complex software. Virtual appliances are an important example of a platform, and they are becoming a very important standard cloud computing deployment object.

Cloud computing requires some standard protocols with which different layers of hardware, software, and clients can communicate with one another. Many of these protocols are standard Internet protocols. Cloud computing relies on a set of protocols needed to manage interprocess communications that have been developed over the years. The most commonly used set of protocols uses XML as the messaging format, the Simple Object Access Protocol (SOAP) protocol as the object model, and a set of discovery and description protocols based on the Web Services Description Language (WSDL) to manage transactions.

Some completely new clients are under development that are specifically meant to connect to the cloud. These clients have as their focus cloud applications and services, and are often hardened and more securely connected. Two examples presented are Jolicloud and Google Chrome OS. They represent a new client model that is likely to have considerable impact.

IN THIS CHAPTER

- Using the cloud computing stack to describe different models
- Understanding how platforms and virtual appliances are used
- Learning how cloud communications work
- Discovering the new world of the cloud client

Exploring the Cloud Computing Stack

Cloud computing builds on the architecture developed for staging large distributed network applications on the Internet over the last 20 years. To these standard networking protocols, cloud computing adds the advances in system virtualization that became available over the last decade. The cloud creates a system where resources can be pooled and partitioned as needed. Cloud architecture can couple software running on virtualized hardware in multiple locations to provide an on-demand service to user-facing hardware and software. It is this unique combination of abstraction and metered service that separates the architectural requirements of cloud computing systems from the general description given for an n-tiered Internet application.

Many descriptions of cloud computing describe it in terms of two architectural layers:

A client as a front end

The "cloud" as a backend

This is a very simplistic description because each of these two components is composed of several component layers, complementary functionalities, and a mixture of standard and proprietary protocols. Cloud computing may be differentiated from older models by describing an encapsulated information technology service that is often controlled through an Application Programming Interface (API), thus modifying the services that are delivered over the network.

A cloud can be created within an organization's own infrastructure or outsourced to another datacenter. While resources in a cloud can be real physical resources, more often they are virtualized resources because virtualized resources are easier to modify and optimize. A compute cloud requires virtualized storage to support the staging and storage of data. From a user's perspective, it is important that the resources appear to be infinitely scalable, that the service be measurable, and that the pricing be metered.

Composability

Applications built in the cloud often have the property of being built from a collection of components, a feature referred to as composability. A composable system uses components to assemble services that can be tailored for a specific purpose using standard parts. A composable component must be:

- **Modular:** It is a self-contained and independent unit that is cooperative, reusable, and replaceable.
- **Stateless:** A transaction is executed without regard to other transactions or requests.

It isn't an absolute requirement that transactions be stateless, some cloud computing applications provide managed states through brokers, transaction monitors, and service buses. In rarer cases, full transactional systems are deployed in the clouds, but these systems are harder to architect in a distributed architecture.

Although cloud computing doesn't require that hardware and software be composable, it is a highly desirable characteristic from a developer or user's standpoint, because it makes system design easier to implement and solutions more portable and interoperable.

There is a tendency for cloud computing systems to become less composable for users as the services incorporate more of the cloud computing stack. From the standpoint of an IaaS (Infrastructure as a Service) vendor such as Amazon Web Services, GoGrid, or Rackspace, it makes no sense to offer non-standard machine instances to customers, because those customers are almost certainly deploying applications built on standard operating systems such as Linux, Windows, Solaris, or some other well-known operating system.

In the next step up the cloud computing stack, PaaS (Platform as a Service) vendors such as Windows Azure or Google AppEngine may narrow the definition of standard parts to standard parts that work with their own platforms, but at least from the standpoint of the individual platform service provider, the intent is to be modular for their own developers.

When you move to the highest degree of integration in cloud computing, which is SaaS (Software as a Service), the notion of composability for users may completely disappear. An SaaS vendor such as Quicken.com or Salesforce.com is delivering an application as a service to a customer, and there's no particular benefit from the standpoint of the service provider that the customer be able to compose its own custom applications. A service provider reselling an SaaS may have the option to offer one module or another, to customize the information contained in the module for a client, to sell the service under their own brand, or to perform some other limited kind of customization, but modifications are generally severely limited.

This idea that composability diminishes going up the cloud computing stack is from the user's point of view. If you are a PaaS or SaaS service provider and your task is to create the platform or service presented to the developer, reseller, or user, the notion of working with a composable system is still a very powerful one. A PaaS or SaaS service provider gets the same benefits from a composable system that a user does—these things, among others:

- Easier to assemble systems
- Cheaper system development
- More reliable operation
- A larger pool of qualified developers
- A logical design methodology

You encounter the trend toward designing composable systems in cloud computing in the widespread adoption of what has come to be called the Service Oriented Architecture (SOA). The essence of a service oriented design is that services are constructed from a set of modules using standard communications and service interfaces. An example of a set of widely used standards describes the services themselves in terms of the Web Services Description Language (WSDL), data exchange between services using some form of XML, and the communications between the services using the SOAP protocol. There are, of course, alternative sets of standards.

Cross-Ref

SOA is described in detail in Chapter 13. ■

What isn't specified is the nature of the module itself; it can be written in any programming language the developer wants. From the standpoint of the system, the module is a black box, and only the interface is well specified. This independence of the internal workings of the module or component means it can be swapped out for a different model, relocated, or replaced at will, provided that the interface specification remains unchanged. That is a powerful benefit to any system or application provider as their products evolve.

Infrastructure

Most large Infrastructure as a Service (IaaS) providers rely on virtual machine technology to deliver servers that can run applications. Virtual servers described in terms of a machine image or instance have characteristics that often can be described in terms of real servers delivering a certain number of microprocessor (CPU) cycles, memory access, and network bandwidth to customers. Virtual machines are containers that are assigned specific resources. The software that runs in the virtual machines is what defines the utility of the cloud computing system.

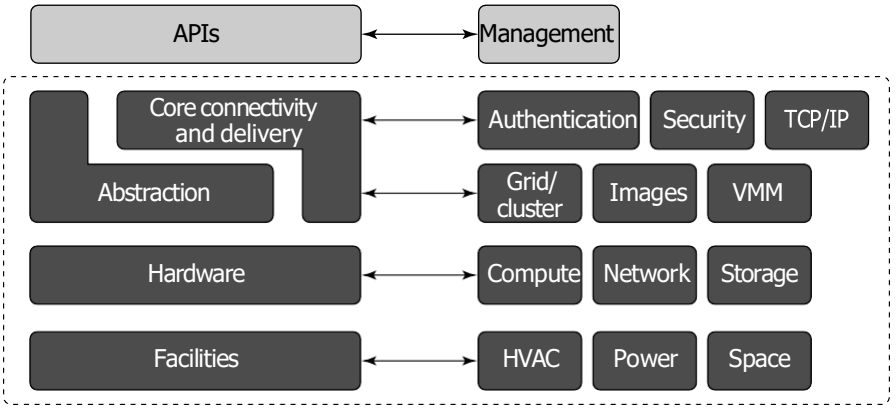
Figure 3.1 shows the portion of the cloud computing stack that is defined as the "server." In the diagram, the API is shown shaded in gray because it is an optional component that isn't always delivered with the server. The VMM component is the Virtual Machine Monitor, also called a hypervisor. This is the low-level software that allows different operating systems to run in their own memory space and manages I/O for the virtual machines.

The notion of a virtual server presents to an application developer a new way of thinking about and programming applications. For example, when a programmer is creating software that requires several different tasks to be performed in parallel, he might write an application that creates additional threads of execution that must be managed by the application. When a developer creates an application that uses a cloud service, the developer can attach to the appropriate service(s) and allow the application itself to scale the program execution. Thus, an application such as a three-dimensional rendering that might take a long time for a single server to accomplish can be scaled in the cloud to many servers at once for a short period of time, accomplishing the task at a similar or lower price but at a much faster rate.

In future applications, developers will need to balance the architectural needs of their programs so their applications create new threads when it is appropriate or create new virtual machines. Applications will also need to be mindful of how they use cloud resources, when it is appropriate to scale execution to the cloud, how to monitor the instances they are running, and when not to expand their application's usage of the cloud. This will require a new way of thinking about application development, and the ability to scale correctly is something that will have to be architected into applications from the ground up.

FIGURE 3.1

This architectural diagram illustrates the portion of the cloud computing stack that is designated as the server.



Platforms

A platform in the cloud is a software layer that is used to create higher levels of service. As you learned in Chapter 1, many different Platform as a Service (PaaS) providers offer services meant to provide developers with different capabilities. In Chapter 7, PaaS is explored more thoroughly, but for now it is useful to cite three of the major examples that are provided in this book:

- Salesforce.com's Force.com Platform
- Windows Azure Platform
- Google Apps and the Google AppEngine

These three services offer all the hosted hardware and software needed to build and deploy Web applications or services that are custom built by the developer within the context and range of capabilities that the platform allows. Platforms represent nearly the full cloud software stack, missing only the presentation layer that represents the user interface. This is the same portion of the cloud computing stack that is a virtual appliance and is shown in Figure 3.2. What separates a platform from a virtual appliance is that the software that is installed is constructed from components and services and controlled through the API that the platform provider publishes.

It makes sense for operating system vendors to move their development environments into the cloud with the same technologies that have been successfully used to create Web applications. Thus, you might find a platform based on a Sun xVM hypervisor virtual machine that includes a NetBeans Integrated Development Environment (IDE) and that supports the Sun GlassFish

Web stack programmable using Perl or Ruby. For Windows, Microsoft would be similarly interested in providing a platform that allowed Windows developers to run on a Hyper-V VM, use the ASP.NET application framework, support one of its enterprise applications such as SQL Server, and be programmable within Visual Studio—which is essentially what the Azure Platform does. This approach allows someone to develop a program in the cloud that can be used by others.

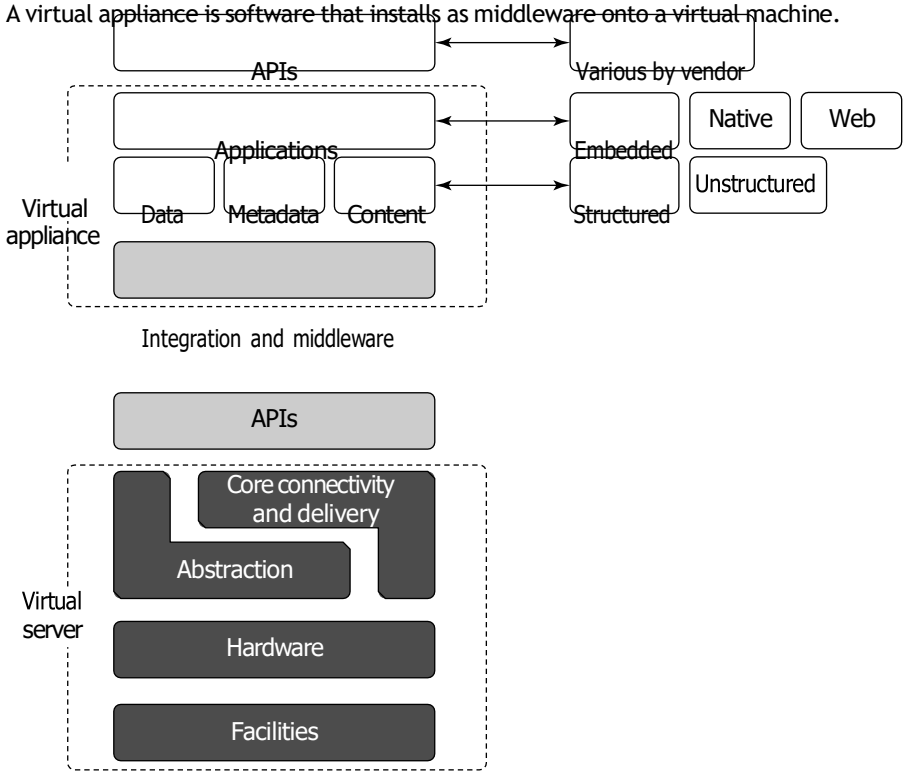
Platforms often come replete with tools and utilities to aid in application design and deployment. Depending upon the vendor, you may find developer tools for team collaboration, testing tools, instrumentation for measuring program performance and attributes, versioning, database and Web service integration, and storage tools. Most platforms begin by establishing a developer community to support the work done in the environment.

Note

To see the entire cloud computing stack, refer to Figure 1.5 in Chapter 1. ■

FIGURE 3.2

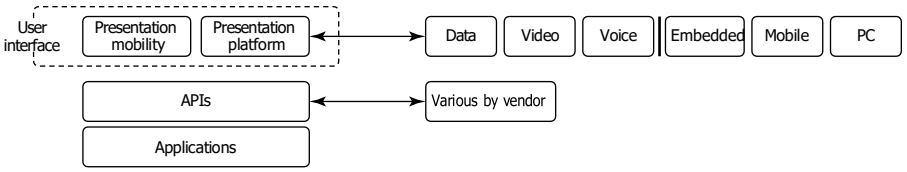
A virtual appliance is software that installs as middleware onto a virtual machine.



Just as a virtual appliance may expose itself to users through an API, so too an application built in the cloud using a platform service would encapsulate the service through its own API. Users would then interact with the platform, consuming services through that API, leaving the platform to manage and scale the service appropriately. Many platforms offer user interface development tools based on HTML, JavaScript, or some other technology. As the Web becomes more media-oriented, many developers have chosen to work with rich Internet environments such as Adobe Flash, Flex, or Air, or alternatives such as Windows Silverlight. A user interface abstracts away the platform API, making those services managed through the UI. Figure 3.3 shows the top portion of the cloud computing stack, which includes the API and the presentation functionality.

FIGURE 3.3

The top of the cloud computing interface includes the user interface and the API for the application layer.



The Application Programming Interface is one of the key differentiators separating cloud computing from the older models of Internet applications, because it is the means for instantiating resources needed to support applications. An API can control data flow, communications, and other important aspects of the cloud application. Unfortunately, each cloud vendor has their own cloud API, none of them are standard, and the best you can hope for is that eventually the major cloud vendor's APIs will interoperate and exchange data. For now, the use of proprietary APIs results in vendor lock-in, which is why you are advised to choose systems that implement APIs based on open standards.

Virtual Appliances

Applications such as a Web server or database server that can run on a virtual machine image are referred to as virtual appliances. The name *virtual appliance* is a little misleading because it conjures up the image of a machine that serves a narrow purpose. Virtual appliances are software installed on virtual servers—application modules that are meant to run a particular machine instance or image type. A virtual appliance is a platform instance. Therefore, virtual appliances occupy the middle of the cloud computing stack (refer to Figure 3.2).

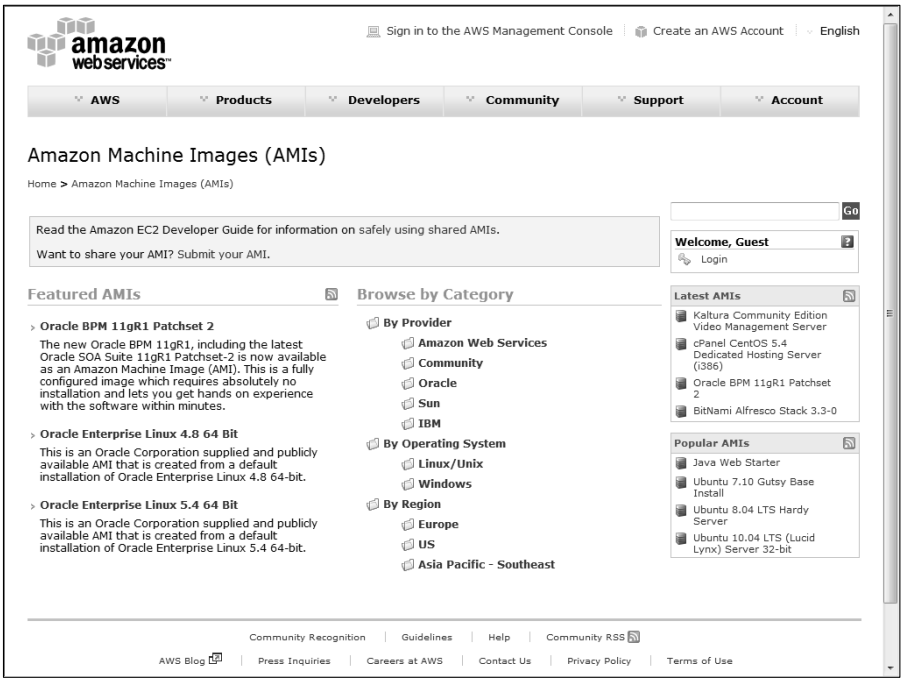
A virtual appliance is a common deployment object in the cloud, and it is one area where there is considerable activity and innovation. One of the major advantages of a virtual appliance is that you can use the appliances as the basis for assembling more complex services, the appliance being one of your standardized components. Virtual appliances remove the need for application configuration and maintenance from your list of system management chores.

You run across virtual appliances in IaaS systems such as Amazon's Elastic Compute Cloud (EC2), which is discussed in detail in Chapter 9. Amazon Machine Images are virtual appliances that have been packaged to run on the gird of Xen nodes that comprise the Amazon Web Service's EC2 system. Shown in Figure 3.4, the AMI library (`http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=171`) includes a variety of operating systems both proprietary and open source, a set of enterprise applications such as Oracle BPM, SQL Server, and even complete application stacks such as LAMP (Linux, Apache, MySQL, and PHP). Amazon has negotiated licenses from these vendors that are part of your per-use pricing when you run these applications on their servers.

Virtual appliances are far easier to install and run than an application that you must set up yourself. However, virtual appliances are also much larger than the application themselves would be because they are usually bundled with the operating system on which they are meant to run. An application that is 50 or 100MB might require a virtual appliance that is 500MB to 1GB in size. Usually, when a virtual appliance is created, the operating system is stripped of all excess functionality that isn't required by the appliance, because the appliance is meant to be used as is.

FIGURE 3.4

Amazon Machine Images are a collection of virtual appliances that you can install on their Xen hypervisor servers.



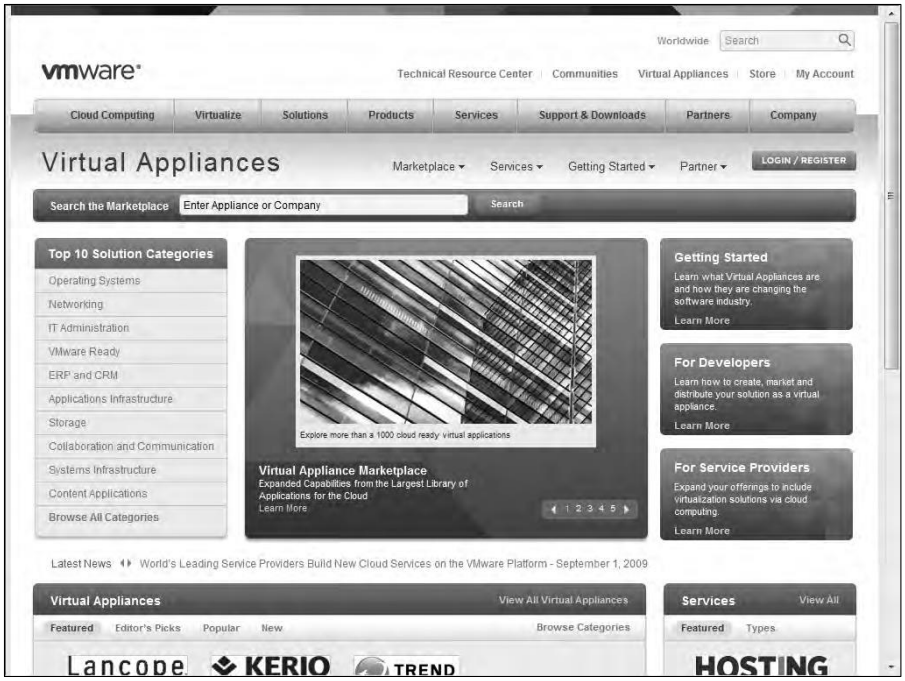
Virtual appliances have begun to affect the PC industry in much the same way that application stores have affected the cell phone industry. You can find various Web sites that either sell or distribute ready-to-use virtual appliances in various forms. Perhaps the best developed of these marketplaces is VMware's Virtual Appliances site (`http://www.vmware.com/appliances/`) shown in Figure 3.5. These appliances are certified by VMware to be ready to use in the enterprise.

Among the other places you can find virtual appliances are at the Web sites of the various operating system vendors, such as Ubuntu, Xen (`http://www.xen.org/`), and others, including these:

- **Bagvapp** (`http://bagside.com/bagvapp/`) offers virtual appliances, including ones based on Windows, all of which run on VMware Player.
- **HelpdeskLive** (`http://helpdesklive.info/download/VirtualBox%20VDI%20free%20images.html`) offers various Linux distributions upon which you can build a virtual machine.
- **Jcinacio** (`http://www.jcinacio.com/`) has Ubuntu appliances.

FIGURE 3.5

VMware's Virtual Appliance marketplace (`http://www.vmware.com/appliances/`) sells virtual appliances that run on VMware's hypervisor in cloud computing applications.



- **Jumpbox** (`http://www.jumpbox.com`) offers open source virtual appliances installed by them as a managed service. Jumpbox offers virtual appliances for many applications including Bugzilla, DokuWiki, Drupal, Joomla!, Nagios, OpenVPN, PostgreSQL, Redmine, WordPress, and many others. Figure 3.6 shows the Jumpbox home page.
- **QEMU** (`http://www.qemu.org/`) is a CPU emulator and virtual machine monitor.
- **Parallels** (`http://ptn.parallels.com/ptn`) hosts a variety of appliances that includes Linux distros, server software, and other products.
- **ThoughtPolice** (`http://www.thoughtpolice.co.uk/vmware/`) offers appliances based on a variety of Linux distributions.
- **VirtualBox** (`http://www.virtualbox.org/`) is a virtual machine technology now owned by Oracle that can run various operating systems and serves as a host for a variety of virtual appliances.
- **Vmachines** (`http://www.vmachines.net/`) is a site with desktop, server, and security-related operating systems that run on VMware.

FIGURE 3.6

Jumpbox (`http://www.jumpbox.com/`) is an open-source virtual appliance installation and management service.

Converting a virtual appliance from one platform to another isn't an easy proposition. Efforts are underway to create file format standards for these types of objects that make this task easier. The best known of these file formats is the Open Virtualization Format (OVF), the work of the Distributed Management Task Force (DMTF) group. Nearly all major virtualization platform vendors support OVF, notably VMware, Microsoft, Oracle, and Citrix.

Cross-Ref

Chapter 5 describes virtual appliances in more detail and gives some examples of vendors offering these types of applications for cloud computing infrastructure development. ■

Communication Protocols

Cloud computing arises from services available over the Internet communicating using the standard Internet protocol suite underpinned by the HTTP and HTTPS transfer protocols. The other protocols and standards that expose compute and data resources in the cloud either format data or communications in packets that are sent over these two transport protocols.

In order to engage in interprocess communication (IPC) processes, many client/server protocols have been applied to distributed networking over the years. Various forms of RPC (Remote Procedure Call) implementations (including DCOM, Java RMI, and CORBA) attempt to solve the problem of engaging services and managing transactions over what is essentially a stateless network. The first of the truly Web-centric RPC technologies was XML-RPC, which uses platform-independent XML data to encode program calls that are transported over HTTP, the networking transport to which nearly everyone is connected.

Note

You can find a full description of the common Internet protocol standards in *Networking Bible* by Barrie Sosinsky, Wiley, 2009. These protocols form the basis for much of the discussion in any good networking textbook. ■

As Internet computing became more firmly entrenched over the last decade, several efforts began to better define methods for describing and discovering services and resources. The most widely used message-passing standard at the moment is the Simple Object Access Protocol (SOAP), which essentially replaces XML-RPC. SOAP uses XML for its messages and uses RPC and HTTP for message passing. SOAP forms the basis for most of the Web services stacks in use today. If you examine the XML file used in a SOAP transaction, you find that it contains a message and the instructions on how to use the message. The message has a set of rules that are translated into application instances and datatypes, and it defines the methods that must be used to initiate procedure calls and then return a response.

Several standards have emerged to allow the discovery and description of Web-based resources. The most commonly used model for discovery and description used with SOAP messaging is the Web Services Description Language (WSDL), a World Wide Web Consortium (`http://www.w3.org/2002/ws/desc/`) Internet standard. WSDL lets a Web service advertise itself in terms of a collection of endpoints or ports associated with a specific network address (URL) that can be

addressed using XML messages to provide a service. In WSDL, a service is a container that performs a set of functions that are exposed to Web protocols. Taken together, the protocol and port are a binding to which messages are passed and operations are performed. A bound service is one that responds to any valid HTTP request sent to it. The important thing to remember about WSDL is that it defines a Web service's public interface.

Cross-Ref

Chapter 13, which describes Service Oriented Architecture, continues the discussion of these various components in building an application in the cloud from a component-based architecture. ■

Using WSDL and SOAP, a number of extensions were created that allow various Web services to describe additional sets of properties and methods that they could provide. These extensions fall under the name WS-*, or the "WS-star" specifications. A number of WS-* extensions are in common use, with the following being the most widely used:

- WS-Addressing
- WS-Discovery
- WS-Eventing
- WS-Federation
- WS-MakeConnection
- WS-Messaging
- WS-MetadataExchange
- WS-Notification
- WS-Policy
- WS-ResourceFramework
- WS-Security
- WS-Transfer
- WS-Trust

These different specifications provide a standard means of adding metadata to a SOAP message by modifying the message header while maintaining the message body structure. In this way, a standard method for metadata exchange is piggybacked onto the WSDL XML message. Each of these different WS-* specifications is in a different state of development.

You use these various WS-* services in your daily work. For example, the Web Services Dynamic Discovery specification (WS-Discovery) is a specification for multicast discovery on a LAN (Local Area Network) that is extended to Web services, most often as SOAP over UDP (User Datagram Protocol). When you open the Network Neighborhood in Windows and use the People Near Me feature, WS-Discovery goes into action and shows you discoverable resources.

These WS-* services carried over XML messages using the SOAP protocol access remote server applications in ways that are becoming increasingly complex. Whereas earlier methods for client/server provided a means through a gateway like CGI to access media content on servers, the current data communications burden servers with accepting and processing very complex requests or engaging their clients in sophisticated negotiations that seek to minimize the amount of processing that must be done and the information that must be exchanged as the response. None of this type of rich media servicing was ever envisaged in the construction of the Internet, and all of it is essentially a kludge.

Over the years, a variety of platform-specific RPC specifications, such as DCOM (Distributed Common Object Model) and CORBA (Common Object Response Broker Architecture), were developed to allow software components that ran on different computers to interoperate with one another. As SOAP and WS-* were developing, those protocols began to build into their specifications server application features from these other technologies in a more platform-independent protocol. What was really needed was a method for standardizing resources on the Web, which is where the idea of REST comes in.

REST stands for Representational State Transfer, and it owes its original description to the work of Roy Fielding, who was also a co-developer of the HTTP protocol. REST assigns a global identifier to a resource so there is a uniform method for accessing information sources. That identifier is a URI expressed in HTTP form. Given a resource then at a known address, various network clients in the form of what are called *user agents* can then communicate with that resource using HTTP commands (requests) to exchange information in the form of documents or files. Typical data transfers might use XML, text, an image file, a JSON document, or some other standard or agreed upon format to perform the data exchange. A transaction following the rules of REST is therefore considered to be RESTful, and this is the basis for cloud computing transactions to be initiated, processed, and completed in most modern implementations.

While REST is heavily used, it is not the only data interchange standard that is used by cloud services. Another example of a data exchange standard is the Atom or Atom Publishing Protocol (APP). Atom is a syndication format that allows for HTTP protocols to create and update information. Microsoft's ADO.NET Data Services Framework is another system for transferring data using a RESTful transaction and standard HTTP commands.

Cloud services span the gamut of computer applications: audio and video streaming, instant messaging, and so forth. Each of these areas uses protocols developed for network use and adapted for use by Web services. The impact of cloud computing on network communication is to encourage the use of open-network protocols in place of proprietary protocols.

For example, in the area of instant messaging, which is a major cloud service, the SIMPLE protocol (which stands for the Session Initiated Protocol for Instant Messaging and Presence Leveraging Extensions) is in widespread use today. SIMPLE is based on the IETF Session Initiation Protocol (SIP) standard and is an open standard. Although most early IM (Instant Messaging) services used proprietary standards, many IM services now support SIMPLE because without this support, it would be hard for the different services to interoperate. Similarly, in the area of VoIP, you find that the open XMPP or the Extensible Messaging and Presence Protocol is used.

Applications

Although the cloud computing stack encompasses many details that describe how clouds are constructed, it is not a perfect vehicle for expressing all the considerations that one must account for in any deployment. An important omission arises from the nature of distributed Web applications and the design of Internet protocols as a stateless service. The Internet was designed to treat each request made to a server as an independent transaction. Therefore, the standard HTTP commands are all atomic in nature: `GET` to read data, `PUT` to writedata, and so on.

While stateless servers are easier to architect and stateless transactions are more resilient and can survive outages, much of the useful work that computer systems need to accomplish are stateful. Here's the classic example. When you go to a reservation system to purchase something, you query inventory, reserve the item, and then pay for it. In a multiuser system, if you don't have a stateful system, you cannot know whether the item you reserved has already been taken by another user before you can enter your payment for the item. Should you decide you don't want the item at some later time, it is much easier to restore the item to inventory and return payments or make other adjustments if you can roll back all the steps as a transactional unit.

Much of the really hard development efforts that have gone into making the Web useful in commerce have centered around creating mechanisms to change a set of stateless transactions into stateful ones. The development of transaction servers, message queuing servers, and other middleware is meant to bridge this problem. Cloud computing is no exception to this problem, and to an extent it amplifies the problem by not only making transactions stateless but also virtualizing resources so transactions are always occurring in physically different locations. In cloud computing, a variety of constructs are brought to bear to solve these issues, but these are the two most important concepts:

- The notion of orchestration—that process flow can be choreographed as a service
- The use of what is referred to as a service bus that controls cloud components

In time, other methods for establishing transactional integrity may be developed that are better suited to cloud computing, but these are the standard methods that are now part of the Service Oriented Architecture. Chapter 13 takes up this topic in more detail, but it is something you should certainly keep in mind as you read through the intervening chapters.

Connecting to the Cloud

Clients can connect to a cloud service in a number of different ways. These are the two most common means:

- A Web browser
- A proprietary application

These applications can be running on a server, a PC, a mobile device, or a cell phone. What these devices have in common with either of these application types is that they are exchanging data over an inherently insecure and transient medium. There are three basic methods for securely connecting over a connection:

- Use a secure protocol to transfer data such as SSL (HTTPS), FTPS, or IPsec, or connect using a secure shell such as SSH to connect a client to the cloud.
- Create a virtual connection using a virtual private network (VPN), or with a remote data transfer protocol such as Microsoft RDP or Citrix ICA, where the data is protected by a tunneling mechanism.
- Encrypt the data so that even if the data is intercepted or sniffed, the data will not be meaningful.

The best client connections use two or more of these techniques to communicate with the cloud. In current browser technology, clients rely on the Web service to make available secure connections, but in the future, it is likely that cloud clients will be hardened so the client itself enforces a secure connection.

If you've ever logged into a hotel connection and browsed the network, you may find that often you can access systems on the network that haven't been protected with a firewall; an improperly configured firewall connection to the cloud is even worse. That has led people to drag portable routers with them, which provide a personal hardware firewall; many of these devices have VPN built directly into them.

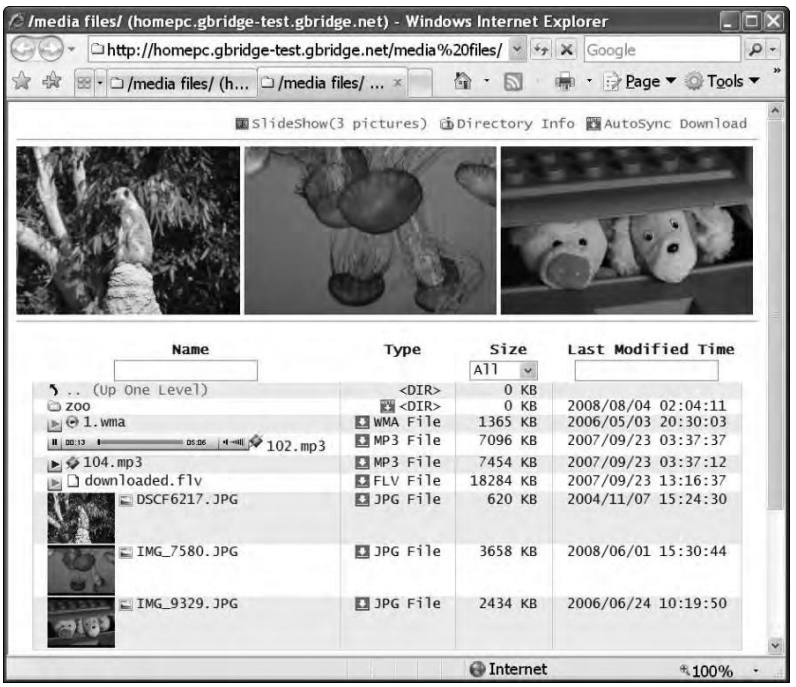
Other solutions include using VPN software; here are three recommended solutions:

- Hotspot VPN (`http://www.hotspotvpn.com/`)
- AnchorFree Hotspot Shield (`http://hotspotshield.com/`)
- Gbridge (`http://www.gbridge.com/`), a third-party VPN based on Google's GoogleTalk infrastructure

Gbridge is an interesting solution that illustrates the use of VPN over a cloud connection. To use this product, you need to log into the GoogleTalk (or Gtalk) network and connect to another computer using your Google account. Gbridge allows additional people to join a connection when invited and supports collaborative features such as desktop sharing using the Virtual Network Computing (VNC) software, chat, live folder browsing, folder synchronization, and automated backup. Gbridge also works with applications deployed using Google Apps, allowing you to securely connect to these applications using a VPN. Figure 3.7 shows browsing a folder over a VPN connection using Gbridge's SecureShares feature.

FIGURE 3.7

Gbridge provides a means for securely connecting one computer to another using Gtalk. Shown here is the SecureShares folder-browsing feature.



The Jolicloud Netbook OS

The popularity of ultralight netbooks and mobile phones has greatly expanded the potential audience of dedicated cloud computing devices, but until recently these devices ran standard operating systems such as Windows, Linux, and Macintosh on the PC and Android, IOS, and Windows Mobile (among others) on cell phones. The primary differentiation between these devices is whether or not they are capable of running video and animation (particularly Adobe Flash).

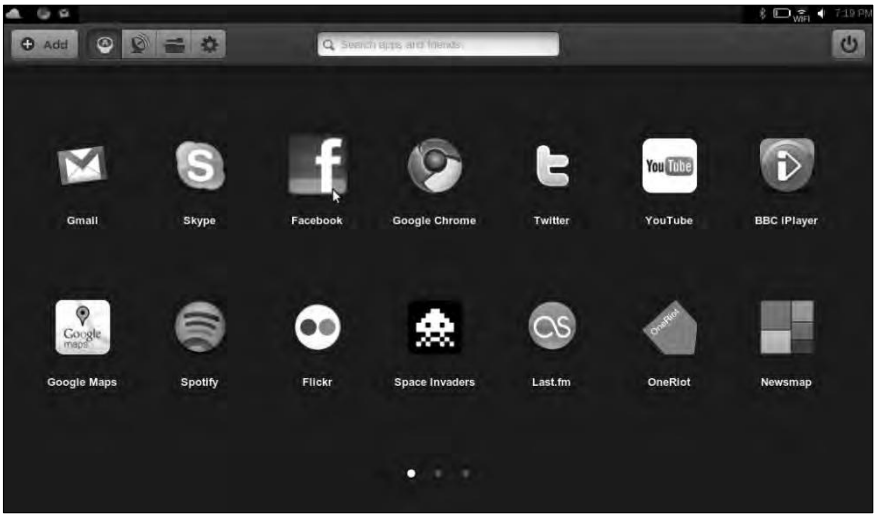
None of these portable devices has been optimized to connect securely to the cloud by narrowing the operating system functions to harden these devices. The French firm Jolicloud (`http://www.jolicloud.com/`) has recently released a lightweight version of Linux designed specifically to run connected to the cloud as a dedicated cloud client. Jolicloud 1.0 ("The Anywhere OS") can be loaded onto a netbook as the only operating system, or it can be set up as a dual boot system that shares files with a Windows partition.

Jolicloud concentrates on building a social platform with automatic software updates and installs. The application launcher is built in HTML 5 and comes preinstalled with Gmail, Skype, Twitter,

Firefox, and other applications. Any HTML 5 browser can be used to work with the Jolicloud interface. Jolicloud maintains a library or App Directory of over 700 applications as part of an app store. When you click to select an application, the company both installs and updates the application going forward, just as the iPhone manages applications on that device. Figure 3.8 shows the Jolicloud interface.

FIGURE 3.8

The Jolicloud cloud client operating system is a social networking platform for netbooks with a dedicated application store.



When you install Jolicloud on multiple devices, the system automatically synchronizes your applications so you are working with the same content on all your devices. You can manage your devices from any cloud-connected device. Your files are also unified in a single location, and the operating system provides access to shared storage cloud services such as box.net, Dropbox, and drop.io, among other services.

Chromium OS: The Browser as an Operating System

The Google Chrome OS is a Linux open-source operating system designed to be a robust cloud client. Unlike many other Linux distributions, Google's Chrome is not a software installation, but is shipped installed on validated hardware from Google-approved OEMs (Original Equipment Manufacturers), just as the Android operating system is shipped on a variety of phones. The intent is to have a tightly coupled hardware offering that supports features in the Chrome OS and that would be highly efficient. Early designs have shown Chrome running on tablet designs that would position it as an Apple iPad competitor running on netbook-type devices in the $300-400 range.

Note

An OEM or original equipment manufacturer builds systems from components and sells them under a brand name. ■

The expectation is that the first versions of Chrome systems will appear in late 2010, perhaps in both consumer and enterprise offerings. There is also an open-source version of this cloud client called Chromium (`http://www.chromium.org/chromium-os`), which shares the same code base. The Chromium architecture is built as a three-tier system with a hardware layer, the browser and window manager, and a set of system software and utilities.

The Chrome OS has been described as a hardened operating system because it incorporates a sandbox architecture for running applications and also performs automatic updates. Also included in the system is a version of remote desktop connection software that creates an encrypted connection like Microsoft's RDP, Citrix's ICA, or a VNC client. The Chrome OS hardware specification includes a Trusted Platform Module, which provides for a "trusted bootpath" along with a hardware switch that can be used to boot the system into a developer model. In that mode, some of the security features are turned off, allowing the user to reset the system.

Demonstrations of early prototypes of Chrome and Chromium OS systems have shown that they are capable of nearly instantaneous startup. Chromium Linux kernel has adopted the Upstart (`http://upstart.ubuntu.com/`) event-based replacement for the `init` daemon, which is used to launch services concurrently, restore stalled jobs, and perform delayed system startup. The fast boot time is possible because the device is devoid of most of the devices in modern PCs. Chromium has also adopted a set of security routines in firmware that run during startup and store the information necessary to perform verified system restoration.

Essentially, the Chrome OS looks like the Chrome browser. Chrome is interesting because Google has essentially stripped down the operating system to run one specific application that connects to the Internet. The user interface is similar to the Chrome Web browser and includes a media player that plays MP3, views JPEGs, and plays media content both online and offline. Adobe Flash is integrated directly into the Chrome OS, just as it is in the Chrome browser. When you launch Chrome, you see links to the major Google cloud applications such as Gmail, Google Apps, and YouTube, along with other major sites such as Facebook, Hulu, Pandora, Twitter, and others. Figure 3.9 shows an early demonstration of the Chrome OS, its multi-tab interface, and its application launcher utility. From this same demonstration is shown the Google Reader application with a page from *Alice in Wonderland*, displayed in Figure 3.10.

Google will include on Chromium the Google Cloud Print service that allows an application to print to any connected printer without accessing a printer driver. This system frees Chromium from having to develop hardware- and OS-specific print subsystems. Instead, a proxy is installed in Chrome that registers a printer with the service, and this proxy manages print jobs for the user.

The Chrome OS devices that appear, as well as the competitors such as the Apple iPad and a host of other similar devices from other system vendors, signal a sea change in the manner in which users access the cloud, and they represent the cloud's impact on the manner in which many users perform their daily work. It's anyone's guess how impactful these introductions will be, but it is clear that they are not simply another competitor to Windows and Macintosh desktop-oriented systems. They represent the move into a cloud-based future where applications run and data is stored remotely. Their success is likely to be contingent upon how fast consumers and businesses become comfortable with the idea of outsourcing these functions. In time, the transition is probably inevitable because the economies of scale and efficiency that cloud computing offers is too compelling to ignore.

FIGURE 3.9

The Chrome OS operating system's application launcher from an early demo of the product found at `http://www.youtube.com/watch?v=ANMrzw7JFzA&feature=channel`

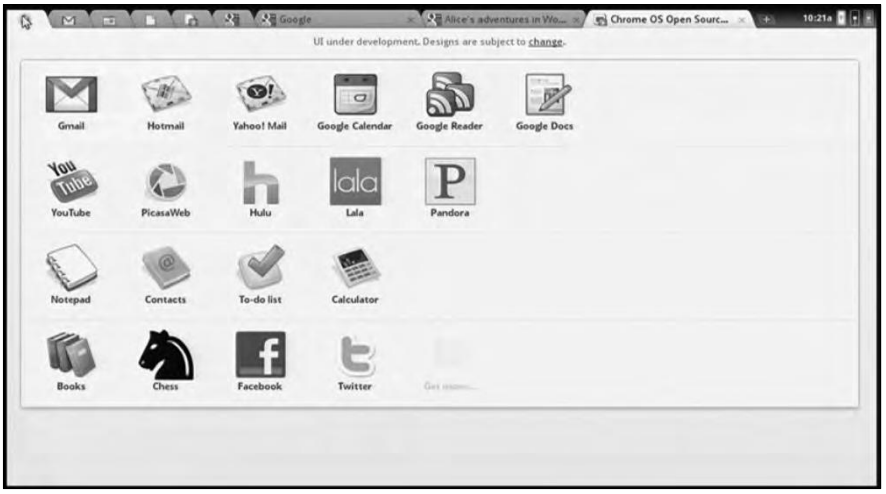
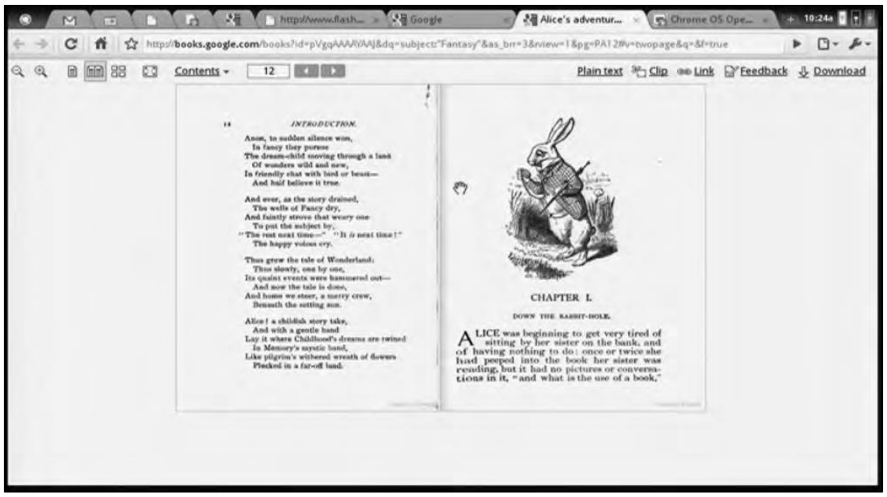


FIGURE 3.10

From the same source as Figure 3.9, this figure illustrates how the Google Reader appears within the Chrome OS as a tab.



Summary

In this chapter, you learned about the various architectural layers that make up the cloud computing stack. Cloud computing may be seen to be an extension of older Internet standards, but it includes some new architecture. In particular, infrastructure components were described, as were platform components. Platform services are software-based and can include a class of applications referred to as virtual appliances.

The focus of the chapter was how all the different parts of cloud computing work together. Many of the important standards are aimed at managing transactions in the cloud and making components work together. This chapter took an initial look at SOAP, XML, and the various WSDL services.

The chapter ended by considering the development of a new class of cloud-connected clients. These clients, which are often hardware and software solutions like the Google Chrome OS, may have major impact in changing the model by which people connect to the Internet and get their work done. They have many advantages, but we will have to wait to see how readily consumers and businesses are willing to adopt this new computing paradigm.

In the next chapter, each of the different Internet service models is more closely described.

CHAPTER

4

Understanding Services and Applications by Type

This chapter describes some of the different types of cloud computing models, categorized as a set of service models. You may think of cloud computing applications as being composed of a set of layers upon which distributed applications may be built or hosted. These layers include Infrastructure, Platform, and Software. Depending on the type and level of service being offered, a client can build on these layers to create cloud-based applications.

The service models described here—Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS)—are useful in categorizing not only cloud computing capabilities, but specific vendor offerings, products, and services. Infrastructure as a Service allows for the creation of virtual computing systems or networks.

Software as a Service represents a hosted application that is universally available over the Internet, usually through a browser. With Software as a Service, the user interacts directly with the hosted software. SaaS may be seen to be an alternative model to that of shrink-wrapped software and may replace much of the boxed software that we buy today.

Platform as a Service is a cloud computing infrastructure that creates a development environment upon which applications may be build. PaaS provides a model that can be used to create or augment complex applications such as Customer Relation Management (CRM) or Enterprise Resource Planning (ERP) systems. PaaS offers the benefits of cloud computing and is often componentized and based on a service-oriented architecture model.

As cloud computing matures, several service types are being introduced and overlaid upon these architectures. The most fully developed of these service types is Identity as a Service (IDaaS). Identity as a Service provides authentication and authorization services on distributed networks. Infrastructure and

IN THIS CHAPTER

- Learning about different service types
- Creating clouds with Infrastructure as a Service
- Working with Software as a Service
- Developing applications on a Platform as a Service
- Securing cloud transactions with Identity as a Service

supporting protocols for IDaaS are described in this chapter. Other service types such as Compliance as a Service (CaaS), provisioning, monitoring, communications, and many vertical services yet to be fully developed are touched upon in this chapter.

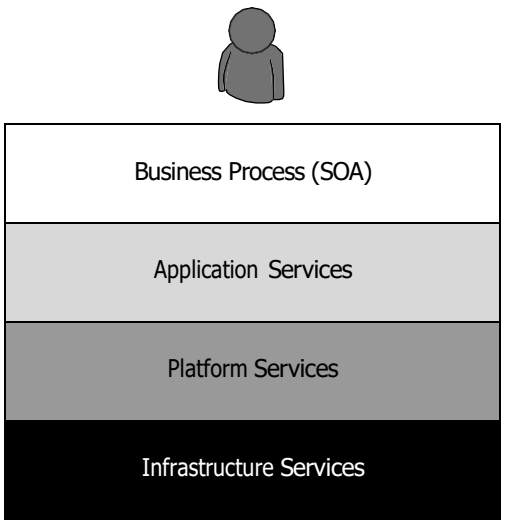
Defining Infrastructure as a Service (IaaS)

You can broadly partition cloud computing into four layers that form a cloud computing ecosystem, as shown in Figure 4.1. The Application layer forms the basis for Software as a Service (SaaS), while the Platform layer forms the basis for Platform as a Service (PaaS) models that are described in the next two sections. Infrastructure as a Service (IaaS) creates what may be determined to be a utility computing model, something that you can tap into and draw from as you need it without significant limits on the scalability of your deployment. You pay only for what you need when you need it. IaaS may be seen to be an incredibly disruptive technology, one that can help turn a small business into a large business nearly overnight. This is a most exciting prospect; one that is fueling a number of IaaS startups during one of the most difficult recessions of recent memory.

Infrastructure as a Service (IaaS) is a cloud computing service model in which hardware is virtualized in the cloud. In this particular model, the service vendor owns the equipment: servers, storage, network infrastructure, and so forth. The developer creates virtual hardware on which to develop applications and services. Essentially, an IaaS vendor has created a hardware utility service where the user provisions virtual resources as required.

FIGURE 4.1

The cloud computing ecosystem



The developer interacts with the IaaS model to create virtual private servers, virtual private storage, virtual private networks, and so on, and then populates these virtual systems with the applications and services it needs to complete its solution. In IaaS, the virtualized resources are mapped to real systems. When the client interacts with an IaaS service and requests resources from the virtual systems, those requests are redirected to the real servers that do the actual work.

IaaS workloads

The fundamental unit of virtualized client in an IaaS deployment is called a *workload.* A workload simulates the ability of a certain type of real or physical server to do an amount of work. The work done can be measured by the number of Transactions Per Minute (TPM) or a similar metric against a certain type of system. In addition to throughput, a workload has certain other attributes such as Disk I/Os measured in Input/Output Per Second IOPS, the amount of RAM consumed under load in MB, network throughput and latency, and so forth. In a hosted application environment, a client's application runs on a dedicated server inside a server rack or perhaps as a standalone server in a room full of servers. In cloud computing, a provisioned server called an instance is reserved by a customer, and the necessary amount of computing resources needed to achieve that type of physical server is allocated to the client's needs.

Figure 4.2 shows how three virtual private server instances are partitioned in an IaaS stack. The three workloads require three different sizes of computers: small, medium, and large.

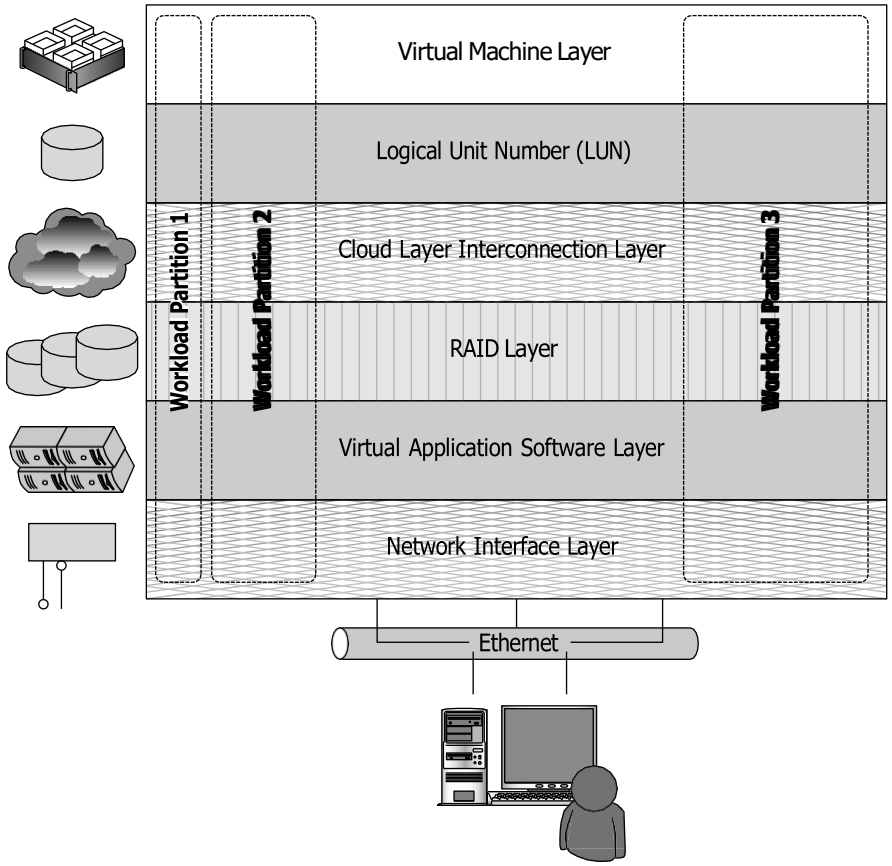
A client would reserve a machine equivalent required to run each of these workloads. The IaaS infrastructure runs these server instances in the data center that the service offers, drawing from a pool of virtualized machines, RAID storage, and network interface capacity. These three layers are expressions of physical systems that are partitioned as logical units. LUNs, the cloud interconnect layer, and the virtual application software layer are logical constructs. LUNs are logical storage containers, the cloud interconnect layer is a virtual network layer that is assigned IP addresses from the IaaS network pool, and the virtual application software layer contains software that runs on the physical VM instance(s) that have been partitioned from physical assets on the IaaS' private cloud.

From an architectural standpoint, the client in an IaaS infrastructure is assigned its own private network. The Amazon Elastic Computer Cloud (EC2), described in detail in Chapter 8, behaves as if each server is its own separate network—unless you create your own Virtual Private Cloud (an EC2 add-on feature), which provides a workaround to this problem. When you scale your EC2 deployment, you are adding additional networks to your infrastructure, which makes it easy to logically scale an EC2 deployment, but imposes additional network overhead because traffic must be routed between logical networks. Amazon Web Service's routing limits broadcast and multicast traffic because Layer-2 (Data Link) networking is not supported. Rackspace Cloud (`http://www.rackspacecloud.com/`) follows the AWS IP assignment model.

Other IaaS infrastructures such as the one Cloudscaling.com (`http://www.cloudscaling.com`) offers or a traditional VMWare cloud-assigned networks on a per-user basis, which allows for Level 2 networking options. The most prominent Level 2 protocols that you might use are tunneling options, because they enable VLANs.

FIGURE 4.2

A virtual private server partition in an IaaS cloud



Consider a transactional eCommerce system, for which a typical stack contains the following components:

- Web server
- Application server
- File server
- Database
- Transaction engine

This eCommerce system has several different workloads that are operating: queries against the database, processing of business logic, and serving up clients' Web pages.

The classic example of an IaaS service model is Amazon.com's Amazon Web Services (AWS). AWS has several data centers in which servers run on top of a virtualization platform (Xen) and may be partitioned into logical compute units of various sizes. Developers can then apply system images containing different operating systems and applications or create their own system images. Storage may be partitions, databases may be created, and a range of services such a messaging and notification can be called upon to make distributed application work correctly.

Cross-Ref

Amazon Web Services offers a classic Service Oriented Architecture (SOA) approach to IaaS. You learn more about AWS in Chapter 9; a description of the Service Oriented Architecture approach to building distributed applications is described in Chapter 13. ■

Pods, aggregation, and silos

Workloads support a certain number of users, at which point you exceed the load that the instance sizing allows. When you reach the limit of the largest virtual machine instance possible, you must make a copy or clone of the instance to support additional users. A group of users within a particular instance is called a *pod.* Pods are managed by a Cloud Control System (CCS). In AWS, the CCS is the AWS Management Console.

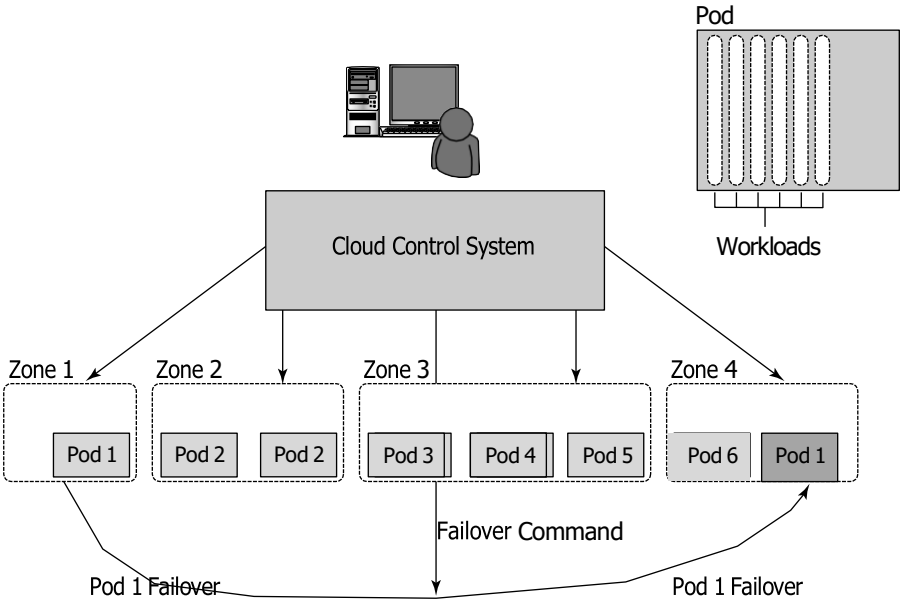
Sizing limitations for pods need to be accounted for if you are building a large cloud-based application. Pods are aggregated into pools within an IaaS region or site called an *availability zone.* In very large cloud computing networks, when systems fail, they fail on a pod-by-pod basis, and often on a zone-by-zone basis. For AWS' IaaS infrastructure, the availability zones are organized around the company's data centers in Northern California, Northern Virginia, Ireland, and Singapore. A failover system between zones gives IaaS private clouds a very high degree of availability. Figure 4.3 shows how pods are aggregated and virtualized in IaaS across zones.

When a cloud computing infrastructure isolates user clouds from each other so the management system is incapable of interoperating with other private clouds, it creates an information silo, or simply a silo. Most often, the term *silo* is applied to PaaS offerings such as Force.com or QuickBase, but silos often are an expression of the manner in which a cloud computing infrastructure is architected. Silos are the cloud computing equivalent of compute islands: They are processing domains that are sealed off from the outside.

When you create a private virtual network within an IaaS framework, the chances are high that you are creating a silo. Silos impose restrictions on interoperability that runs counter to the open nature of build-componentized service-oriented applications. However, that is not always a bad thing. A silo can be its own ecosystem; it can be protected and secured in ways that an open system can't be. Silos just aren't as flexible as open systems and are subject to vendor lock-in.

FIGURE 4.3

Pods, aggregation, and failover in IaaS



Defining Platform as a Service (PaaS)

The Platform as a Service model describes a software environment in which a developer can create customized solutions within the context of the development tools that the platform provides. Platforms can be based on specific types of development languages, application frameworks, or other constructs. A PaaS offering provides the tools and development environment to deploy applications on another vendor's application. Often a PaaS tool is a fully integrated development environment; that is, all the tools and services are part of the PaaS service. To be useful as a cloud computing offering, PaaS systems must offer a way to create user interfaces, and thus support standards such as HTLM, JavaScript, or other rich media technologies.

In a PaaS model, customers may interact with the software to enter and retrieve data, perform actions, get results, and to the degree that the vendor allows it, customize the platform involved. The customer takes no responsibility for maintaining the hardware, the software, or the development of the applications and is responsible only for his interaction with the platform. The vendor is responsible for all the operational aspects of the service, for maintenance, and for managing the product(s) lifecycle.

The one example that is most quoted as a PaaS offering is Google's App Engine platform, which is described in more detail in Chapter 8. Developers program against the App Engine using Google's published APIs. The tools for working within the development framework, as well as the structure of the file system and data stores, are defined by Google. Another example of a PaaS offering is