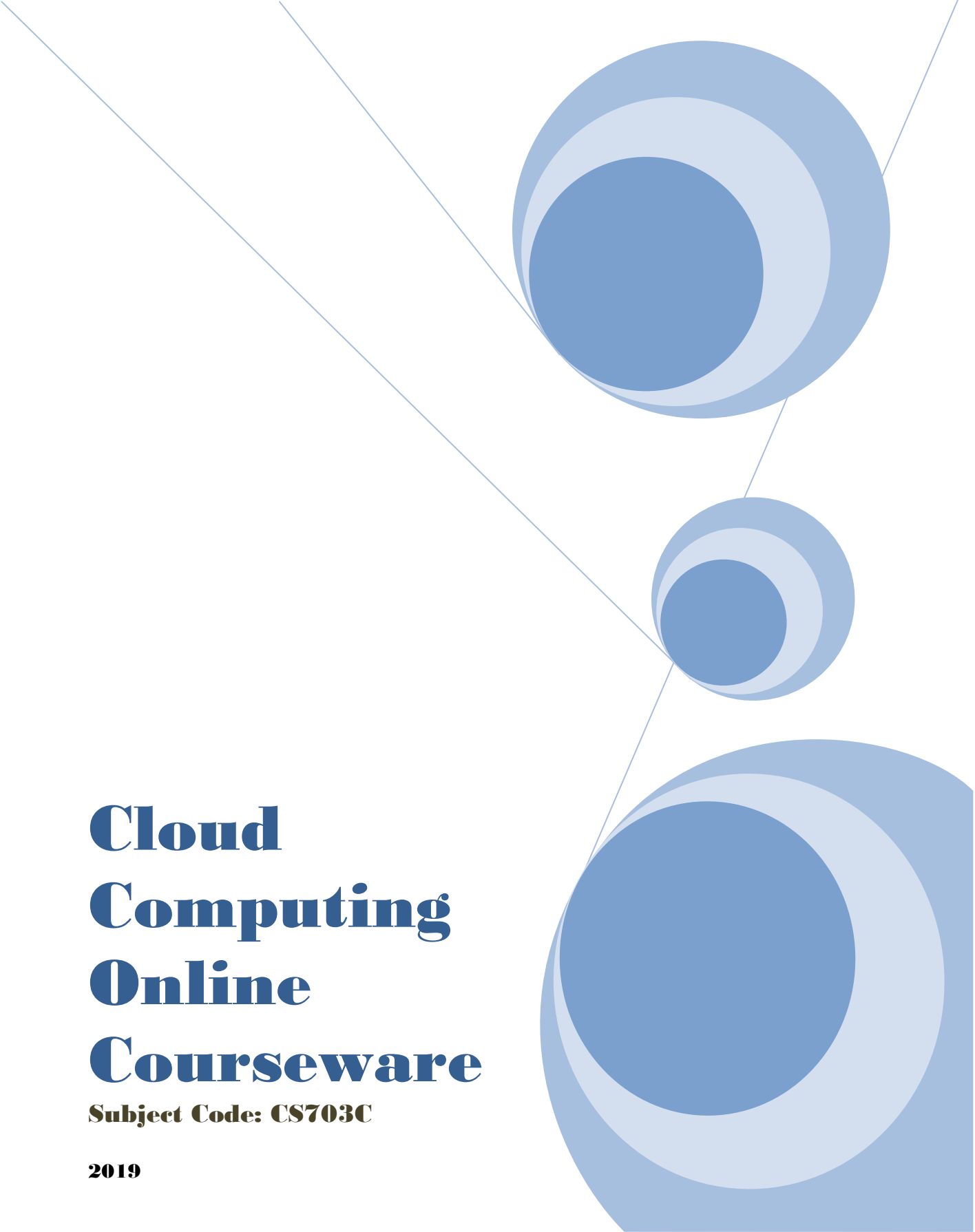


Cloud Computing Online Courseware

Subject Code: CS703C

2019

Proposed Syllabus for B.Tech Computer Science and Engineering Programme (Autonomy)
4th Year, 7th Semester
Paper Name: Cloud Computing
Code: CS703C Contacts:
3L Credits: 3
Allotted hours: 35L

COURSE OBJECTIVES

- The student will also learn how to apply trust-based security model to real-world security problems.
- An overview of the concepts, processes, and best practices needed to successfully secure information within Cloud infrastructures.
- Students will learn the basic Cloud types and delivery models and develop an understanding of the risk and compliance responsibilities and Challenges for each Cloud type and service delivery model.

Module 1: Definition of Cloud Computing and its Basics

1. Definition of Cloud Computing: Defining a Cloud, Cloud Types – NIST model, Cloud Cube model, Deployment models (Public , Private, Hybrid and Community Clouds), Service models – Infrastructure as a Service, Platform as a Service, Software as a Service with examples of services/ service providers, Cloud Reference model, Characteristics of Cloud Computing – a shift in paradigm Benefits and advantages of Cloud Computing
2. Cloud Architecture: Cloud Infrastructure, Architecture of each component, Virtualization versus Traditional Approach, Virtualization Model for Cloud Computing.
3. Services and Applications by Type
IaaS – Basic concept, Workload, partitioning of virtual private server instances, Pods, aggregations, silos
PaaS – Basic concept, tools and development environment with examples
SaaS - Basic concept and characteristics, Open SaaS and SOA, examples of SaaS platform
Identity as a Service (IDaaS) Compliance as a Service (CaaS)

Module 2: Use of Platforms in Cloud Computing

1. Concepts of Abstraction and Virtualization
Virtualization technologies: Types of virtualization, Load Balancing and Virtualization: Basic Concepts, Network resources for load balancing; Classification of Virtualization Environment: Scheduling-based Environment, Load-Distribution-Based Environment, Energy Aware-Based Environment, Operational-Based Environment, Distributed Pattern-Based Environment, Transactional-Based Environment
Mention of The Google Cloud as an example of use of load balancing Hypervisors: Virtual machine technology and types, VMware vSphere Machine imaging (including mention of Open Virtualization Format – OVF)
Porting of applications in the Cloud: The simple Cloud API and AppZero Virtual Application appliance
2. Concepts of Platform as a Service
Definition of services, Distinction between SaaS and PaaS (knowledge of Salesforce.com and Force.com), Application development.Use of PaaS Application frameworks
3. Use of Google Web Services
Discussion of Google Applications Portfolio – Indexed search, Dark Web, Aggregation and disintermediation, Productivity applications and service, Adwords, Google Analytics, Google Translate, a brief discussion on Google Toolkit (including introduction of Google APIs in brief), major features of Google App Engine service.
4. Use of Amazon Web Services
Amazon Web Service components and services: Amazon Elastic Cloud, Amazon Simple Storage system, Amazon Elastic Block Store, Amazon SimpleDB and Relational Database Service
5. Use of Microsoft Cloud Services
Windows Azure platform: Microsoft's approach, architecture, and main elements, overview of Windows Azure AppFabric, Content Delivery Network, SQL Azure, and Windows Live services

Module 3 : Cloud Infrastructure

Types of services required in implementation – Consulting, Configuration, Customization and Support
1. Cloud Management

An overview of the features of network management systems and a brief introduction of related products from large cloud vendors, Monitoring of an entire cloud computing deployment stack – an overview with mention of some products, Lifecycle management of cloud services (six stages of lifecycle)

2. Live Migration of Virtual Machines:

Need of Live Migration of Virtual Machine, A Designing Process of Live Migration, and Security Issues during live migration

3. Concepts of Cloud Security

Infrastructure Security, Infrastructure Security: The Network Level, The Host Level, The Application Level, Data Security and Storage, Aspects of Data Security, Data Security Mitigation Provider Data and Its Security, Identity and Access Management

4. Auditing and Compliance in Cloud Environment:

Data Security in Cloud Computing Environment, Need for Auditing in Cloud Computing Environment, Third Party Service Provider, Cloud Auditing Outsourcing Lifecycle Phases, Auditing Classification.

Module 4 : Concepts of Services and Applications

1. Service Oriented Architecture: Basic concepts of message-based transactions, Protocol stack for an SOA architecture, Event-driven SOA, Enterprise Service Bus, Service catalogs
2. Applications in the Cloud: Concepts of cloud transactions, functionality mapping, Application attributes, Cloud service attributes, System abstraction and Cloud Bursting, Applications and Cloud APIs
3. Cloud-based Storage: Cloud storage definition – Manned and Unmanned
4. Webmail Services: Cloud mail services including Google Gmail, Mail2Web, Windows Live Hotmail, Yahoo mail, concepts of Syndication services

Books Recommended:

1. ***Cloud Computing Bible*** by Barrie Sosinsky, Wiley India Pvt. Ltd, 2013
2. ***Mastering Cloud Computing*** by Rajkumar Buyya, Christian Vecchiola, S. Thamarai Selvi, McGraw Hill Education (India) Private Limited, 2013
3. ***Fundamentals of Cloud Computing*** by Prasant Kumar Pattnaik, Souvik Pal, Manas Ranjan Kabat, Vikas Publications
4. ***Cloud computing: A practical approach***, Anthony T. Velte, Tata Mcgraw-Hill
5. ***Cloud Computing***, Miller, Pearson
6. ***Building applications in cloud:Concept, Patterns and Projects***, Moyer, Pearson

References:

1. Cloud Computing – Second Edition by Dr. Kumar Saurabh, Wiley India

COURSE OUTCOMES

After completion of course, students would be able to:

CS703C .1. Identify security aspects of each cloud model
CS703C .2. Develop a risk-management strategy for moving to the Cloud
CS703C .3. Implement a public cloud instance using a public cloud service provider
CS703C .4. Apply trust-based security model to different layer

CO-PO Mapping

CO	PO1	PO2	POP3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
CS703C.1				√								
CS703C.2			√				√				√	
CS703C.3						√			√			
CS703C.4								√	√			

Module – I

Definition of Cloud Computing and its Basics

Defining Cloud Computing

Cloud computing refers to applications and services that run on a dis-tributed network using virtualized resources and accessed by com-mon Internet protocols and networking standards. It is distinguished by the notion that resources are virtual and limitless and that details of the physical systems on which software runs are abstracted from the user.

In an effort to better describe cloud computing, a number of cloud types have been defined. In this chapter, you learn about two different classes of clouds: those based on the deployment model and those based on the service model. The deployment model tells you where the cloud is located and for what purpose. Public, private, community, and hybrid clouds are deploy-ment models.

Service models describe the type of service that the service provider is offer-ing. The best-known service models are Software as a Service, Platform as a Service, and Infrastructure as a Service—the SPI model. The service models build on one another and define what a vendor must manage and what the client's responsibility is.

Cloud computing represents a real paradigm shift in the way in which sys-tems are deployed. The massive scale of cloud computing systems was enabled by the popularization of the Internet and the growth of some large service companies. Cloud computing makes the long-held dream of utility computing possible with a pay-as-you-go, infinitely scalable, universally available system. With cloud computing, you can start very small and become big very fast. That's why cloud computing is revolutionary, even if the technology it is built on is evolutionary.

Not all applications benefit from deployment in the cloud. Issues with latency, transaction control, and in particular security and regulatory compli-ance are of particular concern.

Defining Cloud Computing

Cloud computing takes the technology, services, and applications that are similar to those on the Internet and turns them into a self-service utility. The use of the word "cloud" makes reference to the two essential concepts:

- **Abstraction:** Cloud computing abstracts the details of system implementation from users and developers. Applications run on physical systems that aren't specified, data is stored in locations that are unknown, administration of systems is outsourced to others, and access by users is ubiquitous.
- **Virtualization:** Cloud computing virtualizes systems by pooling and sharing resources. Systems and storage can be provisioned as needed from a centralized infrastructure, costs are assessed on a metered basis, multi-tenancy is enabled, and resources are scalable with agility.

Computing as a utility is a dream that dates from the beginning of the computing industry itself. A set of new technologies has come along that, along with the need for more efficient and affordable computing, has enabled an on-demand system to develop. It is these enabling technologies that are the focal point of this book.

Many people mistakenly believe that cloud computing is nothing more than the Internet given a different name. Many drawings of Internet-based systems and services depict the Internet as a cloud, and people refer to applications running on the Internet as "running in the cloud," so the confusion is understandable. The Internet has many of the characteristics of what is now being called cloud computing. The Internet offers abstraction, runs using the same set of protocols and standards, and uses the same applications and operating systems. These same characteristics are found in an intranet, an internal version of the Internet. When an intranet becomes large enough that a diagram no longer wishes to differentiate between individual physical systems, the intranet too becomes identified as a cloud.

Cloud computing is an abstraction based on the notion of pooling physical resources and present-ing them as a virtual resource. It is a new model for provisioning resources, for staging applica-tions, and for platform-independent user access to services. Clouds can come in many different types, and the services and applications that run on clouds may or may not be delivered by a cloud service provider. These different types and levels of cloud services mean that it is important to define what type of cloud computing system you are working with.

To help clarify how cloud computing has changed the nature of commercial system deployment, consider these three examples:

- **Google:** In the last decade, Google has built a worldwide network of datacenters to service its search engine. In doing so Google has captured a substantial portion of the world's advertising revenue. That revenue has enabled Google to offer free software to users based on that infrastructure and has changed the market for user-facing software. This is the classic Software as a Service case described in Chapter 8.

- **Azure Platform:** By contrast, Microsoft is creating the Azure Platform. It enables .NET Framework applications to run over the Internet as an alternate platform for Microsoft developer software running on desktops, which you will learn about in Chapter 10.
- **Amazon Web Services:** One of the most successful cloud-based businesses is Amazon Web Services, which is an Infrastructure as a Service offering that lets you rent virtual computers on Amazon's own infrastructure. AWS is the subject of Chapter 9.

These new capabilities enable applications to be written and deployed with minimal expense and to be rapidly scaled and made available worldwide as business conditions permit. This is truly a revolutionary change in the way enterprise computing is created and deployed.

Cloud Types

To discuss cloud computing intelligently, you need to define the lexicon of cloud computing; many acronyms in this area probably won't survive long. Most people separate cloud computing into two distinct sets of models:

- **Deployment models:** This refers to the location and management of the cloud's infrastructure.
- **Service models:** This consists of the particular types of services that you can access on a cloud computing platform.

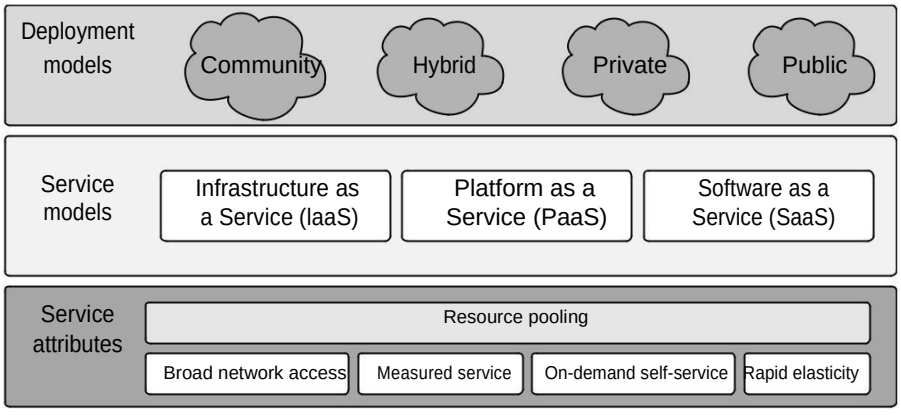
This is a very useful demarcation that is now widely accepted.

The NIST model

The United States government is a major consumer of computer services and, therefore, one of the major users of cloud computing networks. The U.S. National Institute of Standards and Technology (NIST) has a set of working definitions (http://csrc.nist.gov/groups/SNS/cloud-computing/cloud-def-v15.doc) that separate cloud computing into service models and deployment models. Those models and their relationship to essential characteristics of cloud com-puting are shown in Figure 1.1.

The NIST model originally did not require a cloud to use virtualization to pool resources, nor did it absolutely require that a cloud support multi-tenancy in the earliest definitions of cloud computing. Multi-tenancy is the sharing of resources among two or more clients. The latest version of the NIST definition does require that cloud computing networks use virtualization and support multi-tenancy.

The NIST cloud computing definitions



Because cloud computing is moving toward a set of modular interacting components based on standards such as the Service Oriented Architecture (described in Chapter 13), you might expect that future versions of the NIST model may add those features as well. The NIST cloud model doesn't address a number of intermediary services such as transaction or service brokers, provisioning, integration, and interoperability services that form the basis for many cloud computing discussions. Given the emerging roles of service buses, brokers, and cloud APIs at various levels, undoubtedly these elements need to be added to capture the whole story.

The Cloud Cube Model

The Open Group maintains an association called the Jericho Forum (https://www.opengroup.org/jericho/index.htm) whose main focus is how to protect cloud networks. The group has an interesting model that attempts to categorize a cloud network based on four dimen-sional factors. As described in its paper called "Cloud Cube Model: Selecting Cloud Formations for Secure Collaboration" (http://www.opengroup.org/jericho/cloud_cube_model_v1.0.pdf), the type of cloud networks you use dramatically changes the notion of where the boundary between the client's network and the cloud begins and ends.

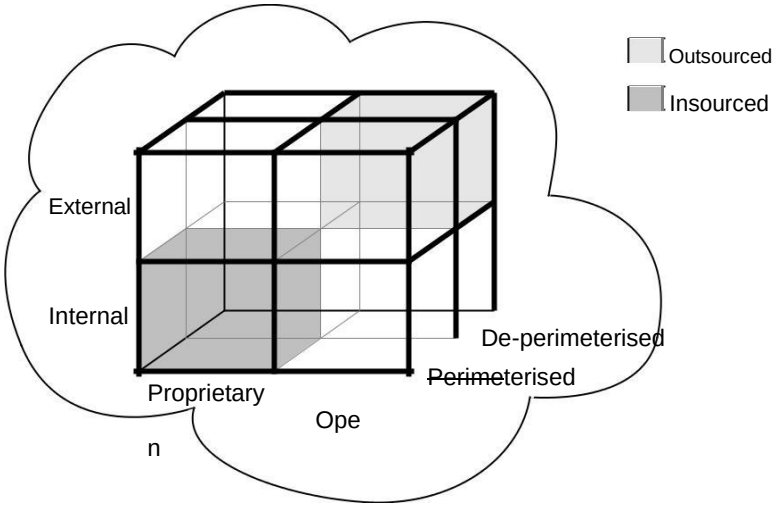
The four dimensions of the Cloud Cube Model are shown in Figure 1.2 and listed here:

- **Physical location of the data:** Internal (I) / External (E) determines your organization's boundaries.
- **Ownership:** Proprietary (P) / Open (O) is a measure of not only the technology ownership, but of interoperability, ease of data transfer, and degree of vendor application lock-in.

- **Security boundary:** Perimeterised (Per) / De-perimiterised (D-p) is a measure of whether the operation is inside or outside the security boundary or network firewall.
- **Sourcing:** Insourced or Outsourced means whether the service is provided by the customer or the service provider.

FIGURE 1.2

The Jericho Forum's Cloud Cube Model



Taken together, the fourth dimension corresponds to two different states in the eight possible cloud forms: Per (IP, IO, EP, EO) and D-p (IP, IO, EP, EO). The sourcing dimension addresses the deliverer of the service. What the Cloud Cube Model is meant to show is that the traditional notion of a network boundary being the network's firewall no longer applies in cloud computing.

Deployment models

A deployment model defines the purpose of the cloud and the nature of how the cloud is located.

The NIST definition for the four deployment models is as follows:

- **Public cloud:** The public cloud infrastructure is available for public use alternatively for a large industry group and is owned by an organization selling cloud services.
- **Private cloud:** The private cloud infrastructure is operated for the exclusive use of an organization. The cloud may be managed by that organization or a third party. Private clouds may be either on- or off-premises.
- **Hybrid cloud:** A hybrid cloud combines multiple clouds (private, community of public) where those clouds retain their unique identities, but are bound together as a unit. A

hybrid cloud may offer standardized or proprietary access to data and applications, as well as application portability.

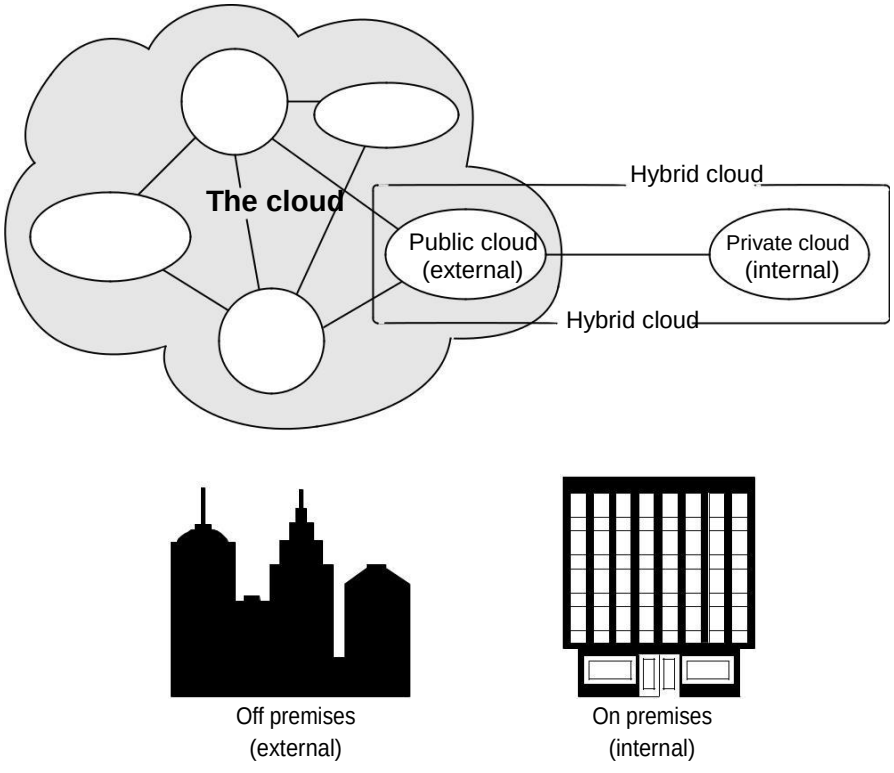
- **Community cloud:** A community cloud is one where the cloud has been organized to serve a common function or purpose.

  It may be for one organization or for several organizations, but they share common concerns such as their mission, policies, security, regulatory compliance needs, and so on. A community cloud may be managed by the constituent organization(s) or by a third party.

Figure 1.3 shows the different locations that clouds can come in. In the sections that follow, these different cloud deployment models are described in more detail.

FIGURE 1.3

Deployment locations for different cloud types



The United States Government, under the auspices of the General Services Administrator (GSA), launched a cloud computing portal called Apps.gov, as shown in Figure 1.4, with the purpose of providing cloud services to federal agencies. Described under the "U.S. Federal Cloud Computing

Initiative (http://www.scribd.com/doc/17914883/US-Federal-Cloud-Computing-Initiative-RFQ-GSA), the goal of the initiative is to make large portions of the federal govern-ment's apparatus available under a cloud computing model. This is a good example of a community cloud deployment, with the government being the community.

FIGURE 1.4

Apps.gov is the U.S. government's cloud computing system for its various agencies.

Apps.gov is also making available connections to free media services from its cloud, such as Twitter and YouTube. An example of this connection in practice is the YouTube channel created by the White House for citizens' outreach. You can find the White House channel at http://www.youtube.com/whitehouse and the general U.S. Government YouTube channel at http://www.youtube.com/usgovernment. You can see YouTube in action when you visit WhiteHouse.gov and click the video link that usually appears on that home page.

Service models

In the deployment model, different cloud types are an expression of the manner in which infrastructure is deployed. You can think of the cloud as the boundary between where a client's network, management, and responsibilities ends and the cloud service provider's begins. As cloud

computing has developed, different vendors offer clouds that have different services associated with them. The portfolio of services offered adds another set of definitions called the service model.

There are many different service models described in the literature, all of which take the following form:

XaaS, or "<*Something*> as a Service"

Three service types have been universally accepted:

- **Infrastructure as a Service:** IaaS provides virtual machines, virtual storage, virtual infra-structure, and other hardware assets as resources that clients can provision.

  The IaaS service provider manages all the infrastructure, while the client is responsible for all other aspects of the deployment. This can include the operating system, applications, and user interactions with the system.

- **Platform as a Service:** PaaS provides virtual machines, operating systems, applications, services, development frameworks, transactions, and control structures.

  The client can deploy its applications on the cloud infrastructure or use applications that were programmed using languages and tools that are supported by the PaaS service provider. The service provider manages the cloud infrastructure, the operating systems, and the enabling software. The client is responsible for installing and managing the application that it is deploying.

- **Software as a Service:** SaaS is a complete operating environment with applications, man-agement, and the user interface.

  In the SaaS model, the application is provided to the client through a thin client interface (a browser, usually), and the customer's responsibility begins and ends with entering and managing its data and user interaction. Everything from the application down to the infrastructure is the vendor's responsibility.

The three different service models taken together have come to be known as the SPI model of cloud computing. Many other service models have been mentioned: StaaS, Storage as a Service; IdaaS, Identity as a Service; CmaaS, Compliance as a Service; and so forth. However, the SPI ser-vices encompass all the other possibilities.

It is useful to think of cloud computing's service models in terms of a hardware/software stack. One such representation called the Cloud Reference Model is shown in Figure 1.5. At the bottom of the stack is the hardware or infrastructure that comprises the network. As you move upward in the stack, each service model inherits the capabilities of the service model beneath it. IaaS has the least levels of integrated functionality and the lowest levels of integration, and SaaS has the most.

Examples of IaaS service providers include:

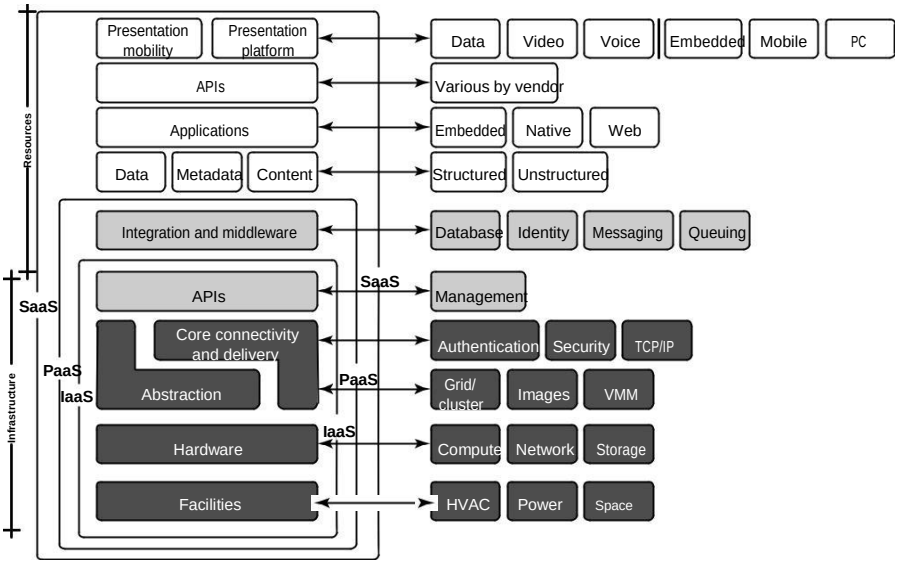
- Amazon Elastic Compute Cloud (EC2)
- Eucalyptus
- GoGrid

- FlexiScale
- Linode
- RackSpace Cloud
- Terremark

All these vendors offer direct access to hardware resources. On Amazon EC2, considered the classic IaaS example, a client would provision a computer in the form of a virtual machine image, pro-vision storage, and then go on to install the operating system and applications onto that virtual system. Amazon has a number of operating systems and some enterprise applications that they offer on a rental basis to customers in the form of a number of canned images, but customers are free to install whatever software they want to run. Amazon's responsibilities as expressed in its Service Level Agreement, which is published on Amazon's Web site, contractually obligates Amazon to provide a level of performance commensurate with the type of resource chosen, as well as a cer-tain level of reliability as measured by the system's uptime.

FIGURE 1.5

The Cloud Reference Model



A PaaS service adds integration features, middleware, and other orchestration and choreography services to the IaaS model. Examples of PaaS services are:

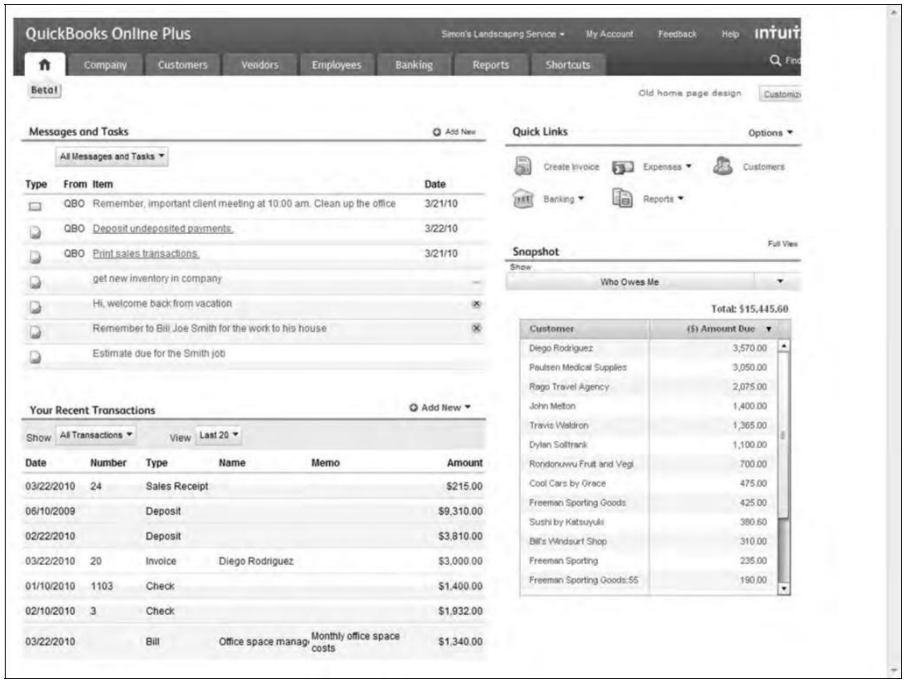
- Force.com
- GoGrid CloudCenter

- Google AppEngine
- Windows Azure Platform

When a cloud computing vendor offers software running in the cloud with use of the application on a pay-as-you-go model, it is referred to as SaaS. With SaaS, the customer uses the application as needed and is not responsible for the installation of the application, its maintenance, or its upkeep. A good example of an SaaS offering is an online accounting package, with the online versions of Quicken and Quickbooks a prime example. Figure 1.6 shows a home page for QuickBooks Online plus on the Intuit.com Web site.

FIGURE 1.6

A home page for a Quickbooks customer on the Intuit.com Web site is an example of an SaaS service.



A client using an SaaS service might—as is the case for Quickbooks online—log into the service from his browser, create an account, and enter data into the system. Intuit.com has a service agreement that not only covers the performance of the hardware and software, but extends to protecting the data that they store for clients, and other fundamental characteristics.

Other good examples of SaaS cloud service providers are:

- GoogleApps
- Oracle On Demand
- SalesForce.com
- SQL Azure

These service model classifications start to get confusing rather quickly when you have a cloud service provider that starts out offering services in one area and then develops services that are classified as another type. For example, SalesForce.com started out as a Customer Relationship Management SaaS platform that allowed clients to add their own applications. Over time SalesForce.com opened an API called the Force API that allowed developers to create applications based on the SalesForce.com technologies. Force.com is thus their PaaS service.

As another example, take the PaaS offering that is the Windows Azure Platform. Windows Azure Platform allows .NET developers to stage their applications on top of Microsoft's infrastructure so that any application built with the .NET Framework can live locally, in Microsoft's cloud network, or some combination thereof. As Microsoft adds enterprise applications to its cloud service portfo-lio, as it has in the case of SQL Azure (and many other enterprise applications to come), these offerings fall under the rubric of being an SaaS service model.

Because a discussion of service models forms the basis for Chapter 4, I refer you to that chapter for a more in-depth discussion of this topic.

Examining the Characteristics of Cloud Computing

Cloud computing builds on so many older concepts in computer technology that it can be hard for people newly introduced to the concept to grasp that it represents a paradigm shift in computing. It's an evolutionary change that enables a revolutionary new approach to how computing services are produced and consumed.

Paradigm shift

When you choose a cloud service provider, you are renting or leasing part of an enormous infra-structure of datacenters, computers, storage, and networking capacity. Many of these datacenters are multi-million-dollar investments by the companies that run them. To give you some sense of scale, it has been estimated that a state-of-the-art microchip fabrication facility can cost anywhere from $2 to $5 billion. By comparison, a state of the art cloud computing datacenter can run in the range of $100 million. Most of the large cloud computing service providers have multiple data-centers located all over the world. An accurate count can be difficult to obtain, but in Chapter 9

the location of some 20 datacenters in Amazon Web Service's cloud are detailed. Google's cloud includes perhaps some 35 datacenters worldwide.

In the 1960s, military initiative aimed at miniaturizing electronics funded many of the semiconductor production lines that led to advanced microprocessors, dense memory arrays, and the sophisticated integrated circuit technology that makes computers, mobile devices, and so much more possible today. In the 1990s, the commercialization of the Internet gave rise to some very large companies that were forced to build very large computing infrastructures to support their businesses.

Amazon.com's infrastructure was built to support elastic demand so the system could accommo-date peak traffic on a busy shopping day such as "Black Monday." Because much of the capacity was idle, Amazon.com first opened its network to partners and then as Amazon Web Services to customers.

Google's business has also grown exponentially and required the building of datacenters worldwide. One of its datacenters in Dalles, Oregon, built in 2006 on the banks of the Columbia River, is shown in Figure 1.7. It is the size of an American football field.

FIGURE 1.7

The Google Dalles, Oregon, datacenter shown in Google Earth is an industrial-sized information technology utility.

As these various datacenters grew in size, businesses have developed their datacenters as "green-field" projects. Datacenters have been sited to do the following:

- Have access to low cost power

- Leverage renewable power source
- Be near abundant water
- Be sited where high-speed network backbone connections can be made
- Keep land costs modest and occupation unobtrusive
- Obtain tax breaks
- Optimize the overall system latency

These characteristics make cloud computing networks highly efficient and capture enough margin to make utility computing profitable.

It has been estimated that the Internet consumes roughly 10 percent of the world's total power, so these companies are very big energy consumers. In some cases, such as Google, these companies may also become some of the major energy producers of the 21st century. Essentially what has happened is that the Internet has funded the creation of the first information technology utilities. That's why cloud computing is such a big deal.

According to the research firm IDC, the following areas were the top five cloud applications in use in 2010:

- Collaboration applications
- Web applications/Web serving
- Cloud backup
- Business applications
- Personal productivity applications

The last five years have seen a proliferation of services and productivity applications delivered on-line as cloud computing applications. Examples of the impact of cloud computing abound in your everyday life, although many people do not make the connection to what was once a straightfor-ward client/server Internet deployment. Movement of these applications to the cloud has been transparent, and in many cases the older on-premises deployment is supported by the same appli-cations hosted in the cloud.

For example, many people have used ChannelAdvisor.com for their auction listings and sales man-agement. That site recently expanded its service to include a CRM connector to Salesforce.com. One of the largest call center operations companies is a cloud-based service, Liveops.com. Figure 1.8 shows the Liveops home page.

Cloud computing has shifted the economics of software delivery in a manner similar to the way that music downloads have shifted the delivery of commercial music. The cost advantages of cloud

computing have enabled new software vendors to create productivity applications that they can make available to people at a much smaller cost than would be possible for shrink-wrapped software. Given the general demise of the big-box computer store along with many other traditional retail models, it has become increasingly difficult for vendors to get shelf space. You can visit your local Wal-Mart to get some sense of this issue.

In Chapter 16, "Working with Productivity Software," some of these applications are described. This new model of computer application delivery has allowed vendors like Google to offer com-plete office suites to individuals for free, supported by its advertiser subscription model. Even Google's business offerings have had some major successes against industry leader Microsoft Office. Last year, Los Angeles County switched to Google Docs.

FIGURE 1.8

Liveops.com is a cloud computing call center service.

Benefits of cloud computing

"The NIST Definition of Cloud Computing" by Peter Mell and Tim Grance (version 14, 10/7/2009) described previously in this chapter (refer to Figure 1.1) that classified cloud computing into the three SPI service models (SaaS, IaaS, and PaaS) and four cloud types (public, private, community, and hybrid), also assigns five essential characteristics that cloud computing systems must offer:

- **On-demand self-service:** A client can provision computer resources without the need for interaction with cloud service provider personnel.
- **Broad network access:** Access to resources in the cloud is available over the network using standard methods in a manner that provides platform-independent access to clients of all types.

  This includes a mixture of heterogeneous operating systems, and thick and thin platforms such as laptops, mobile phones, and PDA.
- **Resource pooling:** A cloud service provider creates resources that are pooled together in a system that supports multi-tenant usage.

  Physical and virtual systems are dynamically allocated or reallocated as needed. Intrinsic in this concept of pooling is the idea of abstraction that hides the location of resources such as virtual machines, processing, memory, storage, and network bandwidth and connectivity.
- **Rapid elasticity:** Resources can be rapidly and elastically provisioned.

  The system can add resources by either scaling up systems (more powerful computers) or scaling out systems (more computers of the same kind), and scaling may be automatic or manual. From the standpoint of the client, cloud computing resources should look limit-less and can be purchased at any time and in any quantity.
- **Measured service:** The use of cloud system resources is measured, audited, and reported to the customer based on a metered system.

  A client can be charged based on a known metric such as amount of storage used, number of transactions, network I/O (Input/Output) or bandwidth, amount of processing power used, and so forth. A client is charged based on the level of services provided.

While these five core features of cloud computing are on almost anybody's list, you also should consider these additional advantages:

- **Lower costs:** Because cloud networks operate at higher efficiencies and with greater utili-zation, significant cost reductions are often encountered.
- **Ease of utilization:** Depending upon the type of service being offered, you may find that you do not require hardware or software licenses to implement your service.
- **Quality of Service:** The Quality of Service (QoS) is something that you can obtain under contract from your vendor.
- **Reliability:** The scale of cloud computing networks and their ability to provide load bal-ancing and failover makes them highly reliable, often much more reliable than what you can achieve in a single organization.
- **Outsourced IT management:** A cloud computing deployment lets someone else manage your computing infrastructure while you manage your business. In most instances, you achieve considerable reductions in IT staffing costs.

- **Simplified maintenance and upgrade:** Because the system is centralized, you can easily apply patches and upgrades. This means your users always have access to the latest soft-ware versions.

- **Low Barrier to Entry:** In particular, upfront capital expenditures are dramatically reduced. In cloud computing, anyone can be a giant at any time.

This very long list of benefits should make it obvious why so many people are excited about the idea of cloud computing. Cloud computing is not a panacea, however. In many instances, cloud computing doesn't work well for particular applications.

Disadvantages of cloud computing

While the benefits of cloud computing are myriad, the disadvantages are just as numerous. As a general rule, the advantages of cloud computing present a more compelling case for small organizations than for larger ones. Larger organizations can support IT staff and development efforts that put in place custom software solutions that are crafted with their particular needs in mind.

When you use an application or service in the cloud, you are using something that isn't necessarily as customizable as you might want. Additionally, although many cloud computing applications are very capable, applications deployed on-premises still have many more features than their cloud counterparts.

All cloud computing applications suffer from the inherent latency that is intrinsic in their WAN connectivity. While cloud computing applications excel at large-scale processing tasks, if your applica-tion needs large amounts of data transfer, cloud computing may not be the best model for you.

Additionally, cloud computing is a stateless system, as is the Internet in general. In order for com-munication to survive on a distributed system, it is necessarily unidirectional in nature. All the requests you use in HTTP: PUTs, GETs, and so on are requests to a service provider. The service provider then sends a response. Although it may seem that you are carrying on a conversation between client and provider, there is an architectural disconnect between the two. That lack of state allows messages to travel over different routes and for data to arrive out of sequence, and many other characteristics allow the communication to succeed even when the medium is faulty. Therefore, to impose transactional coherency upon the system, additional overhead in the form of service brokers, transaction managers, and other middleware must be added to the system. This can introduce a very large performance hit into some applications.

If you had to pick a single area of concern in cloud computing, that area would undoubtedly be privacy and security. When your data travels over and rests on systems that are no longer under your control, you have increased risk due to the interception and malfeasance of others. You can't count on a cloud provider maintaining your privacy in the face of government actions.

In the United States, an example is the National Security Agency's program that ran millions of phone calls from AT&T and Verizon through a data analyzer to extract the phone calls that matched its security criteria. VoIP is one of the services that is heavily deployed on cloud comput-ing systems. Another example is the case of Google's service in China, which had been subject to a

filter that removed content to which the Chinese government objected. After five years of operation, and after Google detected that Chinese hackers were accessing Gmail accounts of Chinese citizens, Google moved their servers for Google.ch to Hong Kong.

So while the cloud computing industry continues to address security concerns, if you have an application that works with sensitive data, you need to be particularly aware of the issues involved. Chapter 12, "Understanding Cloud Security," expands upon these points in more detail.

These days most organizations are faced with regulatory compliance issues of various kinds. In the United States, companies must comply with the accounting requirements of the Sarbanes-Oxley Act; health care providers comply with the data privacy rules of HIPAA, and so on. In Europe, the European Common Market has a raft of its own legislation for companies to deal with. Rules apply to data at rest, and different rules may apply to data in transit. If you stage your cloud computing deployment across states and countries, the bad news is that you may end up having to comply with multiple jurisdictions. Don't expect much support from the cloud system provider or from the governments involved. The laws of most regulatory agencies place the entire burden on the cli-ent. So when it comes to compliance, cloud computing is still the "Wild West" of computing.

Understanding Cloud Architecture

Cloud computing is a natural extension of many of the design princi-ples, 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.

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 pro-tocols. 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 sys-tem is still a very powerful one. A PaaS or SaaS service provider gets the same benefits from a com-posable 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.

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.

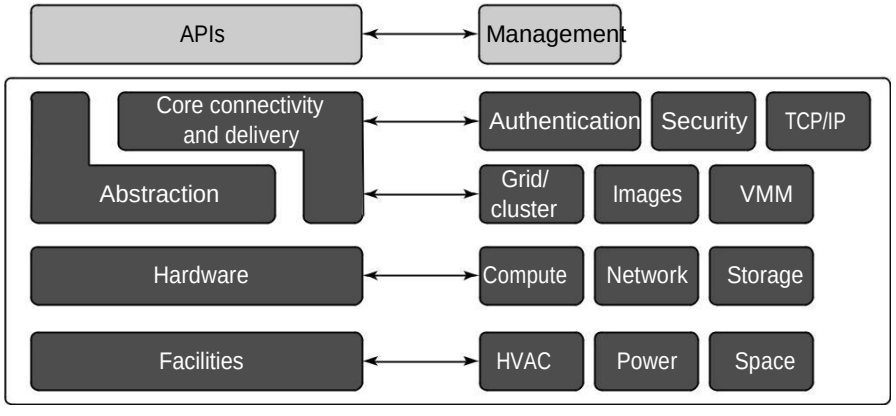
Figure 3.1 shows the portion of the cloud computing stack that is defined as the "server." In the diagram, the API is shown shaded in gray because it is an optional component that isn't always delivered with the server. The VMM component is the Virtual Machine Monitor, also called a hypervisor. This 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, miss-ing 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 inter-ested 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.

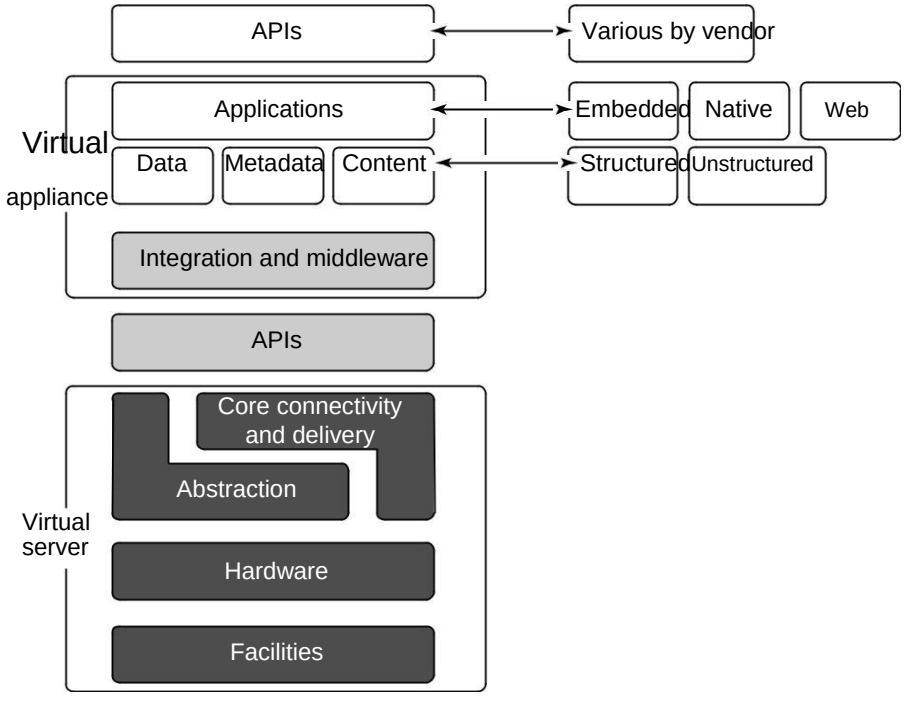
Platforms often come replete with tools and utilities to aid in application design and deployment. Depending upon the vendor, you may find developer tools for team collaboration, testing tools, instrumentation for measuring program performance and attributes, versioning, database and Web service integration, 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

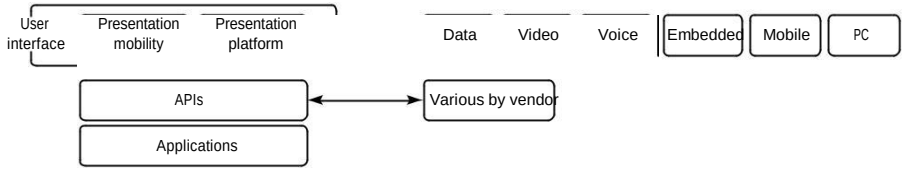
A virtual appliance is software that installs as middleware onto a virtual machine.



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 man-age 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 impor-tant 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 ven-dor's APIs will interoperate and exchange data. For now, the use of proprietary APIs results in ven-dor 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 configura-tion 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 sys-tem. 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 function-ality that isn't required by the appliance, because the appliance is meant to be used as is.

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 net-work. 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 network-ing 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

Part I: Examining the Value Proposition

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 offer-ings, products, and services. Infrastructure as a Service allows for the cre-ation of virtual computing systems or networks.

Software as a Service represents a hosted application that is universally avail-able 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 devel-opment 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 com-ponentized 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 authenti-cation and authorization services on distributed networks. Infrastructure and

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.

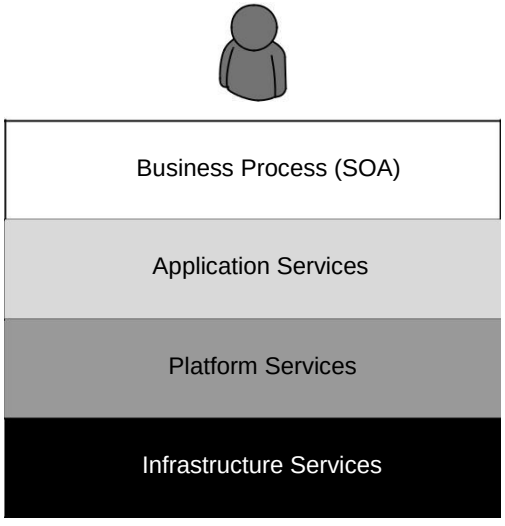
Defining Infrastructure as a Service (IaaS)

You can broadly partition cloud computing into four layers that form a cloud computing ecosys-tem, 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 virtual-ized in the cloud. In this particular model, the service vendor owns the equipment: servers, stor-age, 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 cli-ent'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.

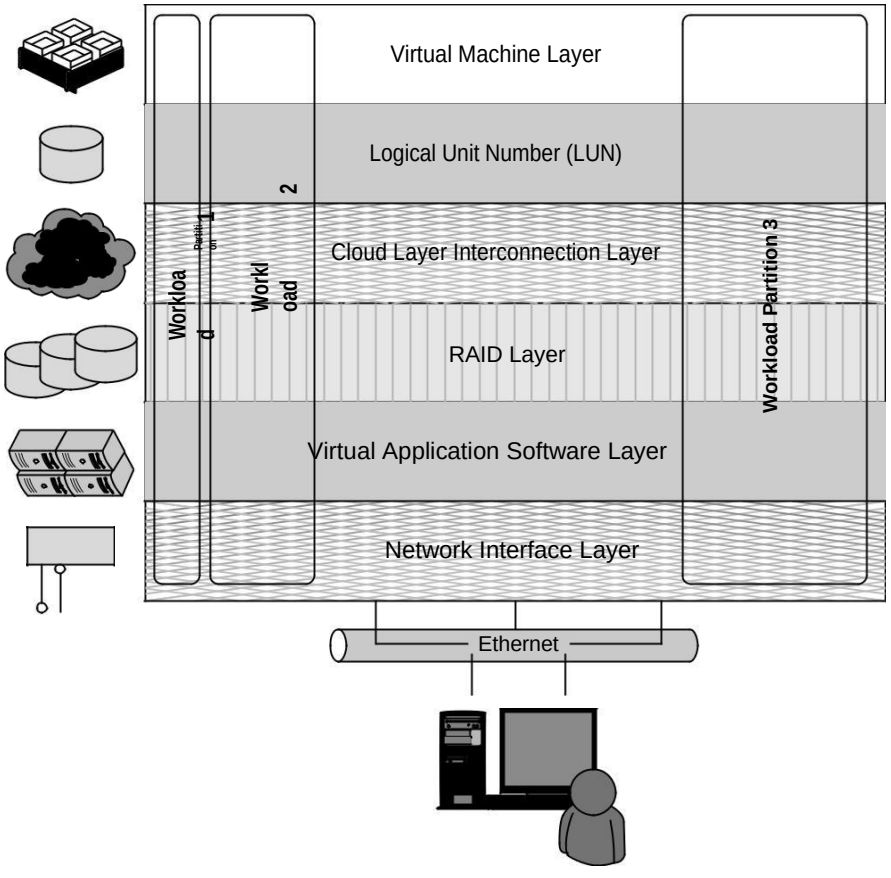
A client would reserve a machine equivalent required to run each of these workloads. The IaaS infrastructure runs these server instances in the data center that the service offers, drawing from a pool of virtualized machines, RAID storage, and network interface capacity. These three layers are expressions 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 tunnel-ing 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 notifica-tion 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

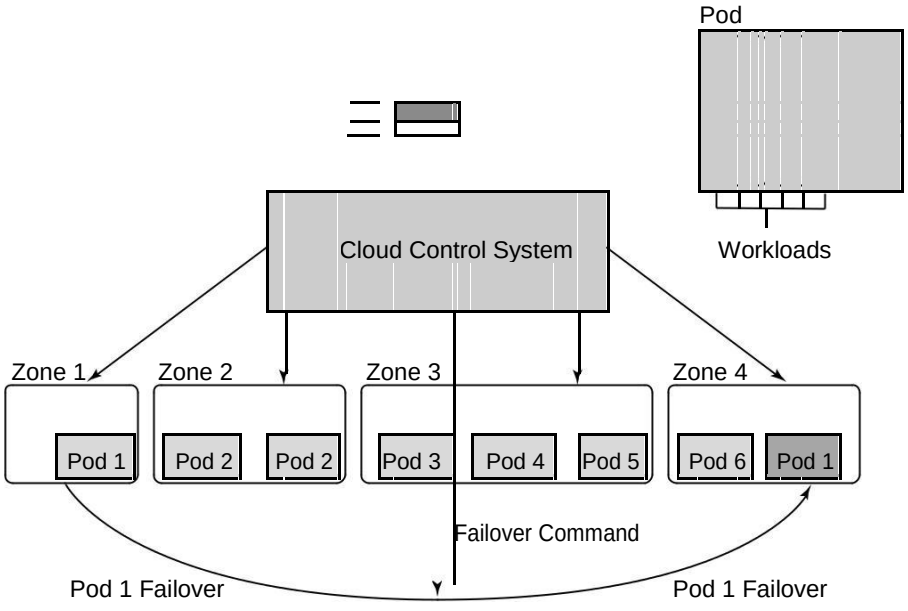
Workloads support a certain number of users, at which point you exceed the load that the instance sizing allows. When you reach the limit of the largest virtual machine instance possible, you must make a copy or clone of the instance to support additional users. A group of users within a particu-lar 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 appli-cation. 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 archi-tected. 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 sys-tem can't be. Silos just aren't as flexible as open systems and are subject to vendor lock-in.

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

---

Force.com, Salesforce.com's developer platform for its SaaS offerings, described in the next section. Force.com is an example of an add-on development environment.

A developer might write an application in a programming language like Python using the Google API. The vendor of the PaaS solution is in most cases the developer, who is offering a complete solution to the

customer. Google itself also serves as a PaaS vendor within this system, because it offers many of its Web service applications to customers as part of this service model. You can think of Google Maps, Google Earth, Gmail, and the myriad of other PaaS offerings as conforming to the PaaS service model, although these applications themselves are offered to customers under what is more aptly described as the Software as a Service (SaaS) model that is described below.

The difficulty with PaaS is that it locks the developer (and the customer) into a solution that is dependent upon the platform vendor. An application written in Python against Google's API using the Google App Engine is likely to work only in that environment. There is considerable vendor lock-in associated with a PaaS solution.

Defining Software as a Service (SaaS)

The most complete cloud computing service model is one in which the computing hardware and software, as well as the solution itself, are provided by a vendor as a complete service offering. It is referred to as the Software as a Service (SaaS) model. SaaS provides the complete infrastructure, software, and solution stack as the service offering. A good way to think about SaaS is that it is the cloud-based equivalent of shrink-wrapped software.

Software as a Service (SaaS) may be succinctly described as software that is deployed on a hosted service and can be accessed globally over the Internet, most often in a browser. With the exception of the user interaction with the software, all other aspects of the service are abstracted away.

Every computer user is familiar with SaaS systems, which are either replacements or substitutes for locally installed software. Examples of SaaS software for end-users are Google Gmail and Calendar, QuickBooks online, Zoho Office Suite, and others that are equally well known. SaaS applications come in all shapes and sizes, and include custom software such as billing and invoicing systems, Customer Relationship Management (CRM) applications, Help Desk applications, Human Resource (HR) solutions, as well as myriad online versions of familiar applications.

Many people believe that SaaS software is not customizable, and in many SaaS applications this is indeed the case. For user-centric applications such as an office suite, that is mostly true; those suites allow you to set only options or preferences. However, many other SaaS solutions expose Application Programming Interfaces (API) to developers to allow them to create custom composite applications. These APIs may alter the security model used, the data schema, workflow characteristics, and other fundamental features of the service's expression as experienced by the user. Examples of an SaaS platformwith an exposed API are Salesforce.com and Quicken.com. So SaaS does not necessarily mean that the software is static or monolithic.

SaaS characteristics

All Software as a Service (SaaS) applications share the following characteristics:

1. The software is available over the Internet globally through a browser on demand.
2. The typical license is subscription-based or usage-based and is billed on a recurring basis.

   In a small number of cases a flat fee may be changed, often coupled with a maintenance fee. Table 4.1 shows how different licensing models compare.
3. The software and the service are monitored and maintained by the vendor, regardless of where all the different software components are running.

   There may be executable client-side code, but the user isn't responsible for maintaining that code or its interaction with the service.
4. Reduced distribution and maintenance costs and minimal end-user system costs generally make SaaS applications cheaper to use than their shrink-wrapped versions.
5. Such applications feature automated upgrades, updates, and patch management and much faster rollout of changes.
6. SaaS applications often have a much lower barrier to entry than their locally installed competitors, a known recurring cost, and they scale on demand (a property of cloud computing in general).
7. All users have the same version of the software so each user's software is compatible with another's.
8. SaaS supports multiple users and provides a shared data model through a single-instance, multi-tenancy model.

   The alternative of software virtualization of individual instances also exists, but is less common.

TABLE 4.1

Shrink-Wrapped versus SaaS Licensing

	Shrink-Wrapped Software	Hybrid Model	SaaS
Licensing	Owned	Subscription (flat fee)	Metered subscription
Location	Locally installed	Available through an application	Cloud based
Management	Local IT staff	Application Service Provider (ASP)	Cloud vendor through a Service Level Agreement (SLA)

Defining Identity as a Service (IDaaS)

The establishment and proof of an identity is a central network function. An identity service is one that stores the information associated with a digital entity in a form that can be queried and man-aged for use in electronic transactions. Identity services have as their core functions: a data store, a query engine, and a policy engine that maintains data integrity.

Distributed transaction systems such as internetworks or cloud computing systems magnify the difficulties faced by identity management systems by exposing a much larger attack surface to an intruder than a private network does. Whether it is network traffic protection, privileged resource access, or some other defined right or privilege, the validated authorization of an object based on its identity is the central tenet of secure network design. In this regard, establishing identity may be seen as the key to obtaining trust and to anything that an object or entity wants to claim ownership of.

Services that provide digital identity management as a service have been part of internetworked systems from Day One. Like so many concepts in cloud computing, IDentity as a Service is a FLAVor (Four Letter Acronym) of the month, applied to services that already exist. The Domain Name Service can run on a private network, but is at the heart of the Internet as a service that provides identity authorization and lookup. The name servers that run the various Internet domains (.COM, .ORG, .EDU, .MIL, .TV, .RU, and so on) *are* IDaaS servers. DNS establishes the identity of a domain as belonging to a set of assigned addresses, associated with an owner and that owner's information, and so forth. If the identification is the assigned IP number, the other properties are its metadata.

You can categorize a myriad of services as IDaaS that run in the cloud. However, when most experts in the area of IDaaS define an identity service, they narrow the definition so the service must operate as a component according to the rules of a Service Oriented Architecture, as is defined in Chapter 13. This narrower definition restricts IDaaS to newer software services, services that interoperate, and therefore services that are standards based. It's best to keep this narrower definition in mind when you discuss IDaaS in a modern context.

What is an identity?

An identity is a set of characteristics or traits that make something recognizable or known. In com-puter network systems, it is one's digital identity that most concerns us. A digital identity is those attributes and metadata of an object along with a set of relationships with other objects that makes an object identifiable. Not all objects are unique, but by definition a digital identity must be unique, if only trivially so, through the assignment of a unique identification attribute. An identity must therefore have a context in which it exists.

This description of an identity as an object with attributes and relationships is one that programmer's would recognize. Databases store information and relationships in tables, rows, and columns, and the identity of information stored in this way conforms to the notion of an entity and a relationship—or alternatively under the notion of an object role model (ORM)—and database architects are always wrestling with the best way of reducing their data set to a basic set of identities. You can extend this notion to the idea of an identity having a profile and profiling services such as Facebook as being an extension of the notion of Identity as a Service in cloud computing.

An identity can belong to a person and may include the following:

- **Things you are:** Biological characteristics such as age, race, gender, appearance, and so forth
- **Things you know:** Biography, personal data such as social security numbers, PINs, where you went to school, and so on
- **Things you have:** A pattern of blood vessels in your eye, your fingerprints, a bank account you can access, a security key you were given, objects and possessions, and more
- **Things you relate to:** Your family and friends, a software license, beliefs and values, activities and endeavors, personal selections and choices, habits and practices, an iGoogle account, and more

To establish your identity on a network, you might be asked to provide a name and password, which is called a single-factor authentication method. More secure authentication requires the use of at least two-factor authentication; for example, not only name and password (things you know) but also a transient token number provided by a hardware key (something you have). To get to multifactor authentication, you might have a system that examines a biometric factor such as a fingerprint or retinal blood vessel pattern—both of which are essentially unique things you are. Multifactor authentication requires the outside use of a network security or trust service, and it is in the deployment of trust services that our first and most common IDaaS applications are employed in the cloud.

Of course, many things have digital identities. User and machine accounts, devices, and other objects establish their identities in a number of ways. For user and machine accounts, identities are created and stored in domain security databases that are the basis for any network domain, in directory services, and in data stores in federated systems. Network interfaces are identified uniquely by Media Access Control (MAC) addresses, which alternatively are referred to as Ethernet Hardware Addresses (EHAs). It is the assignment of a network identity to a specific MAC address that allows systems to be found on networks.

The manner in which Microsoft validates your installation of Windows and Office is called Windows Product Activation and creates an identification index or profile of your system, which is instructive. During activation, the following unique data items are retrieved:

- A 25-character software product key and product ID
- The uniquely assigned Global Unique Identifier or GUID
- PC manufacturer
- CPU type and serial number
- BIOS checksum
- Network adapter and its MAC address
- Display adapter
- SCSCI and IDE adapters
- RAM amount
- Hard drive and volume serial number

- Optical drive
- Region and language settings and user locale

From this information, a code is calculated, checked, and entered into the registration database. Each of these uniquely identified hardware attributes is assigned a weighting factor such that an overall sum may be calculated. If you change enough factors—NIC and CPU, display adapter, RAM amount, and hard drive—you trigger a request for a reactivation based on system changes. This activation profile is also required when you register for the Windows Genuine Advantage pro-gram. Windows Product Activation and Windows Genuine Advantage are cloud computing appli-cations, albeit proprietary ones. Whether people consider these applications to be services is a point of contention.

Networked identity service classes

To validate Web sites, transactions, transaction participants, clients, and network services—various forms of identity services—have been deployed on networks. Ticket or token providing services, certificate servers, and other trust mechanisms all provide identity services that can be pushed out of private networks and into the cloud.

Identity protection is one of the more expensive and complex areas of network computing. If you think about it, requests for information on identity by personnel such as HR, managers, and oth-ers; by systems and resources for access requests; as identification for network traffic; and the myr-iad other requirements mean that a significant percentage of all network traffic is supporting an identification service. Literally hundreds of messages on a network every minute are checking identity, and every Ethernet packet contains header fields that are used to identify the information it contains.

As systems become even more specialized, it has become increasingly difficult to find the security experts needed to run an ID service. So Identity as a Service or the related hosted (managed) identity services may be the most valuable and cost effective distributed service types you can subscribe to.

Identity as a Service (IDaaS) may include any of the following:

- Authentication services (identity verification)
- Directory services
- Federated identity
- Identity governance
- Identity and profile management
- Policies, roles, and enforcement
- Provisioning (external policy administration)
- Registration
- Risk and event monitoring, including audits
- Single sign-on services (pass-through authentication)

The sharing of any or all of these attributes over a network may be the subject of different govern-ment regulations and in many cases must be protected so that only justifiable parties may have access to the minimal amount that may be disclosed. This level of access defines what may be called an identity relationship.

Note

The Burton Group (http://www.burtongroup.com/**), a well-known computer industry analyst firm located in Midvale, Utah, has a trademark on the term IaaS as defined as Identity as a Service for use in the publication of their research in this area. The Burton Group is a well-known authority in the field of network infrastructure, particularly directory services and more recently in cloud computing. In this book, I use the term IaaS as applied to Infrastructure as a Service and use IDaaS to identify Identity as a Service.** ■

Identity system codes of conduct

Certain codes of conduct must be observed legally, and if not legally at the moment, then certainly on a moral basis. Cloud computing services that don't observe these codes do so at their peril. In working with IDaaS software, evaluate IDaaS applications on the following basis:

- **User control for consent:** Users control their identity and must consent to the use of their information.
- **Minimal Disclosure:** The minimal amount of information should be disclosed for an intended use.
- **Justifiable access:** Only parties who have a justified use of the information contained in a digital identity and have a trusted identity relationship with the owner of the information may be given access to that information.
- **Directional Exposure:** An ID system must support bidirectional identification for a pub-lic entity so that it is discoverable and a unidirectional identifier for private entities, thus protecting the private ID.
- **Interoperability:** A cloud computing ID system must interoperate with other identity ser-vices from other identity providers.
- **Unambiguous human identification:** An IDaaS application must provide an unambigu-ous mechanism for allowing a human to interact with a system while protecting that user against an identity attack.
- **Consistency of Service:** An IDaaS service must be simple to use, consistent across all its uses, and able to operate in different contexts using different technologies.

IDaaS interoperability

Identity as a Service provides an easy mechanism for integrating identity services into individual applications with minimal development effort, by allowing the identification logic and storage of an identity's attributes to be maintained externally. IDaaS applications may be separated from other distributed security systems by their compliance with SOA standards (as described in Chapter 13, "Understanding Service Oriented Architecture"), particularly if you want to have these services interoperate and be federated.

Therefore, cloud computing IDaaS applications must rely on a set of developing industry standards to provide interoperability. The following are among the more important of these services:

- **User centric authentication (usually in the form of information cards):** The OpenID and CardSpace specifications support this type of data object.
- **The XACML Policy Language:** This is a general-purpose authorization policy language that allows a distributed ID system to write and enforce custom policy expressions. XACML can work with SAML; when SAML presents a request for ID authorization, XACML checks the ID request against its policies and either allows or denies the request.
- **The SPML Provisioning Language:** This is an XML request/response language that is used to integrate and interoperate service provisioning requests. SPML is a standard of OASIS's Provision Services Technical Committee (PSTC) that conforms to the SOA architecture.
- **The XDAS Audit System:** The Distributed Audit Service provides accountability for users accessing a system, and the detection of security policy violations when attempts are made to access the system by unauthorized users or by users accessing the system in an unauthorized way.

User authentication

OpenID is a developing industry standard for authenticating "end users" by storing their digital identity in a common format. When an identity is created in an OpenID system, that information is stored in the system of any OpenID service provider and translated into a unique identifier. Identifiers take the form of a Uniform Resource Locator (URL) or as an Extensible Resource Identifier (XRI) that is authenticated by that OpenID service provider. Any software application that complies with the standard accepts an OpenID that is authenticated by a trusted provider. A very impressive group of cloud computing vendors serve as identity providers (or OpenID providers), including AOL, Facebook, Google, IBM, Microsoft, MySpace, Orange, PayPal, VeriSign, LiveJournal, Ustream, Yahoo!, and others.

The OpenID standard applies to the unique identity of the URL; it is up to the service provider to store the information and specify the forms of authentication required to successfully log onto the system. Thus an OpenID authorization can include not only passwords, but smart cards, hardware keys, tokens, and biometrics as well. OpenID is supported by the OpenID Foundation (http://openid.net/foundation/), a not-for-profit organization that promotes the technology.

These are samples of trusted providers and their URL formats:

- **Blogger:** <*username*>.blogger.com or <*blogid*>.blogspot.com
- **MySpace:** myspace.com/<*username*>
- **Google:** https://www.google.com/accounts/o8/id
- **Google Profile:** google.com/profiles/<*username*>
- **Microsoft:** accountservices.passport.net/
- **MyOpenID:** <*username*>.myopenid.com
- **Orange:** openid.orange.fr/username or simply orange.fr/
- **Verisign:** <*username*>.pip.verisinglabs.com
- **WordPress:** <*username*>.wordpress.com
- **Yahoo!:** openid.yahoo.com

After you have logged onto a trusted provider, that logon may provide you access to other Web sites that support OpenID. When you request access to a site through your browser (or another application that is referred to as a user-agent), that site serves as the "relying party" and requests of the server or server-agent that it verify the end-user's identifier. You won't need to log onto these other Web sites, if your OpenID is provided. Most trusted providers require that you indicate which Web sites you want to share your OpenID identifier with and the information is submitted automatically to the next site.

CardSpace is a Microsoft software client that is part of the company's Identity Metasystem and built into the Web Services Protocol Stack. This stack is built on the OASIS standards (WS-Trust, WS-Security, WS-SecurityPolicy, and WS-MetadataExchange), so any application that conforms with the OASIS WS- standards can interoperate with CardSpace. CardSpace was introduced with .NET Frameworks 3.0 and can be installed on Windows XP, Server 2003, and later. It is installed by default on Windows Vista and Windows 7.

CardSpace offers another way of authenticating users in the cloud. An Information Card may be requested with an HTML <OBJECT> tag, and the trusted Identity Provider then creates an encrypted and digitally signed token using the Security Token Service (STS) that is part of a WS-Trust request/ reply mechanism. CardSpace may be seen as an alternative mechanism to the use of OpenID and SAML and is used to sign into those services as well as Windows Live ID accounts.

Defining Compliance as a Service (CaaS)

Cloud computing by its very nature spans different jurisdictions. The laws of the country of a request's origin may not match the laws of the country where the request is processed, and it's possible that neither location's laws match the laws of the country where the service is provided. Compliance is much more than simply providing an anonymous service token to an identity so they can obtain access to a resource. Compliance is a complex issue that requires considerable expertise.

While Compliance as a Service (CaaS) appears in discussions, few examples of this kind of service exist as a general product for a cloud computing architecture. A Compliance as a Service applica-tion would need to serve as a trusted third party, because this is a man-in-the-middle type of ser-vice. CaaS may need to be architected as its own layer of a SOA architecture in order to be trusted. A CaaS would need to be able to manage cloud relationships, understand security policies and pro-cedures, know how to handle information and administer privacy, be aware of geography, provide an incidence response, archive, and allow for the system to be queried, all to a level that can be captured in a Service Level Agreement. That's a tall order, but CaaS has the potential to be a great value-added service.

In order to implement CaaS, some companies are organizing what might be referred to as "vertical clouds," clouds that specialize in a vertical market. Examples of vertical clouds that advertise CaaS capabilities include the following:

- **athenahealth** (http://www.athenahealth.com/) for the medical industry
- **bankserv** (http://www.bankserv.com/) for the banking industry
- **ClearPoint PCI** Compliance-as-a-Service for merchant transactions under the Payment Card Industry Data Security Standard
- **FedCloud** (http://www.fedcloud.com/) for government
- **Rackserve PCI** Compliant Cloud (http://www.rackspace.com/; another PCI CaaS service)

It's much easier to envisage a CaaS system built inside a private cloud where the data is under the control of a single entity, thus ensuring that the data is under that entity's secure control and that transactions can be audited. Indeed, most of the cloud computing compliance systems to date have been built using private clouds.

It is easy to see how CaaS could be an incredibly valuable service. A well-implemented CaaS ser-vice could measure the risks involved in servicing compliance and ensure or indemnify customers against that risk. CaaS could be brought to bear as a mechanism to guarantee that an e-mail con-formed to certain standards, something that could be a new electronic service of a network of national postal systems—and something that could help bring an end to the scourge of spam.

QUESTIONS

1. Define Cloud Computing. What is a Cloud? What are the deployment models of cloud computing? Explain them briefly.
2. Explain NIST model and Cloud Cube model in cloud computing. Explain the service models of cloud computing with examples of services/service providers.
3. What are the characteristics of cloud computing? Explain the cloud reference model.
4. Explain the benefits and advantages of cloud computing.
5. Explain the cloud computing architecture with a proper diagram. Explain briefly the architecture of each component in cloud computing.
6. Compare the virtualization approach and the traditional approach in cloud computing. Explain the virtualization model for cloud computing.
7. Give the basic concept of IaaS. Explain IaaS by throwing light on workload, partitioning of virtual private server instances, pods, aggregations and silos.
8. Give the basic concept of PaaS. Explain PaaS in terms of tools and development environment with examples.
9. Give the basic concept of SaaS. What are the characteristics of SaaS.
10. Explain Open SaaS, Identity as a Service (IDaaS), Compliance as a Service (CaaS). Give some examples of SaaS platform.

Module – II

Definition of Cloud Computing and its Basics

Understanding Abstraction and Virtualization

In this chapter, I discuss different technologies that create shared pools of resources. The key to creating a pool is to provide an abstraction mecha-nism so that a logical address can be mapped to a physical resource. Computers use this technique for placing files on disk drives, and cloud computing networks use a set of techniques to create virtual servers, virtual storage, virtual networks, and perhaps one day virtual applications. Abstraction enables the key benefit of cloud computing: shared, ubiquitous access.

In this chapter, you learn about how load balancing can be used to create high performance cloud-based solutions. Google.com's network is an example of this approach. Google uses commodity servers to direct traffic appropriately.

Another technology involves creating virtual hardware systems. An example of this type of approach is hypervisors that create virtual machine technologies. Several important cloud computing approaches use a strictly hardware-based approach to abstraction. I describe VMware's vSphere infrastructure in some detail, along with some of the unique features and technologies that VMware has developed to support this type of cloud.

Finally, I describe some approaches to making applications portable. Application portability is a difficult proposition, and work to make applica-tions portable is in its infancy. Two approaches are described, the Simple API and AppZero's Virtual Application Appliance (VAA). VAAs are contain-ers that abstract an application from the operating system, and they offer the potential to make an application portable from one platform to another.

Using Virtualization Technologies

The dictionary includes many definitions for the word "cloud." A cloud can be a mass of water droplets, gloom, an obscure area, or a mass of similar particles such as dust or smoke. When it comes to cloud computing, the definition that best fits the context is "a collection of objects that are grouped together." It is that act of grouping or creating a resource pool that is what succinctly differentiates cloud computing from all other types of networked systems.

Not all cloud computing applications combine their resources into pools that can be assigned on demand to users, but the vast majority of cloud-based systems do. The benefits of pooling resources to allocate them on demand are so compelling as to make the adoption of these technologies a priority. Without resource pooling, it is impossible to attain efficient utilization, provide reasonable costs to users, and proactively react to demand. In this chapter, you learn about the technologies that abstract physical resources such as processors, memory, disk, and network capacity into virtual resources.

When you use cloud computing, you are accessing pooled resources using a technique called virtualization. Virtualization assigns a logical name for a physical resource and then provides a pointer to that physical resource when a request is made. Virtualization provides a means to manage resources efficiently because the mapping of virtual resources to physical resources can be both dynamic and facile. Virtualization is dynamic in that the mapping can be assigned based on rapidly changing conditions, and it is facile because changes to a mapping assignment can be nearly instantaneous.

These are among the different types of virtualization that are characteristic of cloud computing:

- **Access:** A client can request access to a cloud service from any location.
- **Application:** A cloud has multiple application instances and directs requests to an instance based on conditions.
- **CPU:** Computers can be partitioned into a set of virtual machines with each machine being assigned a workload. Alternatively, systems can be virtualized through load-balanc-ing technologies.
- **Storage:** Data is stored across storage devices and often replicated for redundancy.

To enable these characteristics, resources must be highly configurable and flexible. You can define the features in software and hardware that enable this flexibility as conforming to one or more of the following mobility patterns:

- **P2V:** Physical to Virtual
- **V2V:** Virtual to Virtual
- **V2P:** Virtual to Physical
- **P2P:** Physical to Physical
- **D2C:** Datacenter to Cloud

- **C2C:** Cloud to Cloud
- **C2D:** Cloud to Datacenter
- **D2D:** Datacenter to Datacenter

The techniques used to achieve these different types of virtualization are the subject of this chapter. According to Gartner ("Server Virtualization: One Path that Leads to Cloud Computing," by Thomas J. Bittman, 10/29/2009, Research Note G00171730), virtualization is a key enabler of the first four of five key attributes of cloud computing:

- **Service-based:** A service-based architecture is where clients are abstracted from service providers through service interfaces.
- **Scalable and elastic:** Services can be altered to affect capacity and performance on demand.
- **Shared services:** Resources are pooled in order to create greater efficiencies.
- **Metered usage:** Services are billed on a usage basis.
- **Internet delivery:** The services provided by cloud computing are based on Internet protocols and formats.

Load Balancing and Virtualization

One characteristic of cloud computing is virtualized network access to a service. No matter where you access the service, you are directed to the available resources. The technology used to distribute service requests to resources is referred to as *load balancing.* Load balancing can be implemented in hardware, as is the case with F5's BigIP servers, or in software, such as the Apache mod_proxy_balancer extension, the Pound load balancer and reverse proxy software, and the Squid proxy and cache daemon. Load balancing is an optimization technique; it can be used to increase utilization and throughput, lower latency, reduce response time, and avoid system overload.

The following network resources can be load balanced:

- Network interfaces and services such as DNS, FTP, and HTTP
- Connections through intelligent switches
- Processing through computer system assignment
- Storage resources
- Access to application instances

Without load balancing, cloud computing would very difficult to manage. Load balancing provides the necessary redundancy to make an intrinsically unreliable system reliable through managed redi-rection. It also provides fault tolerance when coupled with a failover mechanism. Load balancing is nearly always a feature of server farms and computer clusters and for high availability applications.

A load-balancing system can use different mechanisms to assign service direction. In the simplest load-balancing mechanisms, the load balancer listens to a network port for service requests. When a request from a client or service requester arrives, the load balancer uses a scheduling algorithm to assign where the request is sent. Typical scheduling algorithms in use today are round robin and weighted round robin, fastest response time, least connections and weighted least connections, and custom assignments based on other factors.

A session ticket is created by the load balancer so that subsequent related traffic from the client that is part of that session can be properly routed to the same resource. Without this session record or persistence, a load balancer would not be able to correctly failover a request from one resource to another. Persistence can be enforced using session data stored in a database and replicated across multiple load balancers. Other methods can use the client's browser to store a client-side cookie or through the use of a rewrite engine that modifies the URL. Of all these methods, a session cookie stored on the client has the least amount of overhead for a load balancer because it allows the load balancer an independent selection of resources.

The algorithm can be based on a simple round robin system where the next system in a list of systems gets the request. Round robin DNS is a common application, where IP addresses are assigned out of a pool of available IP addresses. Google uses round robin DNS, as described in the next section.

Advanced load balancing

The more sophisticated load balancers are workload managers. They determine the current utilization of the resources in their pool, the response time, the work queue length, connection latency and capacity, and other factors in order to assign tasks to each resource. Among the features you find in load balancers are polling resources for their health, the ability to bring standby servers online (priority activation), workload weighting based on a resource's capacity (asymmetric loading), HTTP traffic compression, TCP offload and buffering, security and authentication, and packet shaping using content filtering and priority queuing.

An Application Delivery Controller (ADC) is a combination load balancer and application server that is a server placed between a firewall or router and a server farm providing Web services. An Application Delivery Controller is assigned a virtual IP address (VIP) that it maps to a pool of servers based on application specific criteria. An ADC is a combination network and application layer device. You also may come across ADCs referred to as a content switch, multilayer switch, or Web switch.

These vendors, among others, sell ADC systems:

- A10 Networks (http://www.a10networks.com/)
- Barracuda Networks (http://www.barracudanetworks.com/)
- Brocade Communication Systems (http://www.brocade.com/)
- Cisco Systems (http://www.cisco.com/)
- Citrix Systems (http://www.citrix.com/)

- F5 Networks (http://www.f5.com/)
- Nortel Networks (http://www.nortel.com/)
- Coyote Point Systems (http://www.coyotepoint.com/)
- Radware (http://www.radware.com/)

An ADC is considered to be an advanced version of a load balancer as it not only can provide the features described in the previous paragraph, but it conditions content in order to lower the work-load of the Web servers. Services provided by an ADC include data compression, content caching, server health monitoring, security, SSL offload and advanced routing based on current conditions. An ADC is considered to be an application accelerator, and the current products in this area are usually focused on two areas of technology: network optimization, and an application or frame-work optimization. For example, you may find ADC's that are tuned to accelerate ASP.NET or AJAX applications.

An architectural layer containing ADCs is described as an Application Delivery Network (ADN), and is considered to provide WAN optimization services. Often an ADN is comprised of a pair of redundant ADCs. The purpose of an ADN is to distribute content to resources based on applica-tion specific criteria. ADN provide a caching mechanism to reduce traffic, traffic prioritization and optimization, and other techniques. ADN began to be deployed on Content Delivery Networks (CDN) in the late 1990s, where it added the ability to optimize applications (application fluency) to those networks. Most of the ADC vendors offer commercial ADN solutions.

In addition to the ADC vendors in the list above, these are additional ADN vendors, among others:

- Akamai Technologies (http://www.akamai.com/)
- Blue Coat Systems (http://www.bluecoat.com/)
- CDNetworks (http://www.cdnetworks.com/)
- Crescendo Networks (http://www.crescendonetworks.com/)
- Expand Networks (http://www.expand.com/)
- Juniper Networks (http://www.juniper.net/)

Google's cloud is a good example of the use of load balancing, so in the next section let's consider how Google handles the many requests that they get on a daily basis.

The Google cloud

According to the Web site tracking firm Alexa (http://www.alexa.com/topsites), Google is the single most heavily visited site on the Internet; that is, Google gets the most hits. The invest-ment Google has made in infrastructure is enormous, and the Google cloud is one of the largest in use today. It is estimated that Google runs over a million servers worldwide, processes a billion search requests, and generates twenty petabytes of data per day.

Google is understandably reticent to disclose much about its network, because it believes that its infrastructure, system response, and low latency are key to the company's success. Google never gives datacenter tours to journalists, doesn't disclose where its datacenters are located, and obfus-cates the locations of its datacenters by wrapping them in a corporate veil. Thus, the discretely named Tetra LLC (limited liability company) owns the land for the Council Bluffs, Iowa, site, and Lapis LLC owns the land for the Lenoir, North Carolina, site. This makes Google infrastructure watching something akin to a sport to many people.

So what follows is what we think we know about Google's infrastructure and the basic idea behind how Google distributes its traffic by pooling IP addresses and performing several layers of load balancing.

Google has many datacenters around the world. As of March 2008, Rich Miller of DataCenterKnowledge.com wrote that Google had at least 12 major installations in the United States and many more around the world. Google supports over 30 country specific versions of the Google index, and each localization is supported by one or more datacenters. For example, Paris, London, Moscow, Sao Paolo, Tokyo, Toronto, Hong Kong, Beijing and others support their coun-tries' locale. Germany has three centers in Berlin, Frankfurt, and Munich; the Netherlands has two at Groningen and Eemshaven. The countries with multiple datacenters store index replicas and support network peering relationships. Network peering helps Google have low latency connec-tions to large Internet hubs run by different network providers.

You can find a list of sites as of 2008 from Miller's FAQ at http://www.datacenter knowledge.com/archives/2008/03/27/google-data-center-faq/.

Based on current locations and the company's statements, Google's datacenters are sited based on the following factors (roughly in order of importance):

1. Availability of cheap and, if possible, renewable energy
2. The relative locations of other Google datacenters such that the site provides the lowest latency response between sites
3. Location of nearby Internet hubs and peering sites
4. A source of cooling water
5. The ability to purchase a large area of land surrounding the site

   Speculation on why Google purchases large parcels of land ranges from creating a buffer zone between the datacenter and surrounding roads and towns or possibly to allow for building wind farms when practical.
6. Tax concessions from municipalities that lower Google's overhead

Google maintains a pool of hundreds of IP addresses, all of which eventually resolve to its Mountain View, California, headquarters. When you initiate a Google search, your query is sent to a DNS server, which then queries Google's DNS servers. The Google DNS servers examine the pool of addresses to determine which addresses are geographically closest to the query origin and uses a round robin policy to assign an IP address to that request. The request usually goes to the nearest

datacenter, and that IP address is for a cluster of Google servers. This DNS assignment acts as a first level of IP virtualization, a pool of network addresses have been load balanced based on geography.

A Google cluster can contain thousands of servers. Google servers are racks of commodity (low cost) 1U or 2U servers containing 40 to 80 servers per rack with one switch per rack. Each switch is connected to a core gigabit switch. Google servers run a customized version of Linux with appli-cations of several types.

When the query request arrives at its destination, a Google cluster is sent to a load balancer, which forwards that request to a Squid proxy server and Web cache dameon. This is the second level of IP distribution, based on a measure of the current system loading on proxy servers in the cluster. The Squid server checks its cache, and if it finds a match to the query, that match is returned and the query has been satisfied. If there is no match in the Squid cache, the query is sent to an indi-vidual Google Web Server based on current Web server utilizations, which is the third level of network load balancing, again based on utilization rates.

It is the Google Web Servers that perform the query against the Google index and then format the results into an HTML page that is returned to the requester. This procedure then performs two more levels of load balancing based on utilization rates.

Google's secret sauce is its in-memory inverted index and page rank algorithm. Google's GoogleBot (a spider or robot) crawls the Web and collects document information. Some details of the search and store algorithm are known. Google looks at the title and first few hundred words and builds a word index from the result. Indexes are stored on an index server.

Some documents are stored as snapshots (PDF, DOC, XLS, and so on), but lots of information is not addressed in the index. Each document is given a unique ID ("docid"), and the content of the document is disassembled into segments called shards, subjected to a data compression scheme and stored on a document server. The entire index is maintained in system memory partitioned over each instance of the index's replicas. A page rank is created based on the significant links to that page.

Queries are divided into word lists, and the Google algorithm examines the words and the relation-ships of one word to another. Those word relationships are mapped against the main index to cre-ate a list of documents, a feature called an inverted index. In an inverted index, words are mapped to documents, which can be done very quickly when the index is fully kept in memory.

The Web server takes the result of a query and composes the Web page from that result. Ads included on the page are from ad servers, which provide Google's AdSense and AdWords services. The query also is presented to a spelling server to provide suggestions for alternative spellings to include in the search result. Certain keywords, data input patterns, and other strings are recog-nized as having special operational significance. For example entering "2 plus 2" initiates Google's calculator program, and a ten-digit number returns a reverse phone lookup using the phonebook program. These programs are supported by special application servers.

Google doesn't use hardware virtualization; it performs server load balancing to distribute the processing load and to get high utilization rates. The workload management software transfers the workload from a failed server over to a redundant server, and the failed server is taken offline. Multiple instances of various Google applications are running on different hosts, and data is stored on redundant storage systems.

Understanding Hypervisors

Load balancing virtualizes systems and resources by mapping a logical address to a physical address. Another fundamental technology for abstraction creates virtual systems out of physical systems. If load balancing is like playing a game of hot potato, then virtual machine technologies is akin to playing slice and dice with the potato.

Given a computer system with a certain set of resources, you can set aside portions of those resources to create a virtual machine. From the standpoint of applications or users, a virtual machine has all the attributes and characteristics of a physical system but is strictly software that emulates a physical machine. A system virtual machine (or a hardware virtual machine) has its own address space in memory, its own processor resource allocation, and its own device I/O using its own virtual device drivers. Some virtual machines are designed to run only a single application or process and are referred to as process virtual machines.

A virtual machine is a computer that is walled off from the physical computer that the virtual machine is running on. This makes virtual machine technology very useful for running old versions of operating systems, testing applications in what amounts to a sandbox, or in the case of cloud computing, creating virtual machine instances that can be assigned a workload. Virtual machines provide the capability of running multiple machine instances, each with their own operating system.

From the standpoint of cloud computing, these features enable VMMs to manage application provisioning, provide for machine instance cloning and replication, allow for graceful system failover, and provide several other desirable features. The downside of virtual machine technologies is that having resources indirectly addressed means there is some level of overhead.

Virtual machine types

A low-level program is required to provide system resource access to virtual machines, and this program is referred to as the hypervisor or Virtual Machine Monitor (VMM). A hypervisor running on bare metal is a Type 1 VM or native VM. Examples of Type 1 Virtual Machine Monitors are LynxSecure, RTS Hypervisor, Oracle VM, Sun xVM Server, VirtualLogix VLX, VMware ESX and ESXi, and Wind River VxWorks, among others. The operating system loaded into a virtual machine is referred to as the guest operating system, and there is no constraint on running the same guest on multiple VMs on a physical system. Type 1 VMs have no host operating system because they are installed on a bare system.

An operating system running on a Type 1 VM is a full virtualization because it is a complete simu-lation of the hardware that it is running on.

Note

Not all CPUs support virtual machines, and many that do require that you enable this support in the BIOS. For example, AMD-V processors (code named Pacifica) and Intel VT-x (code named Vanderpool) were the first of these vendor's 64-bit offerings that added this type of support. ■

Some hypervisors are installed over an operating system and are referred to as Type 2 or hosted VM. Examples of Type 2 Virtual Machine Monitors are Containers, KVM, Microsoft Hyper V, Parallels Desktop for Mac, Wind River Simics, VMWare Fusion, Virtual Server 2005 R2, Xen, Windows Virtual PC, and VMware Workstation 6.0 and Server, among others. This is a very rich product category. Type 2 virtual machines are installed over a host operating system; for Microsoft Hyper-V, that operating system would be Windows Server. In the section that follows, the Xen hypervisor (which runs on top of a Linux host OS) is more fully described. Xen is used by Amazon Web Services to provide Amazon Machine Instances (AMIs).

Figure 5.1 shows a diagram of Type 1 and Type 2 hypervisors.

On a Type 2 VM, a software interface is created that emulates the devices with which a system would normally interact. This abstraction is meant to place many I/O operations outside the virtual environment, which makes it both programmatically easier and more efficient to execute device I/O than it would be inside a virtual environment. This type of virtualization is sometimes referred to as *paravirtualization*, and it is found in hypervisors such as Microsoft's Hyper-V and Xen. It is the host operating system that is performing the I/O through a para-API.

FIGURE 5.1

VMware's vSphere cloud computing infrastructure model

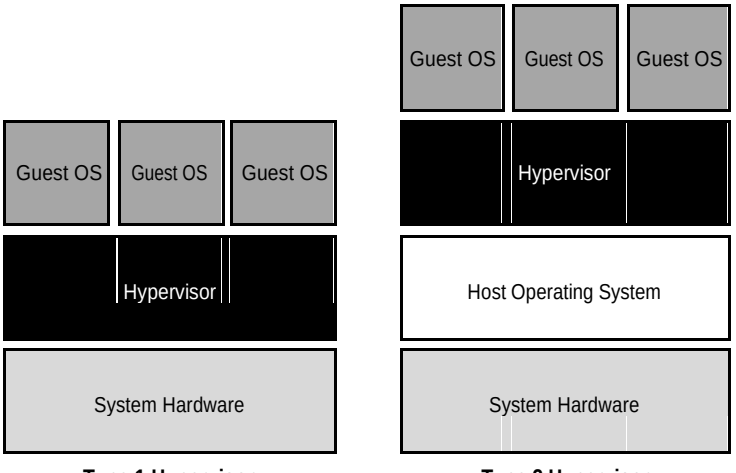


Figure 5.2 shows the difference between emulation, paravirtualization, and full virtualization. In emulation, the virtual machine simulates hardware, so it can be independent of the underlying system hardware. A guest operating system using emulation does not need to be modified in any way. Paravirtualization requires that the host operating system provide a virtual machine interface for the guest operating system and that the guest access hardware through that host VM. An operating system running as a guest on a paravirtualization system must be ported to work with the host interface. Finally, in a full virtualization scheme, the VM is installed as a Type 1 Hypervisor directly onto the hardware. All operating systems in full virtualization communicate directly with the VM hypervisor, so guest operating systems do not require any modification. Guest operating systems in full virtualization systems are generally faster than other virtualization schemes.

The Virtual Machine Interface (VMI) open standard (http://vmi.ncsa.uiuc.edu/) that VMware has proposed is an example of a paravirtualization API. The latest version of VMI is 2.1, and it ships as a default installation with many versions of the Linux operating system.

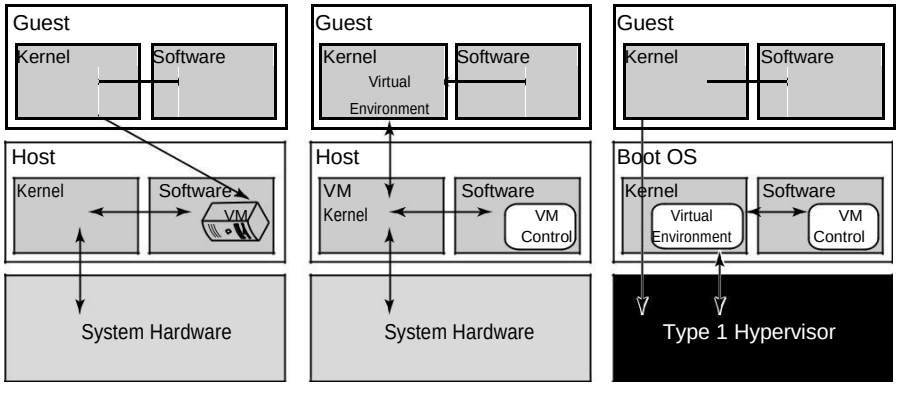
Note

Wikipedia maintains a page called "Comparison of platform virtual machines" at http://en.wikipedia.org/wiki/Comparison_of_platform_virtual_machines**. The page contains a table of features of the most common Virtual Machine Managers. ■**

You are probably familiar with process or application virtual machines. Most folks run the Java Virtual Machine or Microsoft's .NET Framework VM (called the Common Language Runtime or CLR) on their computers. A process virtual machine instantiates when a command begins a pro-cess, the VM is created by an interpreter, the VM then executes the process, and finally the VM exits the system and is destroyed. During the time the VM exists, it runs as a high-level abstraction.

FIGURE 5.2

Emulation, paravirtualization, and full virtualization types



Emulation **Paravirtualization** **Full Virtualization**

Applications running inside an application virtual machine are generally slow, but these programs are very popular because they provide portability, offer rich programming languages, come with many advanced features, and allow platform independence for their programs. Although many cloud computing applications provide process virtual machine applications, this type of abstraction isn't really suitable for building a large or high-performing cloud network, with one exception.

The exception is the process VMs that enable a class of parallel cluster computing applications. These applications are high-performance systems where the virtual machine is operating one process per cluster node, and the system maintains the necessary intra-application communications over the network interconnect. Examples of this type of system are the Parallel Virtual Machine (PVM; see http://www.csm.ornl.gov/pvm/pvm_home.html) and the Message Passing Interface (MPI; see http://www.mpi-forum.org/). Some people do not consider these appli-cation VMs to be true virtual machines, noting that these applications can still access the host operating system services on the specific system on which they are running. The emphasis on using these process VMs is in creating a high-performance networked supercomputer often out of heterogeneous systems, rather than on creating a ubiquitous utility resource that characterizes a cloud network.

Some operating systems such as Sun Solaris and IBM AIX 6.1 support a feature known as *operating system virtualization.* This type of virtualization creates virtual servers at the operating system or kernel level. Each virtual server is running in its own virtual environment (VE) as a virtual private server (VPS). Different operating systems use different names to describe these machine instances, each of which can support its own guest OS. However, unlike true virtual machines, VPS must all be running the same OS and the same version of that OS. Sun Solaris 10 uses VPS to create what is called Solaris Zones. With IBM AIX, the VPS is called a System Workload Partition (WPAR). This type of virtualization allows for a dense collection of virtual machines with relatively low overhead. Operating system virtualization provides many of the benefits of virtualization previously noted in this section.

VMware vSphere

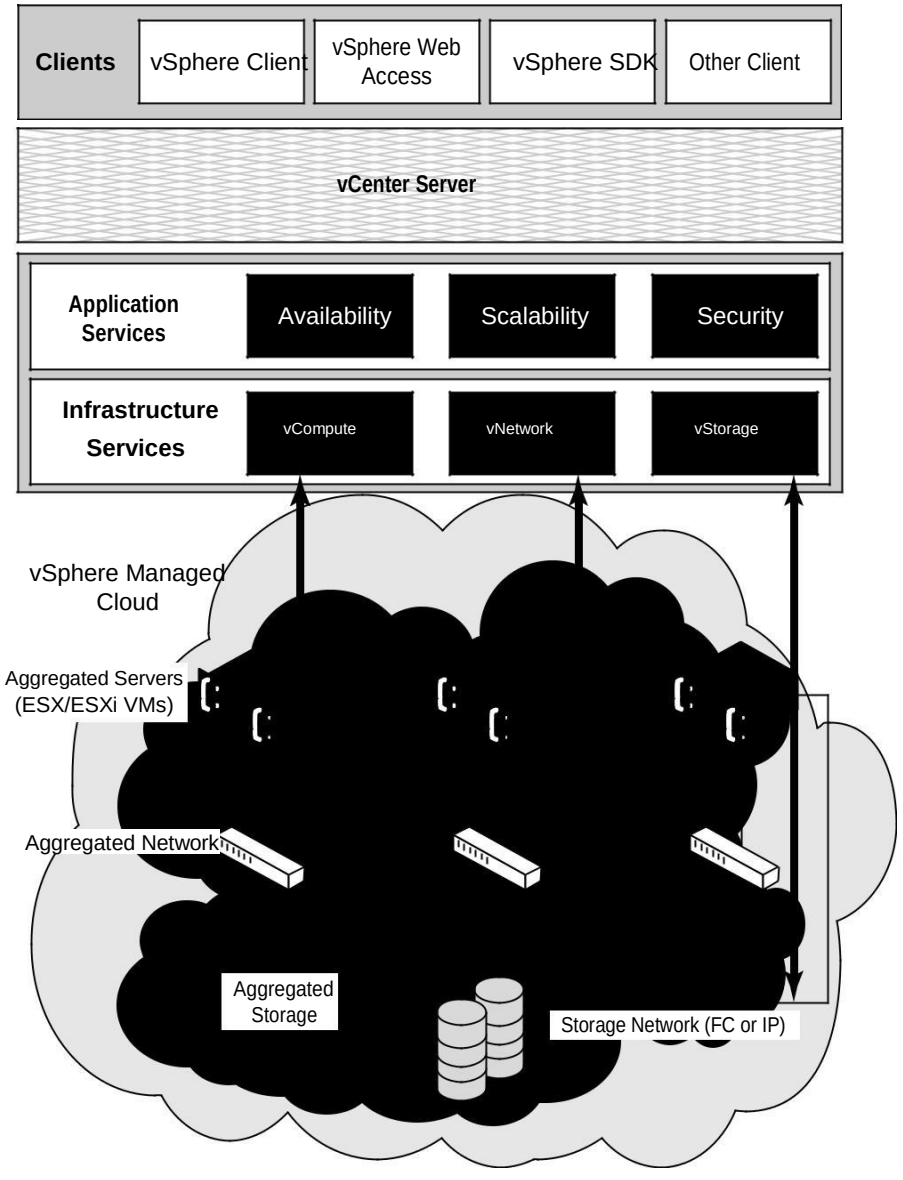
VMware vSphere is a management infrastructure framework that virtualizes system, storage, and networking hardware to create cloud computing infrastructures. vSphere is the branding for a set of management tools and a set of products previously labeled VMware Infrastructure. vSphere pro-vides a set of services that applications can use to access cloud resources, including these:

- **VMware vCompute:** A service that aggregates servers into an assignable pool
- **VMware vStorage:** A service that aggregates storage resources into an assignable pool
- **VMware vNetwork:** A service that creates and manages virtual network interfaces
- **Application services:** Such as HA (High Availability) and Fault Tolerance
- **vCenter Server:** A provisioning, management, and monitoring console for VMware cloud infrastructures

Figure 5.3 shows an architectural diagram of a vSphere cloud infrastructure.

FIGURE 5.3

VMware's vSphere cloud computing infrastructure model



A vSphere cloud is a pure infrastructure play. The virtualization layer that abstracts processing, memory, and storage uses the VMware ESX or ESXi virtualization server. ESX is a Type 1 hypervi-sor; it installs over bare metal (a clean system) using a Linux kernel to boot and installs the vmker-nel hypervisor (virtualization kernel and support files). When the system is rebooted, the vmkernel loads first, and then the Linux kernel becomes the first guest operating system to run as a virtual machine on the system and contains the service console.

VMware is a very highly developed infrastructure and the current leader in this industry. A number of important add-on products are available for cloud computing applications. These are among the more notable products:

- **Virtual Machine File System (VMFS):** A high-performance cluster file system for an ESX/ESXi cluster.
- **VMotion:** A service that allows for the migration of a virtual machine from one physical server to another physical server while the virtual server runs continuously and without any interruption of ongoing transactions.

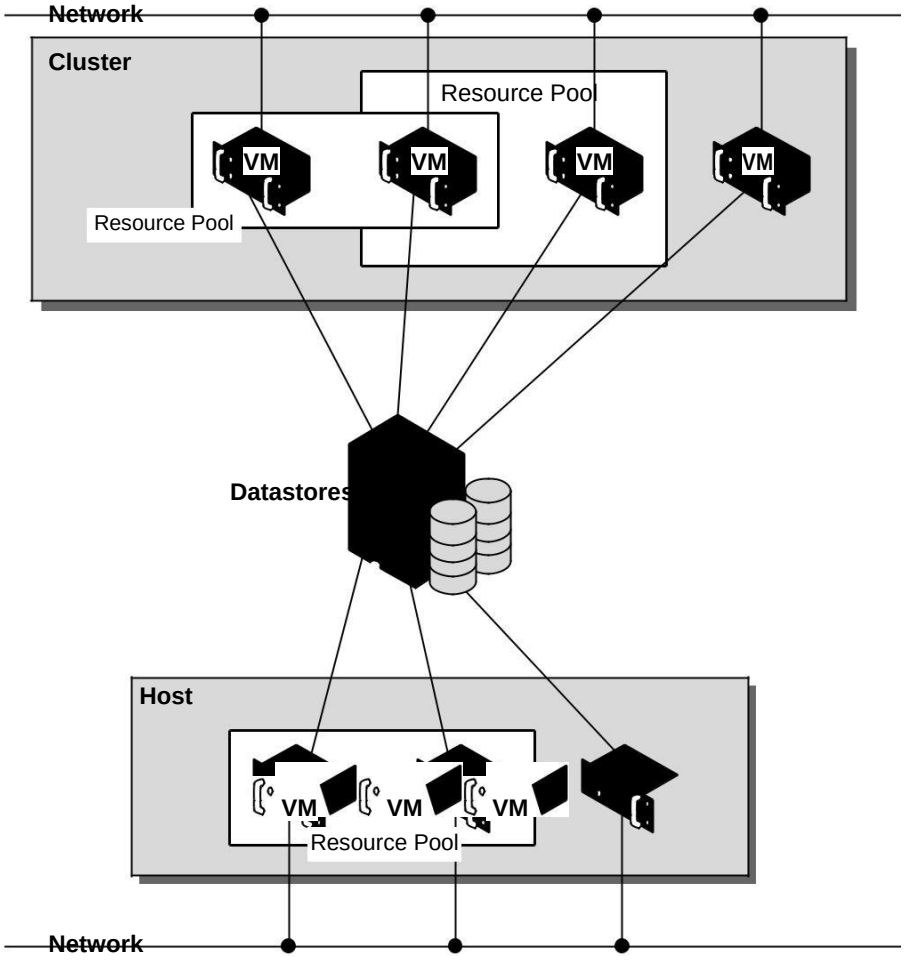
  The ability to live migrate virtual machines is considered to be a technological tour de force and a differentiator from other virtual machine system vendors.
- **Storage VMotion:** A product that can migrate files from one datastore to another datas-tore while the virtual machine that uses the datastore continues to run.
- **Virtual SMP:** A feature that allows a virtual machine to run on two or more physical pro-cessors at the same time.
- **Distributed Resource Scheduler (DRS):** A system for provisioning virtual machines and load balancing processing resources dynamically across the different physical systems that are in use. A part of the DRS called the distributed power management (DPM) module can manage the power consumption of systems.
- **vNetwork Distributed Switch (DVS):** A capability to maintain a network runtime state for virtual machines as they are migrated from one physical system to another. DVS also monitors network connections, provides firewall services, and enables the use of third-party switches such as the Cisco Nexus 1000V to manage virtual networks.

You can get a better sense of how the different resources are allocated by vSphere into a virtual set of components by examining Figure 5.4. Physical computers can be standalone hosts or a set of clustered systems. In either case, a set of virtual machines can be created that is part of a single physical system or spans two or more physical systems.

You can define a group of VMs as a Resource Pool (RP) and, by doing so, manage those virtual machines as a single object with a single policy. Resource Pools can be placed into a hierarchy or nested and can inherit properties of their parent RP. As more hosts or cluster nodes are added or removed, vSphere can dynamically adjust the provisioning of VMs to accommodate the policy in place. This fine tuning of pooled resources is required to accommodate the needs of cloud comput-ing networks.

FIGURE 5.4

Virtual infrastructure elements



The datastore shown at the center of Figure 5.4 is a shared storage resource. These storage resources can be either Direct Attached Storage (DAS) of a server using SCSI, SAS, or SATA con-nections, Fibre Channel disk arrays/SANs, iSCSI disk arrays/SANs, or Network Attached Storage (NAS) disk arrays. Although the lines drawn between the datastore and different VMs indicate a direct connection, with the exception of DAS, the other storage types are shared storage solutions.

Storage virtualization is most commonly achieved through a mapping mechanism where a logical storage address is translated into a physical storage address. Block-based storage such as those used

in SANs use a feature called a Logical Unit Identifier (LUN**)** with specific addresses stored in the form of an offset called the Logical Block Address (LBA). The address space mapping then maps the address of the logical or virtual disk (vdisk) to the logical unit on a storage controller. Storage virtualization may be done in software or in hardware, and it allows requests for virtualized storage to be redirected as needed.

Similarly, network virtualization abstracts networking hardware and software into a virtual net-work that can be managed. A virtual network can create virtual network interfaces (VNICs) or virtual LANs (VLANS) and can be managed by a hypervisor, operating system, or external manage-ment console. In a virtualized infrastructure such as the one presented in this section, internal net-work virtualization is occurring and the hypervisor interacts with networking hardware to create a pseudo-network interface. External network virtualization can be done using network switches and VLAN software.

The key feature that makes virtual infrastructure so appealing for organizations implementing a cloud computing solution is flexibility. Instantiating a virtual machine is a very fast process, typi-cally only a few seconds in length. You can make machine images of systems in the configuration that you want to deploy or take snapshots of working virtual machines. These images can be brought on-line as needed.

Understanding Machine Imaging

In the preceding sections, you have seen how the abstractions that cloud computing needs can be achieved through redirection and virtualization. A third mechanism is commonly used to provide system portability, instantiate applications, and provision and deploy systems in the cloud. This third mechanism is through storing the state of a systems using a system image.

Cross-Ref

Backing up to the cloud often involves imaging or snapshot applications; this process is described in Chapter 15, "Working with Cloud-Based Storage." ■

A system image makes a copy or a clone of the entire computer system inside a single container such as a file. The system imaging program is used to make this image and can be used later to restore a system image. Some imaging programs can take snapshots of systems, and most allow you to view the files contained in the image and do partial restores.

Note

The one open standard for storing a system image is the Open Virtualization Format (OVF; see http://www. dmtf.org/standards/published_documents/DSP0243_1.1.0.pdf**) that is published by the Distributed Task Format (DMTF;** http://www.dmtf.org/**). Some notable virtualization vendors, such as VMware, Microsoft, Citrix, and Oracle (Sun), are supporting this effort. ■**

A prominent example of a system image and how it can be used in cloud computing architectures is the Amazon Machine Image (AMI) used by Amazon Web Services to store copies of a virtual machine. Because this is a key feature of Amazon's Elastic Compute Cloud and is discussed in detail in Chapter 9, I briefly mention it here. An AMI is a file system image that contains an operating system, all appropriate device drivers, and any applications and state information that the working virtual machine would have.

When you subscribe to AWS, you can choose to use one of its hundreds of canned AMIs or to create a custom system and capture that system's image to an AMI. An AMI can be for public use under a free distribution license, for pay-per-use with operating systems such as Windows, or shared by an EC2 user with other users who are given the privilege of access.

Cross-Ref

Refer to Chapter 9, "Using Amazon Web Services," for more information about AMIs and their uses in the EC2 service. ■

The AMI file system is not a standard bit-for-bit image of a system that is common to many disk imaging programs. AMI omits the kernel image and stores a pointer to a particular kernel that is part of the AWS kernel library. Among the choices are Red Hat Linux, Ubuntu, Microsoft Windows, Solaris, and others. Files in AMI are compressed and encrypted, and an XML file is written that describes the AMI archive. AMIs are typically stored in your Amazon S3 (Simple Storage System) buckets as a set of 10MB chunks.

Machine images are sometimes referred to as "virtual appliances"—systems that are meant to run on virtualization platforms. AWS EC2 runs on the Xen hypervisor, for example. The term *virtual appliance* is meant to differentiate the software image from an operating virtual machine. The system image contains the operating system and applications that create an environment. Most virtual appliances are used to run a single application and are configurable from a Web page. Virtual appliances are a relatively new paradigm for application deployment, and cloud computing is the major reason for the interest in them and for their adoption. This area of WAN application portability and deployment, and of WAN optimization of an application based on demand, is one with many new participants. Certeon (http://www.certeon.com/), Expand Networks (http://www.expand.com/), and Replify (http://www.replify.com/) are three vendors offering optimization appliances for VMware's infrastructure.

Porting Applications

Cloud computing applications have the ability to run on virtual systems and for these systems to be moved as needed to respond to demand. Systems (VMs running applications), storage, and network assets can all be virtualized and have sufficient flexibility to give acceptable distributed WAN application performance. Developers who write software to run in the cloud will undoubtedly want the ability to port their applications from one cloud vendor to another, but that is a much more difficult proposition. Cloud computing is a relatively new area of technology, and the major vendors have technologies that don't interoperate with one another.

The Simple Cloud API

If you build an application on a platform such as Microsoft Azure, porting that application to Amazon Web Services or GoogleApps may be difficult, if not impossible. In an effort to create an interoperability standard, Zend Technologies has started an open source initiative to create a common application program interface that will allow applications to be portable. The initiative is called the Simple API for Cloud Application Services (http://www.simplecloud.org/), and the effort has drawn interest from several major cloud computing companies. Among the founding supporters are IBM, Microsoft, Nivanix, Rackspace, and GoGrid.

Simple Cloud API has as its goal a set of common interfaces for:

- **File Storage Services:** Currently Amazon S3, Windows Azure Blob Storage, Nirvanix, and Local storage is supported by the Storage API. There are plans to extend this API to Rackspace Cloud Files and GoGrid Cloud Storage.
- **Document Storage Services:** Amazon SimpleDB and Windows Azure Table Storage are currently supported. Local document storage is planned.
- **Simple Queue Services:** Amazon SQS, Windows Azure Queue Storage, and Local queue services are supported.

Zend intends to add the interface to their open source PHP Framework (http://www.frame work.zend.com) as the Zend_Cloud framework component. Vendors such as Microsoft and IBM are supplying adapters that will use part of the Simple Cloud API for their cloud application services.

AppZero Virtual Application Appliance

Applications that run in datacenters are captive to the operating systems and hardware platforms that they run on. Many datacenters are a veritable Noah's Ark of computing. So moving an applica-tion from one platform to another isn't nearly as simple as moving a machine image from one sys-tem to another.

The situation is further complicated by the fact that applications are tightly coupled with the operating systems on which they run. An application running on Windows, for example, isn't isolated from other applications. When the application loads, it often loads or uses different Dynamic Link Libraries (DLL), and it is through the sharing or modification of DLLs that Windows applications get themselves in trouble. Further modifications include modifying the registry during installation. These factors make it difficult to port applications from one platform to another without lots of careful work. If you are a Platform as a Service (PaaS) application developer, you are packaging a complete software stack that includes not only your application, but the operating system and application logic and rules as well. Vendor lock-in for you application is assured.

The ability to run an application from whatever platform you want is not one of the characteristics of cloud computing, but you can imagine that it is a very attractive proposition. While the Simple Cloud API is useful for applications written in PHP, other methods may be needed to make applications easily portable. One company working on this problem is AppZero (http://www.appzero.com/), and its solution is called the Virtual Application Appliance (VAA).

The AppZero solution creates a virtual application appliance as an architectural layer between the Windows or the UNIX operating system and applications. The virtualization layer serves as the mediator for file I/O, memory I/O, and application calls and response to DLLs, which has the effect of sandboxing the application. The running application in AppZero changes none of the registry entries or any of the files on the Windows Server.

VAA creates a container that encapsulates the application and all the application's dependencies within a set of files; it is essentially an Application Image for a specific OS. Dependencies include DLL, service settings, necessary configuration files, registry entries, and machine and network settings. This container forms an installable server-side application stack that can be run after installation, but has no impact on the underlying operating system. VAAs are created using the AppZero Creator wizard, managed with the AppZero Admin tool, and may be installed using the AppZero Director, which creates a VAA runtime application. If desired, an application called AppZero Dissolve removes the VAA virtualization layer from the encapsulated application and installs that application directly into the operating system.

Note

Microsoft App-V (http://www.microsoft.com/windows/enterprise/products/mdop/app-v. aspx**) and VMware ThinApp (**http://www.vmware.com/products/thinapp/**) are two application deliv-ery platforms, but their main focus is on desktop installations and not on server deployment in the cloud. ■**

Installations can be done over the network after the AppZero application appliance is installed. Therefore, with this system, you could run applications on the same Windows Server and elimi-nate one application from interfering with another; applications would be much more easily ported from one Windows system to another. AppZero's approach provides the necessary abstraction layer that frees an application from its platform dependence.

An interesting use of VAAs involves segmenting an application into several VAAs, some of which are read-only runtime components, while others can be modified. When backing up or replicating VAAs in a cloud, you would need to synchronize only those VAAs that are modified. In many instances, the portion of an application that changes is only a very small component of large appli-cations, which means that this technique can greatly reduce the amount of data required to repli-cate a VM in the cloud.

AppZero envisages using VAAs to create what it calls a *stateless cloud.* In a stateless cloud, the appli-cation's state information is stored on a network share where it is available to run on different cloud systems as needed. This approach allows the cloud system to run with a VM containing a clean operating system (like AWS does now) and provisioned by the VAA. This approach should greatly reduce the number of complete system images that cloud vendors and cloud users should need to store to support their work; it also should make the running of applications on secure, well-performing VEs easier to achieve.

Summary

In this chapter, you learned about some of the more important characteristics of cloud computing networks and applications, including ubiquitousness and on-demand service. To enable a cloud service, you need to create a pool of resources you can call on. The key techniques for enabling this are abstraction and virtualization. Abstraction maps a logical identity or address to a physical identity or address. Changes to the underlying systems, therefore, do not affect the client requesting a service.

Several different methods for abstraction have been considered. A widely used technique is load balancing. With load balancing, system requests are directed to appropriate systems on demand. All large cloud networks use some form of load balancing. You learned about some of the details of Google's load balancing for queries.

Another technology virtualizes hardware. You learned about the different types of hypervisors—software that can serve as a virtualization layer for operating systems accessing the underlying hardware. As an example of hardware virtualization VMware's vSphere infrastructure was considered. vSphere can create virtual machines, virtual datastores, and virtual networks, and move these resources about while the system is active. vSphere is a potent cloud-building technology.

System imaging also can be useful in creating and instantiating machine instances. A brief explana-tion of Amazon Machine Instances was given.

Finally, the topic of application portability was considered. Applications are hard to move from platform to platform, because they are bound up with the operating system on which they run. Eventually, applications will be as portable as virtual machines. A cloud programming interface was described, as was an application delivery appliance.

In Chapter 6, "Capacity Planning," the idea of system workloads is described. Understanding this concept allows you to scale your systems correctly, choose the right type of infrastructure, and do availability planning. Some of the key performance metrics for cloud computing "right sizing" are described.

Exploring Platform as a Service

The Platform as a Service model provides the tools within an environ-ment needed to create applications that can run in a Software as a Service model. For this reason, some overlap between vendors has created Software as a Service products, and those vendors have broadened their services to make their Web applications more customizable. Salesforce. com, the largest CRM application service company in the world, is an exam-ple, with Force.com being its PaaS (Platform as a Service) offering.

Applications developed in PaaS systems can be composite business applications, data portals, or mashups with data derived from multiple sources. PaaS environments can offer integrated lifecycle management or *anchored lifecycle applications.* An integrated system provides a broad range of tools for customization, whereas an anchored system is based on already established software.

Application frameworks are a particularly powerful tool for creating cloud computing applications. For this reason, many vendor products are based on this model. In other chapters, you learn about Google AppEngine and Windows Azure Platform. This chapter presents several examples of PaaS systems that can create captive hosted applications, portable applications, extended blogs or content management systems, or rich Internet data applications. Some of the sites you learn about in this chapter with PaaS tools include Drupal, Eccentex AppBase, LongJump, SquareSpace, and WaveMaker.

Each of these systems or tools presents a very different aspect of cloud appli-cation development. What all these tools have in common is that they are standards-based.

Defining Services

In many ways, the Platform as a Service model is the most interesting of all the hosted services in cloud computing. IaaS offers a service that is akin to installing an application on a computer. That computer is virtual, of course, but it is still a computer. By the time you are using an SaaS model, the software is pretty well mapped out for you. You can do some modest customization, some branding perhaps, but the software's capabilities and design has largely been worked out.

With Platform as a Service systems, you are given a toolkit to work with, a virtual machine to run your software on, and it is up to you to design the software and its user-facing interface in a way that is appropriate to your needs. So PaaS systems range from full-blown developer platforms like Windows Azure Platform to systems like Drupal, Squarespace, Wolf, and others where the tools are modules that are very well developed and require almost no coding. Many Content Management Systems (CMS) are essentially PaaS services where you get standard parts and can build Web sites and other software like Tinker Toys.

Thus you find that PaaS models span a broad range of services, including these, among others:

- **Application development:** A PaaS platform either provides the means to use programs you create in a supported language or offers a visual development environment that writes the code for you.
- **Collaboration:** Many PaaS systems are set up to allow multiple individuals to work on the same projects.
- **Data management:** Tools are provided for accessing and using data in a data store.
- **Instrumentation, performance, and testing:** Tools are available for measuring your applications and optimizing their performance.
- **Storage:** Data can be stored in either the PaaS vendor's service or accessed from a third-party storage service.
- **Transaction management:** Many PaaS systems provide services such as transaction man-agers or brokerage service for maintaining transaction integrity.

PaaS systems exist to allow you to create software that can be hosted as SaaS systems or to allow for the modification of existing SaaS applications. You've seen many examples of PaaS systems already, and whole chapters are dedicated to vendor-specific PaaS platforms. The next chapter describes the Google AppEngine, which is a system for deploying Web applications on Google infrastructure. Chapter 10 describes the Windows Azure Platform with its emphasis on creating Windows applications using the .NET Framework on Microsoft infrastructure.

A good PaaS system has certain desirable characteristics that are important in developing robust, scalable, and hopefully portable applications. On this list would be the following attributes:

- Separate of data management from the user interface
- Reliance on cloud computing standards

- An integrated development environment (IDE)
- Lifecycle management tools
- Multi-tenant architecture support, security, and scalability
- Performance monitoring, testing, and optimization tools

The more vibrant the associated market of a PaaS's third-party add-ons, applications, tools, and services, the better they are. These extras allow you to extend your application by buying function-ality, which is almost always cheaper than having to roll your own.

Salesforce.com versus Force.com: SaaS versus PaaS

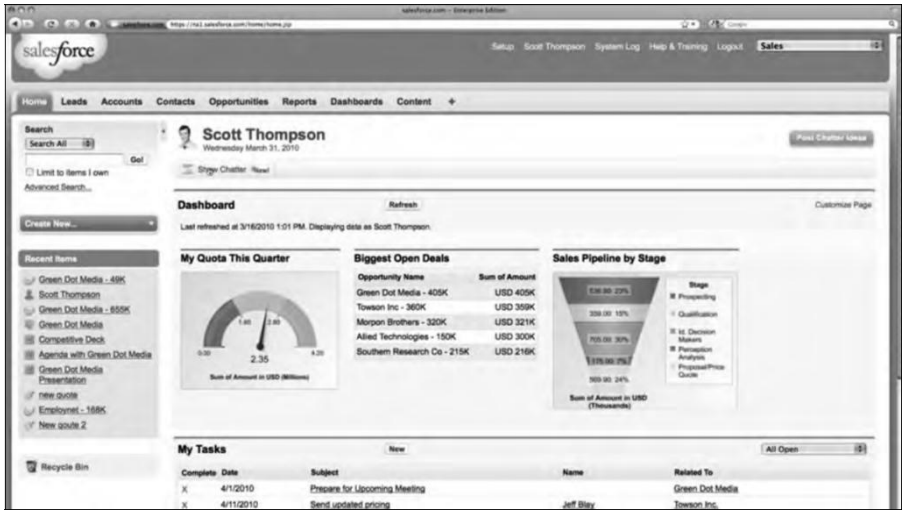
There can be no better example illustrating the difference between a SaaS and PaaS system than that of Salesforce.com and Force.com. Salesforce.com is a Web application suite that is an SaaS. Force.com is Salesforce.com's PaaS platform for building your own services.

Salesforce.com was formed by several Oracle employees in 1999 to create a hosted Customer Relationship Management (CRM) system. CRM has long been one of Oracle's core database ser-vices. The Salesforce.com team created hosted software based on a cloud computing model: pay as you go, simple to use, and multifunctional. The Salesforce.com platform looks like a typical Web site such as Amazon.com, with a multi-tabbed interface—each tab being an individual application.

Shown in Figure 7.1 is a Salesforce.com portal with the multi-tabbed interface exposing the differ-ent applications.

FIGURE 7.1

In Salesforce.com, each tab is an application, and data is shared. Shown here is a dashboard view.



Some of the applications included in the site are:

- Accounts and Contact
- Analytics and Forecasting
- Approvals and Workflow
- Chatter (Instant Messaging/Collaboration)
- Content Library
- E-mail and Productivity
- Jigsaw Business Data
- Marketing and Leads
- Opportunities and Quotes
- Partner Relationship
- Sales
- Service and Support

Which tabs you see, and how capable each hosted application is, depends on the level of service you purchase from Salesforce.com, as well as the particular type of bundle you buy. Salesforce.com tailors its SaaS for individual industries.

As Salesforce.com developed its SaaS production, it became obvious that many customers wanted to extend their Salesforce.com applications beyond what an SaaS offering would allow. Salesforce.com developed a PaaS platform known as Force.com, which allows developers to create applications that could be added to Salesforce.com's offerings and hosted on Salesforce.com's infrastructure.

Figure 7.2 shows the Force.com platform page at Salesforce.com.

Force.com uses a Java-based programming language called Apex for its application building, and it has an interface builder called Visualforce that allows a developer to create interfaces using HTML, Flex, and AJAX. Visualforce uses an XML-type language in its visual interface builder. Using the Force.com platform, more than 1,000 applications have been created and are offered for sale on Salesforce.com's AppExchange, which has greatly enhanced its PaaS offerings. These applications can show up as customizable tabs for different functions in customer applications or as a set of S-controls that are JavaScript widgets.

Because Salesforce.com is browser-based, it is platform-independent. However, the company has extended its audience to mobile devices, such as the Android, Blackberry, iPhone, and Windows Mobile Devices. It also has a server product that supports Salesforce.com applications in-house called the Resin Application Server.

Force.com has been a major hit and has served as the model from many of the PaaS systems of today. The company Salesforce.com is a recognized thought leader in the field of cloud computing. It is a $1.3 billion company as of 2009, with over 2 million subscribers.

Force.com's Web site (http://www.salesforce.com/platform/) leads to a set of developer tools as well as a gallery of sites built on this PaaS.

Application development

A PaaS provides the tools needed to construct different types of applications that can work together in the same environment. These are among the common application types:

- Composite business applications
- Data portals
- Mashups of multiple data sources

A *mashup* is a Web page that displays data from two or more data sources. The various landmarks and overlays you find in Google Earth, or annotated maps, are examples of mashups.

These applications must be able to share data and to run in a multi-tenant environment. To make applications work together more easily, a common development language such as Java or Python is usually offered. The more commonly used the language is, the more developers and developer ser-vices are going to be available to help users of platform applications. The use of application frame-works such as Ruby on Rails is useful in making application building easier and more powerful.

Most of the application building tools in this chapter create their own frameworks. Many are based on visual tools, and often these tools allow developers to extend applications using a common language for Web application development. These applications almost always adopt a Service Oriented Architecture model and use SOAP/REST with XML data exchange.

All PaaS application development must take into account lifecycle management. As an application ages, it must be upgraded, migrated, grown, and eventually phased out or ported. Many PaaS vendors offer systems that are integrated lifecycle development platforms. That is, the vendor provides a full software development stack for the programmer to use, and it isn't expected that the devel-oper will need to go outside of the service to create his application.

An integrated lifecycle platform includes the following:

- The virtual machine and operating system (often offered by an IaaS)
- Data design and storage
- A development environment with defined Application Programming Interfaces
- Middleware
- Testing and optimization tools
- Additional tools and services

Google AppEngine, Microsoft Windows Azure Platform, Eccentex AppBase, LongJump, and Wolf are examples of integrated lifecycle platforms. The latter three services are described in this chap-ter. Refer to Chapter 8 to read about AppEngine, and see Chapter 10 to learn about Azure.

Some PaaS services allow developers to modify existing software. These services are referred to as *anchored lifecycle platforms.* Examples of an anchored lifecycle platform are QuickBooks.com and Salesforce.com. The applications in these two services are fixed, but developers can customize which applications the users see, how those applications are branded, and a number of features associated with the different applications. An anchored service offers less customization, but has a faster development cycle and may be less prone to software errors.

Using PaaS Application Frameworks

Application frameworks provide a means for creating SaaS hosted applications using a unified development environment or an *integrated development environment* (IDE). PaaS IDEs run the gamut from a tool that requires a dedicated programming staff to create and run to point-and-click graphical interfaces that any knowledgeable computer user can navigate and create something useful with.

In selecting the six different examples of Web sites and application building PaaS systems, a full range of user experience is considered. Many Web sites are based on the notion of information management and organization; they are referred to as *content management systems* (CMS). A database is a content management system, but the notion of a Web site as a CMS adds a number of

special features to the concept that includes rich user interaction, multiple data sources, and extensive customization and extensibility. The Drupal CMS was chosen as an example of this type of PaaS because it is so extensively used and has broad industry impact, and it is a full-strength devel-oper tool.

Whereas Drupal is used in major Web sites and organizes vast amounts of information, the site Squarespace.com was chosen to illustrate a point-and-click CMS system aimed at supporting individuals, small businesses, and other small organizations. Squarespace is often associated with blogging tools (as is Drupal), but it is more than that. Squarespace works with photos, imports information from other social tools, and allows very attractive Web sites to be created by average users.

Caution

The portability of the applications you create in a PaaS is an extremely valuable feature. If your service goes out of business, being able to port an application by simply redeploying that application to another IaaS can be a lifesaver. ■

Eccentex AppBase, LongJump, and Wolf were chosen as examples of developer-oriented services aimed at users and developers who want to create Web-based applications based on Service Oriented Architecture protocols and services. These services vary in some details, but they have these common characteristics:

- They separate data-handling from presentation (user interface).
- They offer tools for establishing business objects or entities and the relationships between them.
- They support the incorporation of business rules, logic, and actions.
- They provide tools for creating data entry controls (forms), views, and reports.
- They provide instrumentation, tools for measuring application performance.
- They support packaging and deployment of applications.

These services differ in which language they use, support for different rendering technologies, and in other features. For the most part, they provide point-and-click tools where snippets of code provide exception programming. These services are extensible and customizable through application code. The goal of these services is to create portable applications, although each service includes a hosting platform for developed applications. Some cloud application platforms such as WorkXpress (http://www.workxpress.com) describe their environment as a 5GL PaaS (Fifth Generation) as opposed to something like Force.com, which they call a 3GL/4GL PaaS because 5GL environments have no programming requirement and you can host your application anywhere.

A 5GL programming language solves problems by acting on constraints and inputs and then uses intelligence to solve the problem. By comparison a 4GL programming language requires the programmer to build modules to solve specific problems. For a description of early programming language generations you may want to read the following reference: http://en.wikipedia.org/wiki/Programming_language_generations.

Drupal

Drupal (http://drupal.org/) is a content management system (CMS) that is used as the backend to a large number of Web sites worldwide. The software is an open-source project that was created in the PHP programming language. Drupal is really a programming environment for managing content, and it has elements of blogging and collaboration software as part of its distri-bution. Drupal is offered to the public under the GNU General Public License version 2 and is used by many prominent Web sites. The Drupal core is the standard distribution, with the current version being 6.19; version 7.0 is in preview.

Drupal is in this section because it is a highly extensible way to create Web sites with rich features. Druplas has a large developer community that has created nearly 6,000 third-party add-ons called *contrib modules.* Several thousand Drupal developers worldwide come together twice a year at the DrupalCon convention. It's a vibrant community of users and developers.

The number of Web sites that use Drupal is really quite remarkable, and many of them are very well known. Drupal is very popular with government agencies and with media companies, but its reach extends into nearly any industry, organization, and business type you can think of. Some of these sites are beautifully constructed. A short list of sites includes att.com, data.gov.uk, gouverne-ment.fr, intel.com, lucasfilms.com, mattel.com, thenation.com, whitehouse.gov, and ubuntu.com. Drupal has a gallery of screenshots of sites and features on its Web site, but for a better look at some of the more attractive sites, go to the Showcase of Popular Web sites Developed Using Drupal CMS (http://artatm.com/2010/02/showcase-of-popular-website-developed-using-drupal/), shown in Figure 7.3.

You find Drupal applications running on any Web server that can run PHP 4.4.0 and later. The most common deployments are on Apache, but you also can find Drupal on Microsoft IIS and other Unix Web servers. To store content, Drupal must be used with a database. Because LAMP installations are a standard Web deployment platform, the database most often used is MySQL. Other SQL databases work equally well.

The Drupal core by itself contains a number of modules that provide for the following:

- Auto-updates
- Blogs, forums, polls, and RSS feeds
- Multiple site management
- OpenID authentication
- Performance optimization through caching and throttling
- Search
- User interface creation tools
- User-level access controls and profiles
- Themes
- Traffic management
- Workflow control with events and triggers

Artatm.com has a gallery of some of the more attractive and well-known sites built with Drupal.

Drupal is modular and exposes its functionality through a set of published APIs. The contrib mod-ules can be added to Drupal to replace other modules, enhance capabilities, or provide entirely new features. Third-party modules include messaging systems, visual editors, a content construc-tion kit (CCK) for database schema extension, views, and panels. CCK Fields API is in the latest version of Drupal, version 7.0.

Drupal is reputed to be somewhat difficult to learn, and new versions often break old features. It is much more widely used than its competitor Joomla! (http://www.joomla.org/), and Drupal seems to have better performance than Joomla! as well. Another open source competitor in the content management space is eZ Publish (http://ez.no/).

Eccentex AppBase 3.0

Eccentex is a Culver City, California, company founded in 2005 that has a PaaS development platform for Web applications based on SOA component architecture to create what it calls Cloudware applications using its AppBase architecture. Figure 7.4 shows the AppBase platform page.

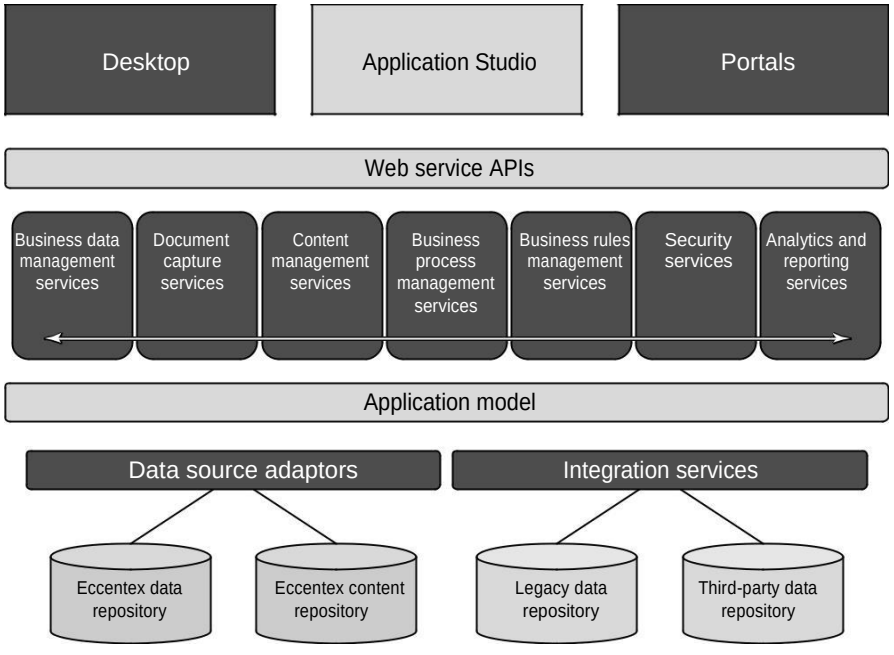
The Eccentex AppBase (http://www.eccentex.com/platform/platform.html) PaaS application delivery platform creates SOA applications that work on several different IaaS vendors.

AppBase includes a set of different tools for building these applications, including the following:

- **Business Objects Build:** This object database has the ability to create rich data objects and create relationships between them.
- **Presentation Builder:** This user interface (UI) builder allows you to drag and drop visual controls for creating Web forms and data entry screens and to include the logic necessary to automate what the user sees.
- **Business Process Designer:** This tool is used to create business logic for your application. With it, you can manage workflow, integrate modules, create rules, and validate data.
- **Dashboard Designer:** This instrumentation tool displays the real-time parameters of your application in a visual form.
- **Report Builder:** This output design tool lets you sort, aggregate, display, and format report information based on the data in your application.
- **Security Roles Management:** This allows you to assign access rights to different objects in the system, to data sets, fields, desktop tabs, and reports. Security roles can be assigned in groups without users, and users can be added later as the application is deployed.

Figure 7.5 shows the AppBase architecture with the various tools identified. You can view a set of screenshots that illustrate the different tools and some features in the build process at http://www.eccentex.com/platform/screenshots.html.

AppBase's architecture with the different tools and modules shown

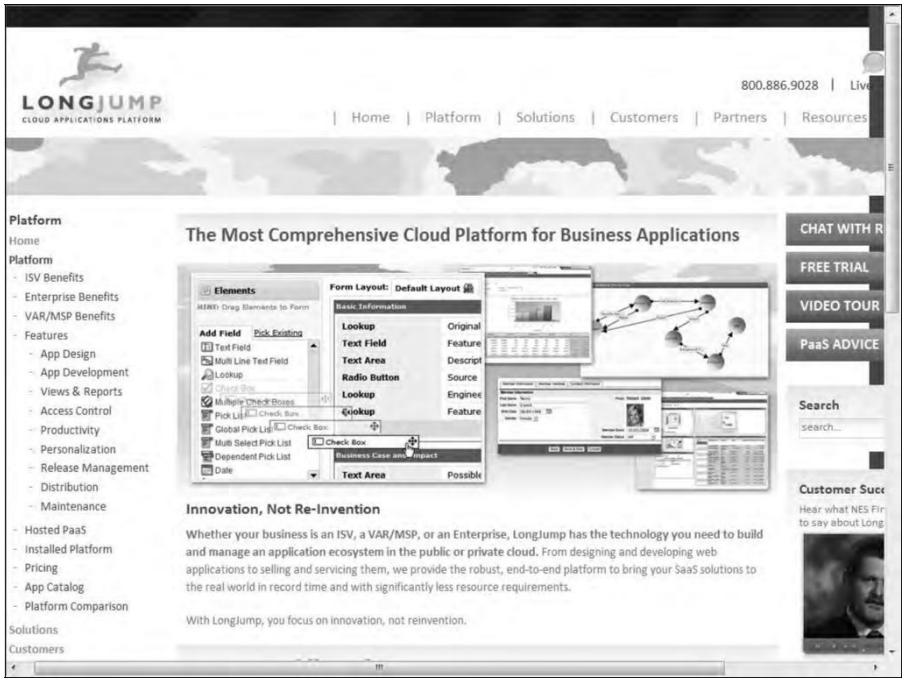


Applications that you create are deployed with the AppBase Application Revision Management console. The applications you create in AppBase, according to the company, may be integrated with Amazon S3 Web Services (storage), Google AppEngine (PaaS), Microsoft Windows Azure (PaaS), Facebook, and Twitter.

LongJump

LongJump (http://www.longjump.com/) is a Sunnyvale, California, company hosting service created in 2003 with a PaaS application development suite. Its development environment is based on Java and uses REST/SOAP APIs. Figure 7.6 shows the LongJump platform page.

LongJump's PaaS (http://www.longjump.com/index.php?option=com_content&view=article&id=8&Itemid=57) is based on standard Java/JavaScript, SOAP, and REST.



LongJump creates browser-based Web applications that are database-enabled. Like other products mentioned in this section, LongJump comes with an Object Model Viewer, forms, reports, layout tools, dashboards, and site management tools. Access control is based on role- and rule-based access, and it allows for data-sharing between teams and between tenants. LongJump comes with a security policy engine that has user and group privileges, authentication, IP range blocking, SSO, and LDAP interoperability. Applications are packaged using a packaging framework that can sup-port a catalog system, XML package file descriptions, and a distribution engine.

LongJump extends Java and uses a Model-View-Controller architecture (MVC) for its framework in the Developer Suite. The platform uses Java Server Pages (JSP), Java, and JavaScript for its various components and its actions with objects built with Java classes. Objects created in custom classes are referenced using POJO (Plain Old Java Object). Localization is supported using a module called the Translation Workbench that includes specified labels, errors, text, controls, and messaging text files (and header files) that allow them to be modified by a translation service to support additional languages. The development environment supports the Eclipse (http://www.eclipse.org/) plug-in for creating widgets using Java standard edition.

Squarespace

Squarespace (http://www.squarespace.com/), shown in Figure 7.7, is an example of a next-generation Web site builder and deployment tool that has elements of a PaaS development environment. The applications are built using visual tools and deployed on hosted infrastructure.

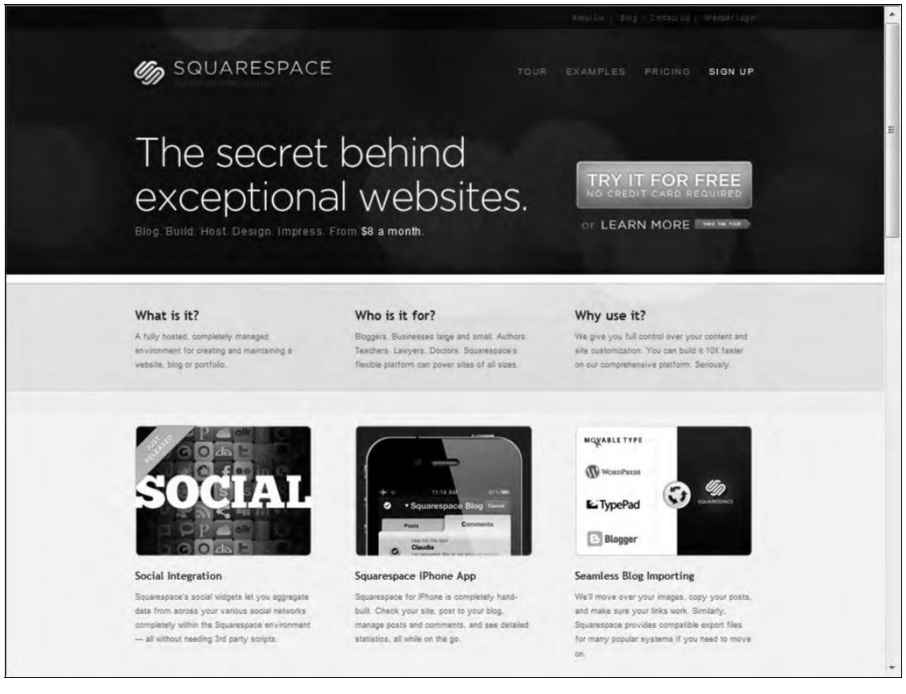
Squarespace presents itself, among other things, as:

- A blogging tool
- A social media integration tool
- A photo gallery
- A form builder and data collector
- An item list manager
- A traffic and site management and analysis tool

The platform has more than 20 core modules that you can add to your Web site. Squarespace sites can be managed on the company's iPhone app.

FIGURE 7.7

Squarespace lets you create beautiful hosted Web sites with a variety of capabilities with visual tools alone.



With Squarespace, users have created some very visually beautiful sites. Users tend to fall into these categories: personal Web sites, portfolios, and business brand identification. Although Squarespace positions itself as a competitor to blogging sites such as Wordpress (http:// wordpress.org/), Tumblr (http://www.tumblr.com/), Posterous (https:// posterous.com/), and other sites of their ilk, the site borders on a full content management system with a variety of useful and eclectic features.

WaveMaker

WaveMaker (http://www.wavemaker.com/) is a visual rapid application development envi-ronment for creating Java-based Web and cloud Ajax applications. The software is open-source and offered under the Apache license. WaveMaker is a WYSIWYG (What You See is What You Get) drag-and-drop environment that runs inside a browser. The metaphor used to build applica-tions is described as the Model-View-Controller system of application architecture. In this regard, WaveMaker has some similarities to PowerBuilder (http://www.sybase.com/products/ internetappdevttools/powerbuilder).

Figure 7.8 shows the WaveMaker home page. A gallery of features is accessible from that page.

FIGURE 7.8

WaveMaker is a visual development environment for creating Java-based cloud applications.

WaveMaker is a framework that creates applications that can interoperate with other Java frameworks and LDAP systems, including the following:

- Dojo Toolkit 1.0 (http://dojotoolkit.org/), a JavaScript library or toolbox
- LDAP directories
- Microsoft Active Directory
- POJO (Plain Old Java Object)
- Spring Framework (http://www.springsource.org/), an open-source application framework for Java that now also includes ACEGI

The visual builder tool is called Visual Ajax Studio, and the development server is called the WaveMaker Rapid Deployment Server for Java applications. When you develop within the Visual Ajax Studio, a feature called LiveLayout allows you to create applications while viewing live data. The data schema is prepared within a part of the tool called LiveForms. Mashups can be created using the Mashup Tool, which integrates applications using Java Services, SOAP, REST, and RSS to access databases.

Applications developed in WaveMaker run on standard Java servers such as Tomcat, DojoToolkit, Spring, and Hibernate. A 4GL version of WaveMaker also runs on Amazon EC2, and the develop-ment environment can be loaded on an EC2 instance as one of its machine images.

Wolf Frameworks

Many application frameworks like Google AppEngine and the Windows Azure Platform are tied to the platform on which they run. You can't build an AppEngine application and port it to Windows Azure without completely rewriting the application. There isn't any particular necessity to build an application framework in this way, but it suits the purpose of these particular vendors: for Google to have a universe of Google applications that build on the Google infrastructure, and for Microsoft to provide another platform on which to extend .NET Framework applications for their developers.

If you are building an application on top of an IaaS vendor such as AWS, GoGrid, or RackSpace, what you really want are application development frameworks that are open, standards-based, and portable. Wolf Frameworks is an example of a PaaS vendor offering a platform on which you can build an SaaS solution that is open and cross-platform. Wolf Frameworks (http://www.wolfframeworks.com/) was founded in Bangalore, India, in 2006, and it has offices in the United States.

Wolf Frameworks is based on the three core Windows SOA standard technologies of cloud computing:

- AJAX, asynchronous Java
- XML
- .NET Framework

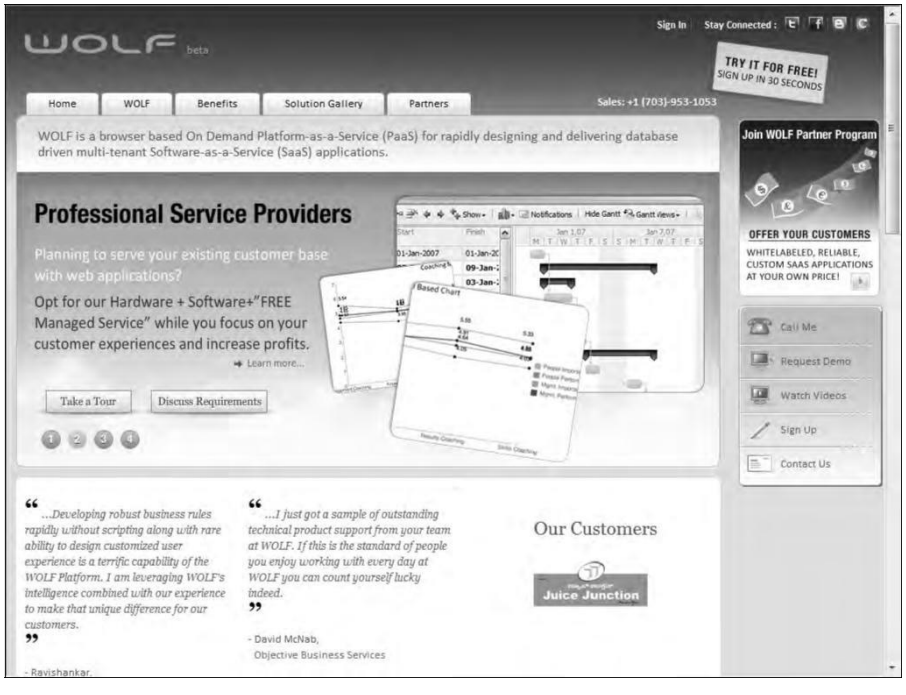
Wolf Frameworks uses a C# engine and supports both Microsoft SQL Server and MySQL database. Applications that you build in Wolf are 100-percent browser-based and support mashable and multisource overlaid content. Figure 7.9 shows the Wolf Frameworks home page.

The Wolf platform is interesting in a number of ways. Wolf has architected its platform so applica-tions can be built without the need to write technical code. It also allows application data to be written to the client's database server of choice, and data can be imported or exported from a vari-ety of data formats. In Wolf, you can view your Business Design of the software application that you build in XML.

Wolf supports forms, search, business logic and rules, charts, reports, dashboards, and both cus-tom and external Web pages. After you create entities and assign their properties, you create busi-ness rules with a rules designer. You can automate tasks via business rules. There are tools for building the various site features such as forms, reports, dashboards, and so on. Connections to the datacenter are over a 128-bit encrypted SSL connection, with authentication, access control, and a transaction history and audit trail. Security to multiple modules can be made available through a Single Sign-On (SSO) mechanism.

FIGURE 7.9

Wolf Frameworks offers an open platform based on SOA standards for building portable SaaS solutions.

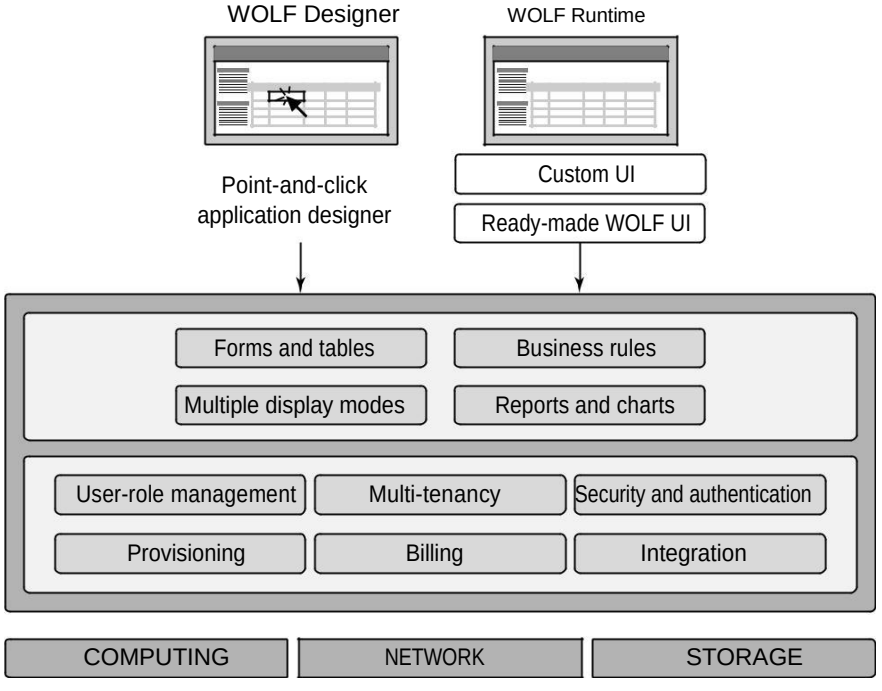


In Wolf, the data and transaction management conforms to the business rules you create. The data and UI rendering are separate systems. Thus, you can change the UI as you need to without affecting your stored data. Wolf lets you work with Adobe Flash or Flex or with Microsoft Silverlight. You can also use third-party on- or off-premises applications with your SaaS applica-tion. A backup system lets you back up data with a single click. Figure 7.10 shows the WOLF platform architecture.

These features enable Wolf developers to create a classic multitenant SOA application without the need for high-level developer skills. These applications are interoperable, portable from one Windows virtual machine to another, and support embedded business applications. You can store your Wolf applications on a private server or in the Wolf cloud.

FIGURE 7.10

The Wolf platform architecture; source: http://www.wolfframeworks.com/platform.asp.



Summary

In this chapter, you learned about one of the core service models in cloud computing: Platform as a Service. With PaaS, the goal is to create hosted scalable applications that are used in a Software as a Service model. For this reason, some vendors start out offering SaaS systems and then broaden them to make them more customizable and programmable as PaaS systems.

Applications built using PaaS tools need to be standards-based. They often are constructed using similar sets of tools: data object and relationship builders; process and business logic systems; forms, views, and reporting tools; and more. This chapter looked at some of the better-known PaaS systems and considered what those tools have in common. You learned about a number of tools in this chapter, including Drupal, Eccentex AppBase, Force.com, LongJump, Squarespace, Wolf, and some others.

Chapter 8 continues the discussion of PaaS by describing one of the largest PaaS systems in use today: Google's AppEngine.

Using Google Web Services

Google is the prototypical cloud computing services company, and it supports some of the largest Web sites and services in the world. In this chapter, you learn about Google's applications and services for users and the various developer tools that Google makes available.

At the center of Google's core business is the company's search technology. Google uses automated technology to index the Web. It makes its search ser-vice available to users as a standard search engine and to developers as a col-lection of special search tools limited to various areas of content. The application of Google's searches to content aggregation has led to enormous societal changes and to a growing trend of disintermediation.

The most important commercial part of Google's activities is its targeting advertising business: AdWords and AdSense. Google has developed a range of services including Google Analytics that supports its targeted advertising business.

Google applications are cloud-based applications. The range of application types offered by Google spans a variety of types: productivity applications, mobile applications, media delivery, social interactions, and many more. The different applications are listed in this chapter. Google has begun to com-mercialize some of these applications as cloud-based enterprise application suites that are being widely adopted.

Google has a very large program for developers that spans its entire range of applications and services. Among the services highlighted are Google's AJAX APIs, the Google Web Toolkit, and in particular Google's relatively new Google Apps Engine hosting service. Using Google App Engine, you can cre-ate Web applications in Java and Python that can be deployed on Google's infrastructure and scaled to a large size.

o

Exploring Google Applications

Few companies have had as much impact on their industries as Google has had on the computer industry and on the Internet in particular. Some companies may have more Internet users (Microsoft comes to mind) or have a stock valuation higher than Google (Apple currently fits that description), but Google remains both a technology and thought leader for all things Internet. For a company whose motto is "Don't be evil," the impact of consumer tracking and targeted advertis-ing, free sourcing applications, and the relentless assault on one knowledge domain after another has had a profound impact on the lives of many people. I call it the Google Effect.

The bulk of Google's income comes from the sales of target advertising based on information that Google gathers from your activities associated with your Google account or through cookies placed on your system using its AdWords system. In 2009, Google's revenue was $23.6 billion, and it controlled roughly 65 percent of the search market through its various sites and services. The com-pany is highly profitable, and that has allowed Google to create a huge infrastructure as well as launch many free cloud-based applications and services that this chapter details. These applica-tions are offered mostly on a free usage model that represents Google's Software as a Service port-folio. A business model that offers cloud-based services for free that are "good enough" is very compelling. While Google is slowly growing a subscription business selling these applications to enterprises, its revenue represents only a small but growing part of Google's current income.

Google's cloud computing services falls under two umbrellas. The first and best-known offerings are an extensive set of very popular applications that Google offers to the general public. These applications include Google Docs, Google Health, Picasa, Google Mail, Google Earth, and many more. You can access a jump table of Google's cloud-based user applications by following the "More" and "Even More" links on Google's home page to the More Google Products page at http://www.google.com/intl/en/options/ shown in Figure 8.1; these features are described in Table 8.1.

Because I cover many of these products in other chapters in this book, the focus in this chapter is to survey the applications that Google offers, to understand why Google offers them as services, and to gain some insight into their potential future role. Google's cloud-based applications have put many other vendors' products—such as office suites, mapping applications, image-manage-ment programs, and many other categories of traditional shrink-wrapped software—under consid-erable pressure.

The second of Google's cloud offerings is its Platform as a Service developer tools. In April 2008, Google introduced a development platform for hosted Web applications using Google's infrastruc-ture called the Google App Engine (GAE). The goal of GAE is to allow developers to create and deploy Web applications without worrying about managing the infrastructure necessary to have their applications run. GAE applications may be written using many high-level programming lan-guages (most prominently Java and Python) and the Google App Engine Framework, which lowers the amount of development effort required to get an application up and running. Goggle also allows a certain free level of service so that the application must exceed a certain level of processor load, storage usage, and network bandwidth (Input/Output) before charges are assessed.

More Google Products equals fewer commercial products.

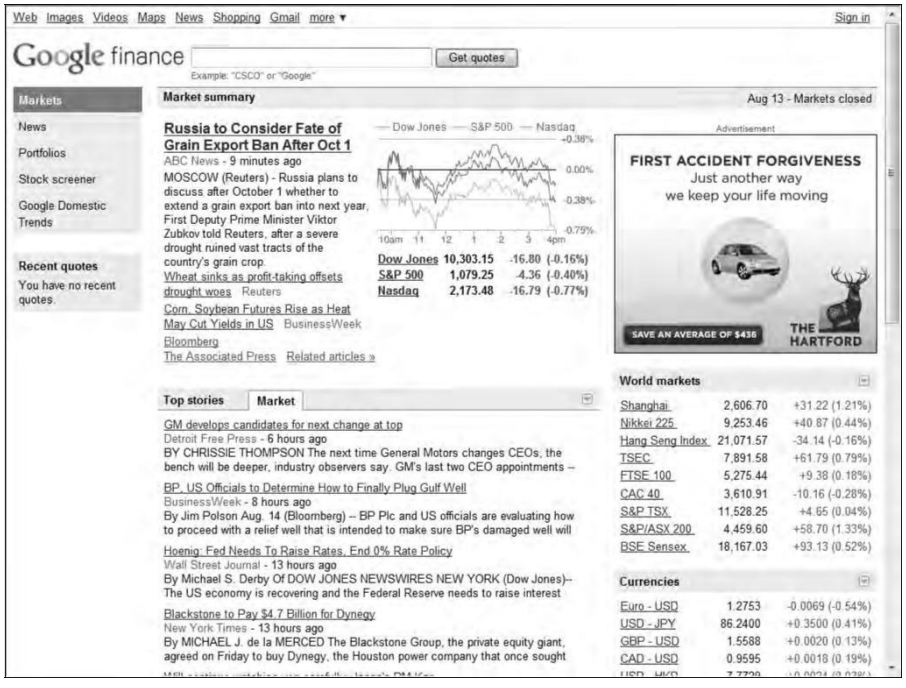
Google App Engine applications must be written to comply with Google's infrastructure. This nar-rows the range of application types that can be run on GAE; it also makes it very hard to port applications to GAE. After an application is deployed on GAE, it is also difficult to port that appli-cation to another platform. Even with all these limitations, the Google App Engine provides devel-opers a low-cost option on which to create an application that can run on a world-class cloud infrastructure—with all the attendant benefits that this type of deployment can bestow.

Surveying the Google Application Portfolio

It is fair to say that nearly all the products in Google's application and service portfolio are cloud computing services in that they all rely on systems staged worldwide on Google's one million plus servers in nearly 30 datacenters. Roughly 17 of the 48 services listed leverage Google's search engine in some specific way. Some of these search-related sites search through selected content such as Books, Images, Scholar, Trends, and more. Other sites such as Blog Search, Finance, News, and some others take the search results and format them into an Aggregation page. Figure 8.2 shows one of these aggregation pages: Google Finance.

FIGURE 8.2

Google's Finance page at http://www.google.com/finance/ is an example of an aggregation page provided by results from Google's search engine.



Indexed search

Google's search technology is based on automated page indexing and information retrieval by Web crawlers, also called spiders or robots. Content on pages is scanned up to a certain number of words and placed into an index. Google also caches copies of certain Web pages and stores copies of documents it finds such as DOC or PDF files in its cache.

Google uses a patented algorithm to determine the importance of a particular page based on the number of quality links to that page from other sites, along with other factors such as the use of keywords, how long the site has been available, and traffic to the site or page. That factor is called the PageRank, and the algorithm used to determine PageRank is a trade secret. Google is always tweaking the algorithm to prevent Search Engine Optimization (SEO) strategies from gaming the system. Based on this algorithm, Google returns what is called a Search Engine Results Page (SERP) for a query that is parsed for its keywords.

It is really important to understand what Google (and other search engines) offers and what it doesn't offer. Google does not search all sites. If a site doesn't register with the search engine or

isn't the target of a prominent link at another site, that site may remain undiscovered. Any site can place directions in their ROBOTS.TXT file indicating whether the site can be searched or not, and if so what pages can be searched. Google developed something called the Sitemaps protocol, which lets a Web site list in an XML file information about how the Google robot can work with the site. Sitemaps can be useful in allowing content that isn't browsable to be crawled; they also can be use-ful as guides to finding media information that isn't normally considered, such as AJAX, Flash, or Silverlight media. The Sitemaps protocol has been widely adopted in the industry.

Note

While dynamic content presented in AJAX isn't normally indexed, Google now has a procedure that helps the Google engine crawl this information. You can read about it at: http://code.google.com/web/ ajaxcrawling/**.** ■

The dark Web

Online content that isn't indexed by search engines belongs to what has come to be called the "Deep Web"—that is, content on the World Wide Web that is hidden. Any site that suppresses Web crawlers from indexing it is part of the Deep Web. You need go no further than the world's number two Web site, Facebook, for a prominent example of a site that isn't indexed in search engines.

Entire networks exist that aren't searchable, particularly peer-to-peer networks. Ian Clarke's Freenet, which is a P2P network, supports both "darknet" and "opennet" connections. Freenet (http://freenetproject.org/) has been downloaded by millions of people.

The Deep Web includes:

- Database generated Web pages or dynamic content
- Pages without links
- Private or limited access Web pages and sites
- Information contained in sources available through executable code such as JavaScript
- Documents and files that aren't in a form that can be searched, which includes not only media files, but information in non-standard file formats

Although efforts are underway to enable information on the Deep Web to be searchable, the amount of information stored that is not accessible is many times larger than the amount of information that can currently be accessed. Some estimates at the size of the Dark Web suggest that it could be an order of magnitude larger than the content contained in the world's search engines.

It is always a good idea to keep these search engine limitations in mind when you work with this technology.

Aggregation and disintermediation

Aggregation pages are a great user service, but they are very controversial—as are a number of Google's search applications and services. It has long been argued that Google's display of informa-tion from various sites violates copyright laws and damages content providers. In several lawsuits, Google successfully defended its right to display capsule information under the Digital Millennium Copyright Act, while in other instances Google responds to requests from interested parties to remove information from its site.

The Authors Guild's filed a class action suit in 2005 regarding unauthorized scanning and copying of books for the creation of the Google Books feature. Google reached a negotiated agreement with the Authors Guild that specified Google's obligations under the fair use exemption. Google argues that the publicity associated with searchable content adds value to that content, and it is clear that this is an argument that will continue into the future.

What is clear is that Google has been a major factor in a trend referred to as disintermediation. Disintermediation is the removal of intermediaries such as a distributor, agent, broker, or some similar functionary from a supply chain. This connects producers directly with consumers, which in many cases is a very good thing. However, disintermediation also has the unfortunate side effect of impacting organizations such as news collection agencies (newspapers, for example), publishers, many different types of retail outlets, and many other businesses, some of which played a positive role in the transactions they were involved in.

Google began to introduce productivity applications starting in 2004 with Gmail. The expansion of these services has continued unabated ever since. Some of these applications are homegrown, but many of them were acquired by acquisition. An example of an acquired product is Writely, the online word processor that is now at the heart of Google Docs and is described in Chapter 14.

Productivity applications and services

These products store your information online in a form that Google can use to build a profile of your activities, and it is unclear how the company uses the information it stores. Google states that your information is never viewed individually by humans, and the company lists its policies in the Privacy Center, which you can find at http://www.google.com/privacypolicy.html. Google has been vigilant in protecting its privacy reputation, but the collection of such a large amount of personal data must give any thoughtful person reason for pause.

Note

Space considerations preclude a more complete description of Google applications and services. Several books treat this topic in detail, including *Google Apps For Dummies* by Ryan Teeter and Karl Barksdale, Wiley, 2008. ■

Table 8.1 lists the current Google "products" listed on its Even More page.

TABLE 8.1

Google Products

Product Name	URL	Google Description
Alerts	http://www.google.com/alerts?hl=en	Sends a periodic e-mail alert to you based on your search term. Search news, blogs, discussions, video, or everything.
Blog Search	http://www.google.com/blogsearch?hl=en	Displays an aggregation page from blogs.
Blogger	http://www.blogger.com/start?hl=en	A blogging site for personal blogs. See Chapter 18 for a description of blogging services.
Books	http://books.google.com/books?hl=en	A vast library of book content in the public domain and previews of copyrighted material.
Calendar	http://www.google.com/calendar/render?hl=en	Calendar service for managing schedules and events and sharing them with others.
Chrome	http://www.google.com/chrome?hl=en&brand=CHMI	Google's browser and operating system wannabe.
Checkout	http://checkout.google.com/	A payment processing system.
Code	http://code.google.com/intl/en/	Developer tools and resources. Described more fully later in this chapter.
Custom Search	http://www.google.com/coop/cse/?hl=en	Creates a custom search utility for a particular Web site.
Desktop	http://desktop.google.com/en/?ignua=1	Indexes content on your local drive for fast searches. Adds a sidebar with gadgets.
Directory	http://www.google.com/dirhp?hl=en	Search the Web by topics, a la Yahoo!
Docs	http://docs.google.com/	Online productivity applications. Described in Chapter 16.
Earth	http://earth.google.com/intl/en/	An online atlas and mapping service with mashups.
Finance	http://www.google.com/finance	A financial news aggregation service and site.
GOOG-411	http://www.google.com/goog-411/	Mobile phone search.
Google Health	http://www.google.com/health/	Health information management system.

(continued)

TABLE 8.1 *(continued)*

Product Name	URL	Google Description
Groups	http://www.google.com/grphp?hl=en	Discussion groups on specific topics.
iGoogle	http://www.google.com/ig?hl=en&source=mpes	AJAX customized home page.
Images	http://images.google.com/imghp?hl=en	Web image search.
Knol	http://knol.google.com/k?hl=en	Short articles submitted by users.
Labs	http://labs.google.com/	A collection of applications and utilities under development and testing.
Orkut	https://www.orkut.com/	Social media service with instant messaging. Described in Chapter 18.
Maps	http://maps.google.com/?hl=en	Mapping and direction service.
Maps for Mobile	http://www.google.com/mobile/default/maps.html	Mapping and direction service. Works with GPS on mobile devices.
Mobile	http://www.google.com/mobile/	Mobile search using voice and location.
News	http://news.google.com/news?ned=en	News aggregation service and Web site.
Pack	http://pack.google.com/?hl=en	Free Windows-based software selected by Google, including Chrome, apps, Desktop, Earth, Picasa, Adobe Reader, Talk, RealPlayer, Skype, and others.
Patent Search	http://www.google.com/patents?hl=en	Patent and trademark search of the United States Patents and Trademark Office.
Picasa	http://picasa.google.com/intl/en/	Photo-editing and management software.
Product Search	http://www.google.com/products	Shopping search function.
Reader	http://www.google.com/reader/view/?hl=en&source=mmm-en	An RSS reader.
Scholar	http://www.google.com/schhp?hl=en	Search site for research and scholarly work from many disciplines.
Search for Mobile	http://www.google.com/mobile/default/search.html	Google's search application optimized for mobile devices.
Sites	http://sites.google.com/	Web site and wiki creation and staging tool.
SketchUp	http://sketchup.google.com/intl/en/	Allows users to create 3D models and share them with others.

Product Name	URL	Google Description
Talk	http://www.google.com/talk/	Instant messaging and chat utility. Can be integrated in Gmail.
Toolbar	http://toolbar.google.com/intl/en/	Provides search features inside different browsers.
Translate	http://translate.google.com/?hl=en	Language translation utility.
Trends	http://www.google.com/trends	Statistical information on different search terms.
Videos	http://video.google.com/?hl=en	Searches for videos on the Web.
Voice	http://voice.google.com/	Free phone service, formerly called Grand Central. Described in Chapter 19.
Web Search	http://www.google.com/webhp?hl=en	Google's core Web search engine of indexed pages sorted with page rank.
Web Search Features	http://www.google.com/intl/en/help/features.html	A help page for special Web searches in Google.
YouTube	http://www.youtube.com/	Flash video sharing site. Described in Chapter 19.

Source: http://www.google.com/intl/en/options/.

Enterprise offerings

As Google has built out its portfolio, it has released special versions of its products for the enter-prise. The following are among Google's products aimed at the enterprise market:

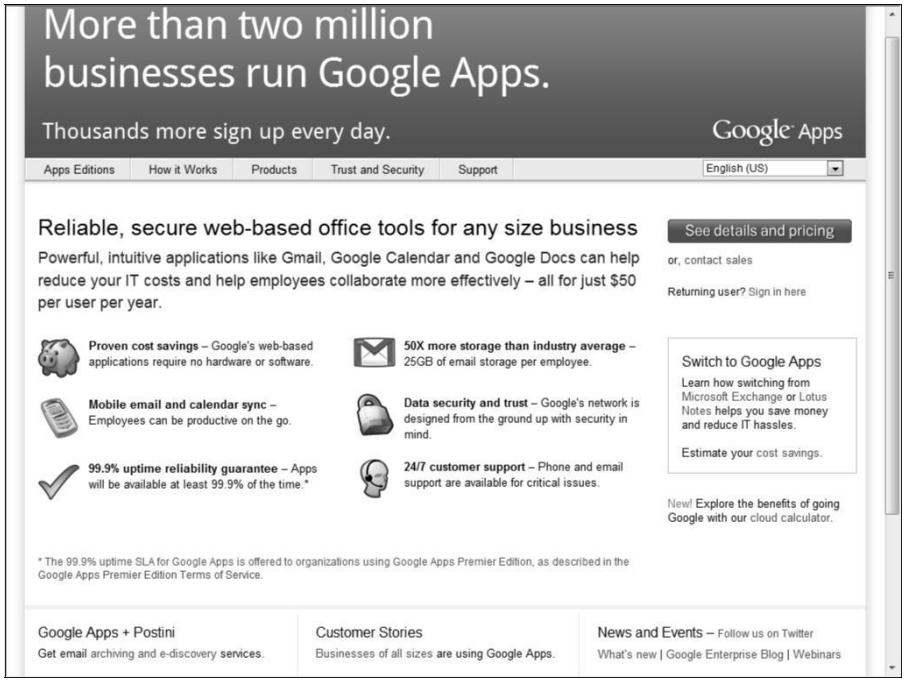
- **Google Commerce Search** (http://www.google.com/commercesearch/): This is a search service for online retailers that markets their products in their site searches with a number of navigation, filtering, promotion, and analytical functions.
- **Google Site Search** (http://www.google.com/sitesearch/): Google sells its search engine customized for enterprises under the Google Site Search service banner. The user enters a search string in the site's search, and Google returns the results from that site.
- **Google Search Appliance** (http://www.google.com/enterprise/gsa): This server can be deployed within an organization to speed up both local (Intranet) and Internet searching. The three versions of the Google Search Appliance can store an index of up to 300,000 (GB-1001), 10 million (GB-5005), or 30 million (GB-8008) documents. Beyond indexing, these appliances have document management features, perform custom searches, cache content, and give local support to Google Analytics and Google Sitemaps.
- **Google Mini** (http://www.google.com/enterprise/mini/): The Mini is the smaller version of the GSA that stores 300,000 indexed documents.

Google also has some success in marketing its productivity applications as office suites to organiza-tions. Google uses different names for the different bundles under a branded program called Google Apps for Business (http://www.google.com/apps/intl/en/business/index. html). Figure 8.3 shows the home page for Google's various office suite bundles. The company has packages for governments, schools, non-profits, and ISPs (a reseller program). Google claims that some 8 million students now use Google Apps, and Google Apps has had some large govern-ment purchases, such as the City of Los Angeles.

For business and other organizations such as governmental agencies, the company has a branded Google Apps Premier Edition, which is a paid service. The different versions offer Gmail, Docs, and Calendar as core applications. The Premier Edition adds 25GB of Gmail storage, e-mail server synchronization, Groups, Sites, Talk, Video, enhanced security, directory services, authentication and authorization services, and the customer's own supported domain—all hosted in the cloud. Premium Edition also adds access to Google APIs and a 24/7 support service with a 99.9-percent uptime guarantee Service Level Agreement. The cost per use is $50 per user account/per year.

FIGURE 8.3

Google Apps for Business is the commercial versions of the company's productivity suites.



To support Google's Premier and Education Editions' Gmail, Google purchased the Postini archiving and discovery service. Google Postini Services (http://www.google.com/ postini/) provides security services such as threat assessment, proactive link blocking and Web policy enforcement, e-mail message encryption, message archiving, and message discovery services. These are paid services that add from $12 to $45 per user/per year, based on the options chosen. Postini allows e-mail to be retained for up to 10 years and can be used to demonstrate regulatory compliance.

Many of Google's productivity applications are quite capable, but none is a state-of-the-art client you might expect to find in a locally installed office suite. When compared one-on-one to Microsoft Office applications, Google's online offerings give users the essential features for a frac-tion of the Microsoft Office price.

Most sophisticated users prefer Microsoft Office, but for the average user (that is most people) Google App bundles are good enough. When that low price is coupled with the collaborative tools and features Google offers, the value of Google Apps will be increasingly more appealing. We can reasonably expect that cloud-based productivity apps will put their shrink-wrapped competitors under great pressure. Microsoft's current strategy of putting crippled Office applications on the Web in Windows Live isn't going to be competitive.

AdWords

AdWords (http://www.google.com/AdWords) is a targeted ad service based on matching advertisers and their keywords to users and their search profiles. This service transformed Google from a competent search engine into an industry giant and is responsible for the majority of Google's revenue stream. AdWords' two largest competitors are Microsoft adcenter (http:// adcenter.microsoft.com/) and Yahoo! Search Marketing (http://searchmarketing. yahoo.com/).

Ads are displayed as text, banners, or media and can be tailored based on geographical location, frequency, IP addresses, and other factors. AdWords ads can appear not only on Google.com, but on AOL search, Ask.com, and Netscape, along with other partners. Other partners belonging to the Google Display Network can also display AdSense ads. In all these cases, the AdWords system determines which ads to match to the user searches.

Here's how the system works: Advertisers bid on keywords that are used to match a user to their product or service. If a user searches for a term such as "develop abdominal muscles," Google returns products based on those terms. You might see an ad with Chuck Norris selling a modern-day version of a torture rack that, if it doesn't give you a six-pack, at least makes your wallet lighter. Up to 12 ads per search can be returned.

Google gets paid for the ad whenever a user clicks it. The system is referred to as pay-per-click advertising, and the success of the ad is measured by what is called the click-through rate (CTR). Google calculates a *quality score* for ads based on the CTR, the strength of the connection between the ad and the keywords, and the advertiser's history with Google. This quality score is a Google trade secret and is used to price the minimum bid of a keyword.

In 2007, Google purchased DoubleClick, an Internet advertising services company. DoubleClick helps clients create ads, provides hosting services, and tracks results for analysis. DoubleClick ads leave browser cookies on systems that collect information from users that determine the number of times a user has been exposed to a particular ad, as well as various system characteristics. Some spyware trackers flag DoubleClick cookies as spyware. Both AdWords and DoubleClick are sold as packages to large clients.

Google Analytics

Google Analytics (GA; http://google.com/analytics) is a statistical tool that measures the number and types of visitors to a Web site and how the Web site is used. It is offered as a free ser-vice and has been adopted by many Web sites. GA is built on the Urchin 5 analytical package that Google acquired in 2006. Figure 8.4 shows the Google Analytics home page.

According to Builtwith.com (http://trends.builtwith.com/analytics/Google-Analytics), Google Analytics was in use on 54 percent of the top 10,000 and 100,000, and 35 percent of the top one million of the world's Web sites. Builtwith.com speculates that Google Analytics JavaScript tag is the most widely used URL in the world today. The service BackendBattles.com (http://www.backendbattles.com/backend/Google_Analytics) sets GA's market share at 57 percent for the top 10,000 sites.

FIGURE 8.4

Google Analytics is the most widely used Web traffic analysis tool on the Internet.

Analytics works by using a JavaScript snippet called the Google Analytics Tracking Code (GATC) on individual pages to implement a *page tag*. When the page loads, the JavaScript runs and creates a first-party browser cookie that can be used to manage return visitors, perform tracking, test browser characteristics, and request tracking code that identifies the location of the visitor. GATC requests and stores information from the user's account. The code stored on the user's system acts like a beacon and collects visitor data that it sends back to GA servers for processing.

Among the visitors that can be tracked are those that land from search engines; referral links in e-mail, documents, and Web pages; display ads; PPC networks; and some other sources. GA aggregates the data and presents the information in a visual form. GA also is connected to the AdWords system so it can track the performance of particular ads in different contexts. You can view referral location statistics and time spent on a page, and you can filter by visitor site. GA lets you save and store up to 50 individual site profiles, provided the site has less than 5 million pageviews per month. This restriction is lifted for an AdWords subscription.

GA cookies are blocked by a number of technologies, such as Firefox Adblock and NoScript or by turning off JavaScript execution in other browsers. You also can delete GA cookies manually or block them, which also defeats the system.

Google Translate

Of all the Google applications, the one that might have significant immediate impact is Google Translate. Computer technology is very close to having the necessary hardware and software to realize the dream of a "universal translator" that the TV show *Star Trek* proposed some 45 years ago. The current version of Google Translate performs machine translation as a cloud service between two of your choice of 35 different languages. That's not truly universal, but until aliens appear, it will do for most people.

Google Translate was introduced in 2007 and replaced the SYSTRAN system that many other computer services utilize. The translation method uses a statistical approach that was first developed by Franz-Joseph Och in 2003. Och now heads the Translate effort at Google.

Translate uses what is referred to as a corpus linguistics approach to translation. You start off building a translation system for a language pair by collecting a database of words and then matching that database to two bilingual text corpuses. A text corpus or parallel collection is a database of word- and phrase-usage taken from the language in everyday use obtained by examining docu-ments translated by professionals to software analysis. Among the documents that are analyzed are the translations of the United Nations and European Parliament, among others.

Google Translate can be accessed directly at http://translate.google.com/ translate_t?hl=en#, where you can select the language pair to be translated. You can do the following:

- Enter text directly into the text box, and click the Translate button to have the text translated.

  If you select the Detect Language option, Translate tries to determine the language automatically and translate it into English.
- Enter a URL for a Web page to have Google display a copy of the translated Web page.

- Enter a phonetic equivalent for script languages.
- Upload a document to the page to have it translated.

Translate parses the document into words and phrases and applies its statistical algorithm to make the translation. As the service ages, the translations are getting more accurate, and the engine is being added to browsers such as Google Chrome and through extension into Mozilla Firefox. The Google Toolbar offers page translation as one of its options, selectable in the Tools settings.

The Google Translator Toolkit (http://translate.google.com/toolkit) shown in Figure 8.5 provides a means for using the Translate to perform translations that you can edit. Shown in the figure is the translation of an article from the English version of Wikipedia into Spanish. The toolkit provides access to tools to aid you in editing the translation.

Translation services have been in development for many years. IBM has had a large effort in this area, and the Microsoft Bing search engine also has a translation engine. There are many other translation engines, and some of them are even cloud-based like Google Translate. What makes Google's efforts potentially unique is the company's work in language transcription—that is, the conversion of voice to text. As part of Google Voice and its work with Android-based cell phones, Google is sampling and converting millions and millions of conversations. Combining these two Web services together could create a translation device based on a cloud service that would have great utility.

FIGURE 8.5

The Google Translator Toolkit lets you translate documents, Web pages, and other material from one language to another and provides tools to improve on the translation.

Exploring the Google Toolkit

Google has an extensive program that supports developers who want to leverage Google's cloud-based applications and services. These APIs reach into every corner of Google's business. Google's Code Home page for developers may be found at http://code.google.com and is shown in Figure 8.6. From this site, you can access developer tools, information on how to use its various APIs to include Google services in your own work, and technical resources.

FIGURE 8.6

Google's Code page at http://code.google.com/intl/en/

Google has a number of areas in which it offers development services, including the following:

- **AJAX APIs** (http://code.google.com/intl/en/apis/ajax/) are used to build widgets and other applets commonly found in places like iGoogle. AJAX provides access to dynamic information using JavaScript and HTML.
- **Android** (http://developer.android.com/index.html) is a phone operating system development.

- **Google App Engine** (http://appengine.google.com/) is Google's Platform as a Service (PaaS) development and deployment system for cloud computing applications.
- **Google Apps Marketplace** (http://code.google.com/intl/en/googleapps/ marketplace/) offers application development tools and a distribution channel for cloud-based applications.
- **Google Gears** (http://gears.google.com/) is a service that provides offline access to online data.

  Google Gears includes a database engine installed on the client that caches data and synchronizes it. Gears allows cloud-based applications to be available to a client even when a network connection to the Internet isn't available. Using Gears, you could work on your mail in Gmail offline, for example.
- **Google Web Toolkit** (GWT; http://code.google.com/webtoolkit) is a set of development tools for browser-based applications.

  GWT is an open-source platform that has been used to create Google Wave and Google AdWords. GWT allows developers to create AJAX applications using Java or with the GWT compiler using JavaScript.
- **Project Hosting** (http://code.google.com/intl/en/projecthosting/) is a project management tool for managing source code.

The Google APIs

Most Google services are exposed by an API, which is why you find a version of Google's search engine, Google Maps, YouTube videos, Google Earth, AdWords, AdSense, and even elements of Google Apps exposed in many other Web sites. You can get to the listing of the Google APIs by clicking the More Products link on the Code page (refer to Figure 8.6). The page you see is http://code.google.com/intl/en/more/, which is shown in Figure 8.7.

Google's APIs can be categorized as belonging to the following categories:

- **Ads and AdSense:** These APIs allow Google's advertising services to be integrated into Web applications. The most commonly used services in this category are AdWords, AdSense, and Google Analytics.
- **AJAX:** The Google AJAX APIs provide a means to add content such as RSS feeds, maps, search boxes, and other information sources by including a snippet of JavaScript into your code.
- **Browser:** Google has several APIs related to building browser-based applications, includ-ing four for the Chrome browser. This category includes the Google Cloud Print API, the Installable Web Apps API for creating installation packages, the Google Web Toolkit for building AJAX applications using Java, and V8, which is a high-performance JavaScript engine.
- **Data:** The Data APIs are those that exchange data with a variety of Google services. The list of Google Data APIs includes Google Apps, Google Analytics, Blogger, Base, Book,

Calendar, Code Search, Google Earth, Google Spreadsheets, Google Notebook, and Picasa Web Albums.

- **Geo:** A number of APIs exist to give location-specific information hooking into maps and geo-specific databases. Some of the more popular APIs in this category include Google Earth, Directions, JavaScripts Maps, Maps API for Flash, and Static Maps.
- **Search:** The search APIs leverage Google's core competency and its central service. APIs such as Google AJAX Search, Book Search, Code Search, Custom Search, and Webmaster Tools Data APIs allow developers to include Google searches in their applications and web sites.
- **Social:** Many Google APIs are used for information exchange and communication tools. They support applications such as Gmail, Calendar, and others, and they provide a set of foundation services. The popular social APIs are Blogger Data, Calendar, Contacts, OpenSocial, Picasa, and YouTube.

Summary

In this chapter, you learned about all things Google. The range of applications and services that Google offers is truly impressive; the company is essentially a self-contained ecosystem. Google's empire is built on its highly regarded search engine. The company monetized search technology by attaching target advertising to searches that its users perform. This revenue has allowed Google to create a range of applications and services on the Web that are having real impact in society.

In this chapter, the applications and services were listed, as were the APIs that are built on these applications and services. Google makes nearly all the products accessible through its APIs. That is why you find Google's services on so many of the world's Web sites.

This chapter ended by describing Google App Engine, a Platform as a Service Web-hosting offering that allows you to create Web applications and deploy them on Google's own infrastructure. Development and deployment of these applications are free, as is some basic usage of the applica-tion. You can scale your applications on a pay-per-use basis to whatever size you need.

In Chapter 9, I examine the approach of Amazon Web Services in cloud computing. AWS offers a very different service model, operating as an Infrastructure as a Service (IaaS) provider.

Using Amazon Web Services

Amazon.com is one of the most important and heavily trafficked Web sites in the world. It provides a vast selection of products using an infrastructure based on Web services. As Amazon.com has grown, it has dramatically grown its infrastructure to accommodate peak traffic times. Over time the company has made its network resources available to partners and affiliates, which also has improved its range of products.

Starting in 2006, Amazon.com made its Web service platform available to developers on a usage-basis model. The technologies described in this chapter represent perhaps the best example of Web services achieved through the Service Oriented Architecture of components that you learn about in Chapter 13. Through hardware virtualization on Xen hypervisors, Amazon.com has made it possible to create private virtual servers that you can run worldwide. These servers can be provisioned with almost any kind of application software you might envisage, and they tap into a range of support services that not only make distributed cloud computing applications possible, but make them robust. Some very large Web sites are running on Amazon.com's infrastructure without their client audience being any the wiser.

Amazon Web Services is based on SOA standards, including HTTP, REST, and SOAP transfer protocols, open source and commercial operating systems, application servers, and browser-based access. Virtual private servers can provision virtual private clouds connected through virtual private networks providing for reasonable security and control by the system administrator.

AWS has a great value proposition: You pay for what you use. While you may not save a great deal of money over time using AWS for enterprise class Web applications, you encounter very little barrier to entry in terms of get-ting your site or application up and running quickly and robustly. AWS has much to teach us about the future of cloud computing and how virtual infra-structure can be best leveraged as a business asset.

Understanding Amazon Web Services

The Amazon is the world's largest river. Amazon.com is the world's largest online retailer with net sales in $24.51 billion, according to their 2009 annual report. The company is a long way past sell-ing books and records. While Amazon.com is not the earth's biggest retailer (that spot is reserved for Wal-Mart), Amazon.com offers the largest number of retail product SKUs through a large eco-system of partnerships. By any measure, Amazon.com is a huge business. To support this business, Amazon.com has built an enormous network of IT systems to support not only average, but peak customer demands. Amazon Web Services (AWS) takes what is essentially unused infrastructure capacity on Amazon.com's network and turns it into a very profitable business. Figure 9.1 shows the Amazon Web Services home page (http://aws.amazon.com/).

AWS is having enormous impact in cloud computing. Indeed, Amazon.com's services represent the largest pure Infrastructure as a Service (IAAS) play in the marketplace today. It is also one of the best examples of what is possible using a Service Oriented Architecture (SOA), which is described in Chapter 13. The structure of Amazon.com's Amazon Web Services (AWS) is therefore highly educational in understanding just how disruptive cloud computing can be to traditional fixed asset IT deployments, how virtualization enables a flexible approach to system rightsizing, and how dispersed systems can impart reliability to mission critical systems.

FIGURE 9.1

Amazon Web Services home page

For these reasons, even though Amazon.com's IaaS services are described in other chapters indi-vidually, this chapter provides background to the entire portfolio and shows why Amazon

Web Services is a $500 million business that hosts eight of the top ten Facebook games (http://gigaom.com/2010/08/02/amazon-web-services-revenues/; and http://venturebeat.com/2010/08/03/amazon-web-services-generating-an-estimated-500m-in-revenue-thanks-in-part-to-growth-of-social-games/). In 2008 AWS claimed 330,000 unique accounts, although the press release for that claim has now disappeared.

Cross-Ref

In Chapter 4, "Understanding Services and Applications by Type," this form of cloud computing is defined as one that provides computer infrastructure usually in the form of a virtualized operating system environment. IaaS is characterized by virtual private servers running operating system and networked application instances, virtual storage, virtual data centers, and networks sold on a per-use or utility basis. ■

Amazon Web Services represents only a small fraction of Amazon's overall business sales at the moment, but it is a rapidly growing component. Amazon doesn't break down its sales by individual areas in its annual report, but according to Randy Bias who blogs on the site Cloudscaling.com (http://cloudscaling.com/blog/cloud-computing/amazons-ec2-generating-220m-annually) the largest component of Amazon's offerings is Amazon's Elastic Compute Cloud (EC2), which generates in excess of $220 million annually as of October 2009. EC2 is esti-mated to run on over 40,000+ servers worldwide divided into six availability zones. (You learn about EC2 later in this chapter.) EC2 is an Infrastructure as a Service (IaaS) play, a market that was pegged to be around $400-$600 M/year and growing 10%-20%/year even in the face of a dramatic market slowdown. Rackspace Cloud (http://www.rackspacecloud.com/), EC2's nearest competitor, is pegged to be around 10% the size of EC2 by Bias.

Amazon Web Service Components and Services

Amazon Web Services is comprised of the following components, listed roughly in their order of importance:

- **Amazon Elastic Compute Cloud** (EC2; http://aws.amazon.com/ec2/), is the central application in the AWS portfolio. It enables the creation, use, and management of virtual private servers running the Linux or Windows operating system over a Xen hyper-visor. Amazon Machine Instances are sized at various levels and rented on a computing/ hour basis. Spread over data centers worldwide, EC2 applications may be created that are highly scalable, redundant, and fault tolerant. EC2 is described more fully the next section. A number of tools are used to support EC2 services:

  Amazon Simple Queue Service (SQS; http://aws.amazon.com/sqs/) is a message queue or transaction system for distributed Internet-based applications. See "Examining the Simple Queue Service (SQS)" later in this chapter for a description of this AWS feature. In a loosely coupled SOA system, a transaction manager is required to ensure that messages are not lost when a component isn't available.

Amazon Simple Notification Service (SNS; http://aws.amazon.com/sns/) is a Web service that can publish messages from an application and deliver them to other applications or to subscribers. SNS provides a method for triggering actions, allowing clients or applications to subscribe to information (like RSS), or polling for new or changed information or perform updates.

EC2 can be monitored by **Amazon CloudWatch** (http://aws.amazon.com/cloudwatch/), which provides a console or command line view of resource utilization, site Key Performance Indexes (performance metrics), and operational indicators for factors such as processor demand, disk utilization, and network I/O. The metrics obtained by CloudWatch may be used to enable a feature called **Auto Scaling** (http://aws.amazon.com/autoscaling/) that can automatically scale an EC2 site based on a set of rules that you create. Autoscaling is part of Amazon Cloudwatch and available at no additional charge.

Amazon Machine Instances (AMIs) in EC2 can be load balanced using the **Elastic Load Balancing** (http://aws.amazon.com/elasticloadbalancing/) fea-ture. The Load Balancing feature can detect when an instance is failing and reroute traffic to a healthy instance, even an instance in other AWS zones. The Amazon CloudWatch metrics request count and request latency that show up in the AWS con-sole are used to support Elastic Load Balancing.

- **Amazon Simple Storage System** (S3; http://aws.amazon.com/s3/**)** is an online backup and storage system, which is described in "Working with Amazon Simple Storage System (S3)" later in this chapter.

  A high speed data transfer feature called AWS Import/Export (http://aws.amazon.com/importexport/) can transfer data to and from AWS using Amazon's own internal network to portable storage devices.

- **Amazon Elastic Block Store** (EBS; http://aws.amazon.com/ebs/**)** is a system for creating virtual disks (volume) or block level storage devices that can be used for Amazon Machine Instances in EC2.

- **Amazon SimpleDB** (http://aws.amazon.com/simpledb/) is a structured data store that supports indexing and data queries to both EC2 and S3. SimpleDB isn't a full database implementation, as you learn in "Exploring SimpleDB (S3)" later in this chapter; it stores data in "buckets" and without requiring the creation of a database schema. This design allows SimpleDB to scale easily. SimpleDB interoperates with both Amazon EC2 and Amazon S3.

- **Amazon Relational Database Service** (RDS; http://aws.amazon.com/rds/) allows you to create instances of the MySQL database to support your Web sites and the many applications that rely on data-driven services. MySQL is the "M" in the ubiquitous LAMP Web services platform (for Linux, APACHE, MySQL, and PERL), and the inclusion of this service allows developers to port applications, their source code, and databases directly over to AWS, preserving their previous investment in these technologies. RDS provides features such as automated software patching, database backups, and automated database scaling via an API call.

- **Amazon Cloudfront** (http://aws.amazon.com/cloudfront/) is an edge-storage or content-delivery system that caches data in different physical locations so that user access to data is enhanced through faster data transfer speeds and lower latency. Cloudfront is similar to systems such as Akamai.com, but is proprietary to Amazon.com and is set up to work with Amazon Simple Storage System (Amazon S3). Cloudfront is currently in beta, but has been well received in the trade press. See "Defining Cloudfront" later in this chapter for more details.

Cross-Ref

The importance of a message queue system for distributed applications is described in Chapter 13, "Understanding Service Oriented Architecture." ■

While the list above represents the most important of the AWS offerings, it is only a partial list—a list that is continually growing and very dynamic. A number of services and utilities support Amazon partners or the AWS infrastructure itself. These are the ones you may encounter:

- **Alexa Web Information Service** (http://aws.amazon.com/awis/) and **Alexa Top Sites** (http://aws.amazon.com/alexatopsites/) are two services that collect and expose information about the structure and traffic patterns of Web sites. This infor-mation can be used to build or structure Web sites, access related sites, analyze historical patterns for growth and relationships, and perform data analysis on site information. Alexa Top Sites can rank sites based on their usage and be used to structure awareness of site popularity into the structure of Web service you build.
- **Amazon Associates Web Services** (A2S) is the machinery for interacting with Amazon's vast product data and eCommerce catalog function. This service, which was called Amazon E-Commerce Service (ECS), is the means for vendors to add their products to the Amazon.com site and take orders and payments.
- **Amazon DevPay** (http://aws.amazon.com/devpay/) is a billing and account man-agement service that can be used by businesses that run applications on top of AWS. DevPay provides a developer API that eliminates the need for application developers to build order pipelines, because Amazon does the billing based on your prices and then uses Amazon Payments to collect the payments.
- **Amazon Elastic MapReduce** (http://aws.amazon.com/elasticmapreduce/) is an interactive data analysis tool for performing indexing, data mining, file analysis, log file analysis, machine learning, financial analysis, and scientific and bioinformatics research. Elastic MapReduce is built on top of a Hadoop framework using the Elastic Compute Cloud (EC2) and Simple Storage Service (S3).
- **Amazon Mechanical Turk** (http://aws.amazon.com/mturk/) is a means for accessing human researchers or consultants to help solve problems on a contractual or temporary basis. Problems solved by this human workforce have included object identifi-cation, video or audio recording, data duplication, and data research. Amazon.com calls this type of work Human Intelligence Tasks (HITs). The Mechanical Turk is currently in beta.

- **AWS Multi-Factor Authentication** (AWS MFA; http://aws.amazon.com/mfa/**)** is a special feature that uses an authentication device you have in your possession to provide access to your AWS account settings. This hardware key generates a pseudo-random six-digit number when you press a button that you enter into your logon. This gives you two layers of protection: your user id and password (things you know) and the code from your hardware key (something you have). This multifactor security feature can be extended to Cloudfront and Amazon S3. The Enzio Time Token from Gemalto (http://online noram.gemalto.com/) is available for use with Amazon Web Service; the key costs $12.99.

  Secure access to your EC2 AMIs is controlled by passwords, Kerberos, and 509 Certificates.

- **Amazon Flexible Payments Service** (FPS; http://aws.amazon.com/fps/) is a pay-ments-transfer infrastructure that provides access for developers to charge Amazon's cus-tomers for their purchases. Using FPS, goods, services, donations, money transfers, and recurring payments can be fulfilled. FPS is exposed as an API that sorts transactions into packages called Quick Starts that make this service easy to implement.

- **Amazon Fulfillment Web Services** (FWS; http://aws.amazon.com/fws/) allows merchants to fill orders through Amazon.com fulfillment service, with Amazon handling the physical delivery of items on the merchant's behalf. Merchant inventory is preposi-tioned in Amazon's fulfillment centers, and Amazon packs and ships the items. There is no charge for using Amazon FWS; fees for the Fulfillment by Amazon (FBA; http:// www.amazon.com/gp/seller/fba/fulfillment-by-amazon.html) service apply. Between FBA and FWS, you can create a nearly virtual store on Amazon.com.

- **Amazon Virtual Private Cloud** (VPC; http://aws.amazon.com/vpc/) provides a bridge between a company's existing network and the AWS cloud. VPC connects your network resources to a set of AWS systems over a Virtual Private Network (VPN) connec-tion and extends security systems, firewalls, and management systems to include their provisioned AWS servers. Amazon VPC is integrated with Amazon EC2, but Amazon plans to extend the capabilities of VPC to integrate with other systems in the Amazon cloud computing portfolio.

- **AWS Premium Support** (http://aws.amazon.com/premiumsupport/) is Amazon's technical support and consulting business. Through AWS Premium Support, subscribers to AWS can get help building or supporting applications that use EC2, S3, Cloudfront, VPC, SQS, SNS, SimpleDB, RDS, and the other services listed above. Service plans are available on a per-incidence, monthly, or unlimited basis at different levels of service.

With this overview of AWS components complete, let's look at the central part of Amazon Web Service's value proposition, the creation and deployment of virtual private servers using the Elastic Compute Cloud (EC2) service.

Working with the Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (EC2) is a virtual server platform that allows users to create and run virtual machines on Amazon's server farm. With EC2, you can launch and run server instances called Amazon Machine Images (AMIs) running different operating systems such as Red Hat Linux and Windows on servers that have different performance profiles. You can add or subtract virtual servers elastically as needed; cluster, replicate, and load balance servers; and locate your different servers in different data centers or "zones" throughout the world to provide fault tolerance. The term *elastic* refers to the ability to size your capacity quickly as needed.

The difference between an instance and a machine image is that an instance is the emulation of a hardware platform such as X86, IA64, and so on running on the Xen hypervisor. A machine image is the software and operating system running on top of the instance. A machine image may be thought of as the contents of a boot drive, something that you could package up with a program such as Ghost, Acronis, or TrueImage to create a single file containing the exact contents of a volume. A machine image should be composed of a hardened operating system with as few features and capabilities as possible and locked down as much as possible.

Consider a situation where you want to create an Internet platform that provides the following:

- A high transaction level for a Web application
- A system that optimizes performance between servers in your system
- Data driver information services
- Network security
- The ability to grow your service on demand

Implementing that type of service might require a rack of components that included the following:

- An application server with access to a large RAM allocation
- A load balancer, usually in the form of a hardware appliance such as F5's BIG-IP
- A database server
- Firewalls and network switches
- Additional rack capacity at the ISP

A physical implementation of these components might cost you something in the neighborhood of $25,000 depending upon the scale of your application. With AWS, you might be able to have an equivalent service for as little as $1,000 and have a high level of availability and reliability to boot. This difference may surprise you, but it is understandable when you consider that AWS can run its services with a much greater efficiency than your company would alone and therefore amortize its investment in hardware over several customers. That is the promise and the potential of cloud computing realized and why large Web sites such as Recovery.gov have moved to AWS.

Amazon Machine Images

AMIs are operating systems running on the Xen virtualization hypervisor. Each virtual private server is accorded a size rating called its *EC2 Compute Unit*, which is pegged to the equivalent of a 1.0–1.2 GHz 2007 Opteron or 2007 Xeon processor. Table 9.1 shows the current set of Instance types, which broadly fall into the following three classes:

1. **Standard Instances:** The standard instances are deemed to be suitable for standard server applications.
2. **High Memory Instances:** High memory instances are useful for large data throughput applications such as SQL Server databases and data caching and retrieval.
3. **High CPU Instances:** The high CPU instance category is best used for applications that are processor- or compute-intensive. Applications of this type include rendering, encod-ing, data analysis, and others.

TABLE 9.1

Amazon Machine Image Instance Types

Type	Compute Engine	RAM (GB)	Storage (GB)1	Platform	I/O Performance	API Name
Micro instance	Up to 2 EC2 Compute Units (1 virtual core) in short bursts	0.613	EBS (Elastic Block Storage) storage only	32-bit or 64-bit	Low	T1.micro
Standard instance – small (default)	1 EC2 Compute Unit (1 virtual core)	1.7	160	32-bit	Moderate	m1.small
Standard instance – large	4 EC2 Compute Units (2 virtual cores X 2 EC2 Units)	7.5	850	64-bit	High	m1.large
Standard instance – extra large	8 EC2 Compute Units (4 virtual cores X 2 EC2 Units)	15	1,690	64-bit	High	m1.xlarge
High Memory Double Extra Large Instance	13 EC2 Compute Units (4 virtual cores X 3.25 EC2 Units)	34.2	850	64-bit	High	m2.2xlarge

Type	Compute Engine	RAM (GB)	Storage (GB)1	Platform	I/O Performance	API Name

High Memory Quadruple Extra Large Instance	26 EC2 Compute Units (8 virtual cores X 3.25 EC2 Units)	68.4	1,690	64-bit	High	m2.4xlarge
High CPU Medium Instance	5 EC2 Compute Units (2 virtual cores X 2.5 EC2 Units)	1.7	350	32-bit	Moderate	c1.medium
High CPU Extra Large Instance	20 EC2 Compute Units (8 virtual cores X 2.5 EC2 Units)	7	1,690	64-bit	High	c1.xlarge

1. Storage is not persistent. All assigned storage is lost upon rebooting. To store data on AWS, you need to create a Simple Storage Service (S3) bucket or an Elastic Block Storage (EBS) volume.

Pricing models

The pricing of these different AMI types depends on the operating system used, which data center the AMI is located in (you can select its location), and the amount of time that the AMI runs. Rates are quoted based on an hourly rate. Additional charges are applied for:

- the amount of data transferred
- whether Elastic IP Addresses are assigned
- your virtual private server's use of Amazon Elastic Block Storage (EBS)
- whether you use Elastic Load Balancing for two or more servers
- other features

AMIs that have been saved and shut down incur a small one-time fee, but do not incur additional hourly fees.

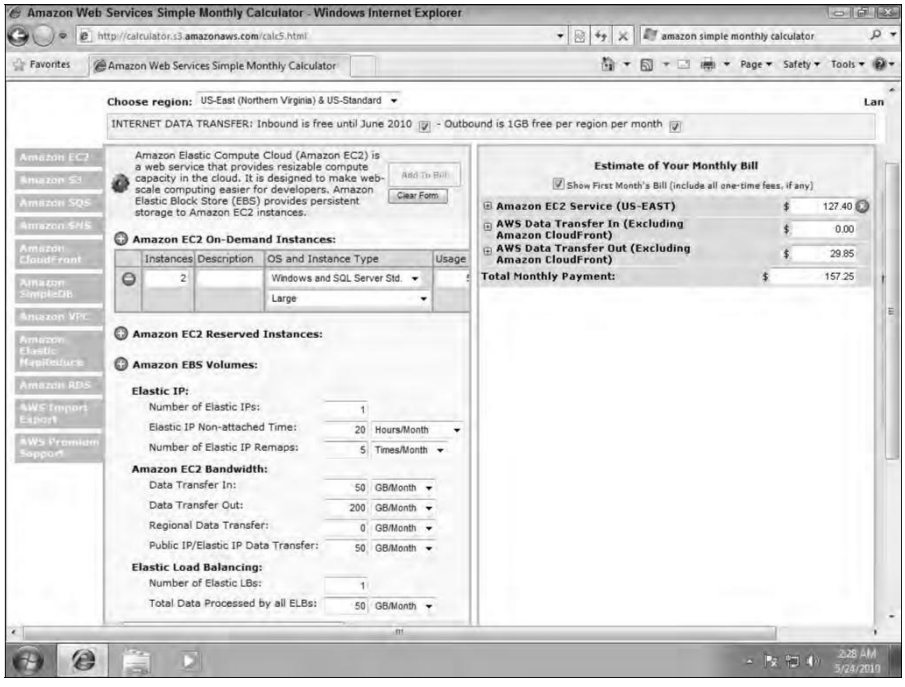
The three different pricing models for EC2 AMIs are as follows:

- **On-Demand Instance:** This is the hourly rate with no long-term commitment.
- **Reserved Instances:** This is a purchase of a contract for each instance you use with a sig-nificantly lower hourly usage charge after you have paid for the reservation.
- **Spot Instance:** This is a method for bidding on unused EC2 capacity based on the current spot price. This feature offers a significantly lower price, but it varies over time or may not be available when there is no excess capacity.

Pricing varies by zone, instance, and pricing model. A chart of the different current prices may be found at http://aws.amazon.com/ec2/. This page also includes current Amazon Elastic Block Store volume and snapshot charges to Amazon S3, as well as data transfer rates. Figure 9.2 shows the AWS Simple Monthly Calculator that you can find at http://calculator.s3.amazonaws.com/calc5.html to help you estimate your monthly charges.

FIGURE 9.2

The Amazon Web Services Simple Monthly Calculator for determining system costs on AWS



System images and software

You can choose to use a template AMI system image with the operating system of your choice or create your own system image that contains your custom applications, code libraries, settings, and data. Security can be set through passwords, Kerberos tickets, or certificates.

These operating systems are offered:

- Red Hat Enterprise Linux
- OpenSuse Linux
- Ubuntu Linux

- Sun OpenSolaris
- Fedora
- Gentoo Linux
- Oracle Enterprise Linux
- Windows Server 2003/2008 32-bit and 64-bit up to Data Center Edition
- Debian

Most of the system image templates that Amazon AWS offers are based on Red Hat Linux, Windows Server, Oracle Enterprise Linux, and OpenSolaris from the list above. Table 9.2 lists some of the more common enterprise applications that are available from AWS either as part of its canned templates or for use in building your own AMI system image. Hundreds of free and paid AMIs can be found on AWS.

TABLE 9.2

EC2 Enterprise Software Types

Application Type	Software
Application Development Environments	IBM sMash, JBoss Enterprise Application Platform, and Ruby on Rails
Application Servers	IBM WebSphere Application Server, Java Application Server, and Oracle WebLogic Server
Batch Processing	Condor, Hadoop, and Open MPI
Databases	IBM DB2, IBM Informix Dynamic Server, Microsoft SQL Server Standard 2005, MySQL Enterprise, and Oracle Database 11g
Video Encoding and Streaming	Windows Media Server and Wowza Media Server Pro
Web Hosting	Apache HTTP, IIS/ASP.Net, IBM Lotus Web Content Management, and IBM WebSphere Portal Server

When you create a virtual private server, you can use the Elastic IP Address feature to create what amounts to a static IP v4 address to your server. This address can be mapped to any of your AMIs and is associated with your AWS account. You retain this IP address until you specifically release it from your AWS account. Should a machine instance fail, you can map your Elastic IP Address to fail over to a different AMI. You don't need to wait until a DNS server updates the IP record assign-ment, and you can use a form to configure the reverse DNS record of the Elastic IP address change.

There are currently four different EC2 service zones or regions:

- US East (Northern Virginia)
- US West (Northern California)
- EU (Ireland)
- Asia Pacific (Singapore)

Working with Amazon Storage Systems

When you create an Amazon Machine Instance you provision it with a certain amount of storage. That storage is temporal, it only exists for as long as your instance is running. All of the data contained in that storage is lost when the instance is suspended or terminated, as the storage is reassigned to the pool for other AWS users to use. For this and other reasons you need to have access to persistent storage. The Amazon Simple Storage System provides block storage, but is set up in a way that is somewhat unique from other storage systems you may have worked with in the past.

Amazon Simple Storage System (S3)

Amazon S3's cloud-based storage system allows you to store data objects ranging in size from 1 byte up to 5GB in a flat namespace. In S3, storage containers are referred to as buckets, and buckets serve the function of a directory, although there is no object hierarchy to a bucket, and you save objects and not files to it. It is important that you do not associate the concept of a filesystem with S3, because files are not supported; only objects are stored. Additionally, you do not "mount" a bucket as you do a filesystem.

The S3 system allows you to assign a name to a bucket, but that name must be unique in the S3 namespace across all AWS customers. Access to an S3 bucket is through the S3 Web API (either with SOAP or REST) and is slow relative to a real-world disk storage system. S3's performance limits its use to non-operational functions such as data archiving and retrieval or disk backup. The REST API is preferred to the SOAP API, because it is easier to work with large binary objects with REST.

You can do the following with S3 buckets through the APIs:

- Create, edit, or delete existing buckets
- Upload new objects to a bucket and download them
- Search for and find objects and buckets
- Find metadata associate with objects and buckets
- Specify where a bucket should be stored
- Make buckets and objects available for public access

One tool commonly used to manage data for Amazon S3 is the s3cmd command line client (http://s3tools.org/s3cmd).

The S3 service is used by many people as the third level backup component in a 3-2-1 backup strategy. That is, you have your original data (1), a copy of your data (2), and an off-site copy of

---

your data (3); the latter of these may be S3. In this regard, S3 acts as a direct competitor to Carbonite's backup system. One of the options available to you is versioning for Amazon S3. With versioning, every version of an object stored in an S3 bucket is retained, provided you enable the versioning feature. Any HTTP or REST operation such as PUT, POST, COPY, or DELETE creates a new object that is stored along with the older version. A GET operation retrieves the newest ver-sion of the object, but the ability to recover and undo actions is available. Versioning also can be used for preserving data and for archiving purposes.

Amazon S3 provides large quantities of reliable storage that is highly protected but to which you have low bandwidth access. S3 excels in applications where storage is archival in nature. For example, you find S3 in use by large photo sharing sites. In the next section you'll see Amazon's

Elastic Block Storage or EBS. In EBS you create virtual drives that you can use with your machine instances in the same way that you would use a hard drive with a physical system. EBS tends to be used in transactional systems where high-speed data access is required.

Caution

Keep in mind that while Amazon S3 is highly reliable, it is not highly available. You can definitely get your data back from S3 at some point with guaranteed 100% fidelity, but the service is not always connected and experi-ences service outages. By comparison, an EBS volume is offered with an annual failure rate of 0.1% to 0.5%, about a factor of 10 better than typical disk drives you use in your own physical servers. ■

Amazon Elastic Block Store (EBS)

The third of Amazon's data storage systems are devoted to Amazon Elastic Block Storage (EBS), which is a persistent storage service with a high operational performance. Advantages of EBS are that it can store file system information and its performance is higher and much more reliable than Amazon S3. That makes EBS valuable as an operational data storage medium for AWS. The cost of creating an EBS volume is also greater than creating a similarly sized S3 bucket.

An EBS volume can be used as an instance boot partition. The advantages of an EBS boot partition are that you can have a volume up to 1TB, retain your boot partition separately from your EC2 instance, and use a boot partition volume as a means for bundling an AMI into a single package. EBS boot partitions can be stopped and started, and they offer fast AMI boot times.

EBS is similar in concept to a Storage Area Network or SAN; you create block storage volumes varying in size from 1GB to 1TB and make those volumes available to your machine instances. The performance of a volume is dependent upon the network I/O and therefore varies as a function of the size of your instance (see Table 9.3), as well as the type of disk I/O operations (random, sequential, request size, and READS or WRITE) that are in progress.

When you create volumes, they appear first as raw block storage devices that must be formatted for use. A volume is mounted on a particular instance and is available to that instance alone; that is, volumes may not be shared between instances. Volumes may be located in the same zone as the AMI to which they are attached. Volumes appear as if they are devices (physical drives) when attached to an instance. You can mount multiple volumes on a single instance, if desired, and cre-ate striped RAID volumes for faster performance. The filesystem for mounted volumes appears when you open the volume, and you can install applications or copy data to mounted volumes as you would any physical disk.

EBS supports volume replication within the same availability zone, which can add an extra level of fault tolerance to the data set. The use of replication means that mirroring a volume won't add much additional fault tolerance. Snapshots are the recommended approach to improving your volume's reliability.

You can make an instance image or snapshot of your AMI, and these point-in-time snapshots are then copied out to Amazon S3. You can use these snapshots as system images to create new AMIs or to restore a volume (and instance) to that point-in-time snapshot when needed. You can share snapshots with other authorized users by using a volume's context menu in the AWS Management Console and selecting the Snapshot Permissions command.

When you create a new volume from an S3 snapshot, the data is slowly copied to the new volume. As you start working on the new volume, any missing data is downloaded preferentially as needed.

Each snapshot you take adds incremental changes to the previous snapshot, which means that while the first snapshot takes a fair amount of time, subsequent snapshots are usually executed quickly and with only a modest amount of extra storage space required.

Tip

EBS supports a special feature of AWS called a Public Data Set, which is a data repository that is made avail-able at no extra charge to AWS customers; only the data transfer and compute fees are paid for working with the data set. Examples of Public Data Sets in use are the Annotated Human Genome map, U.S. Census Databases (1980, 1990, and 2000), UniGene transcript sequences, and Freebase.com data dump, among oth-ers. To learn more about Public Data Sets, see http://aws.amazon.com/publicdatasets/**.** ■

EBS is a service priced on the amount of storage space used, how long you use it, and the number of I/O requests made to the volume. You can use a utility like IOSTAT to measure I/O of your systems to estimate these transaction costs, which vary greatly by operating system and application. Amazon quotes an example of a medium-sized database of 100GB with 100 I/O per sec costing about $10 per month for the allocated storage and $26 per month for the I/O (there are 2.6 million seconds in a month). Snapshots are priced on the storage blocks used, not on the size of the volume being stored. Amazon also charges for the amount of data transferred to Amazon S3 during a snapshot.

Table 9.3 summarizes the various properties of the three different forms of EC2 data storage devices.

TABLE 9.3

EC2 Storage Type Properties

Property	AMI Instance	Amazon Simple Storage Service (S3)	Amazon Elastic Block Storage (EBS)	Amazon CloudFront
Adaptability	Medium	Low	High	Medium
Best usage	Transient data storage	Persistent or archival storage	Operational data storage	Data sharing and large data object streaming

Property	AMI Instance	Amazon Simple Storage Service (S3)	Amazon Elastic Block Storage (EBS)	Amazon CloudFront
Cost	Low	Medium	High	Low
Ease of use	Low	High	High	High
Data protection	Very Low	Very High	High	Low
Latency	Medium	Low	High	High
Least best used as	Persistent storage	Operational storage	For small I/O transfers	Operational data
Reliability	High	Medium	High	Medium
Throughput	Variable	Slow	High	High

CloudFront

Amazon CloudFront is referred to as a content delivery network (CDN), and sometimes called *edge computing.* In edge computing, content is pushed out geographically so the data is more readily available to network clients and has a lower latency when requested. You enable CloudFront through a selection in the AWS Management Console.

You can think of a CDN as a distributed caching system. CloudFront servers are located throughout the world—in Europe, Asia, and the United States. As such, CloudFront represents yet another level of Amazon cloud storage. A user requesting data from a CloudFront site is referred to the nearest geographical location. CloudFront supports "geo-caching" data by performing static data transfers and streaming content from one CloudFront location to another.

At the time this chapter was written CloudFront was in beta, but it has been well received. Direct competitors for CloudFront include Akamai Technologies (http://www.akamai.com/), Edgecast Networks (http://www.edgecast.com/), and Limelight Networks (http://www.limelightnetworks.com/). CloudFront's aggressive pricing model is expected to put pressure on these other services over time. Pricing for CloudFront is based on how much data is transferred to clients, and it doesn't require a service contract. You can estimate CloudFront's costs using the AWS Simple Monthly Calculator (refer to Figure 9.3); costs vary by region.

When you create a CloudFront implementation, a CloudFront domain name is registered for your domain name in the form <domainname>.cloudfront.net, and objects in the CloudFront domain can be mapped to your own domain. You store your source files on CloudNet servers in Amazon S3 buckets and then use the CloudFront API to register the S3 bucket with the CloudNet distribution. Then in your applications, Web pages, and links, you reference the distribution location.

CloudFront represents the last of the Amazon Web Services that store and serve objects and files. To store data in a way that makes it searchable and organizes it, Amazon offers two different data-base services that are covered in the next section.

Understanding Amazon Database Services

Amazon offers two different types of database services: Amazon SimpleDB, which is non-relational, and Amazon Relational Database Service (Amazon RDS), both of which were in beta at the time of this writing. Dynamic data access is a central element of Web services, particularly "Web 2.0" ser-vices, so although AMIs support several of the major databases, it isn't surprising that they would create their own databases as part of the AWS Service Oriented Architecture.

Amazon SimpleDB

Amazon SimpleDB is an attempt to create a high performance data store with many database features but without the overhead. This is analogous to the goals used to create the Amazon Simple Storage System (S3). The service is meant to be low touch, in that it abstracts many of the common concerns of database administrators for hardware requirements, software maintenance, indexing, and performance optimization.

To create a high performance "simple" database, the data store created is flat; that is, it is non-relational and joins are not supported. Data stored in SimpleDB domains doesn't require maintenances of a schema and is therefore easily scalable and highly available because replication is built into the system. Data is stored as collections of items with attribute-value pairs, and the system is akin to using the database function within a spreadsheet. To support replication, a set of two consistency functions are part of SimpleDB that check data across the different copies. Transactions are per-formed as a set of conditional PUTS and DELETES, and you can INSERT, REPLACE, or DELETE values for item attributes. These transaction capabilities do not enable features like ROLLBACK, but they allow you to create solutions that maintain optimistic concurrency control and will perform an INSERT based on the value of a counter or timestamp.

You grow a SimpleDB database by scaling out and creating additional data domains, and SimpleDB integrates with EC2 instances and S3 storage. Data objects stored in S3 can be queried in SimpleDB, returning information about the objects' metadata and pointers to the objects' location.

Data in SimpleDB is automatically indexed and may be queried as needed. The API is relatively simple, consisting of domain creation, put and get attributes, and SELECT statements. According to Amazon, query performance is near the level you would see for a database on a LAN, as access through a browser. Although a SimpleDB database is replicated and therefore made highly avail-able and fault tolerant, the service lacks many of the speed enhancements available to relational systems. A data domain may be located geographically in any of AWS's regions.

The design goal was to remove as much of the database system maintenance as possible. In a Web services architecture, many applications don't require the performance level of a relational data-base. Among the featured uses of SimpleDB are data logging, online gaming, and metadata index-ing. SimpleDB would not be the best choice for a high-volume transaction system. Data transfers within regions between SimpleDB and other AWS services are free. Service charges accrue based on SimpleDB Machine Hours and inter-regional data transfers.

The three areas of use for SimpleDB that Amazon Web Services highlights are: logging (http://aws.amazon.com/simpledb/usecases_logging/), online gaming (http://aws.amazon.com/simpledb/usecases_online_gaming/), and metadata indexing (http://aws.amazon.com/simpledb/usecases_metadata_indexing/).

Amazon Relational Database Service (RDS)

Amazon Relational Database Service is a variant of the MySQL5.1 database system, but one that is somewhat simplified. The purpose of RDS is to allow database applications that already exist to be ported to RDS and placed in an environment that is relatively automated and easy to use. RDS automatically performs functions such as backups and is deployable throughout AWS zones using the AWS infrastructure.

In RDS, you start by launching a database instance in the AWS Management Console and assigning the DB Instance class and size of the data store. The DB Instance is then connected to your MySQL database. Any database tool that works with MySQL 5.1 will work with RDS. Additionally, you can monitor your database usage as part of Amazon CloudWatch. Table 9.4 shows the different Instance Classes for an Amazon RDS database. Pricing is based on machine hour rates by class, by amount of storage per month, and per million requests.

TABLE 9.4

Amazon Relational Database Service Instance Class

Type1	Compute Engine	RAM (GB)	Platform	Price2
Small DB Instance (default)	1 EC2 Compute Unit (1 virtual core)	1.7	64-bit	$0.11
Large DB Instance	2 EC2 Compute Units (2 virtual cores X 2 EC2 Units)	7.5	64-bit	$0.44
Extra Large DB Instance	8 EC2 Compute Units (4 virtual cores X 2 EC2 Units)	15	64-bit	$0.88
Double Extra Large DB Instance	13 EC2 Compute Units (4 virtual cores X 3.25 EC2 Units)	34	64-bit	$1.55
Quadruple Extra Large DB Instance	26 EC2 Compute Units (8 virtual cores X 3.25 EC2 Units)	68	64-bit	$3.10

1. Storage available is from 5GB to 1TB.

2. Price for U.S. N. Virginia deployment for database machine; storage price is $0.10 per GB-month; and I/O rate price is $0.10 per 1 million requests for the same location. Data transfer rates also apply.

Among the important features of RDS is the automated point-in-time backup system for data in the database as well as for the MySQL transaction logs. Backups can be saved for up to eight days. In addition to backup, RDS supports database snapshots. A DB Snapshot is stored as a full database backup and is retained until you specifically delete it from your storage container. Snapshots may be scheduled or may be manually initiated by an administrator.

The deployment of RDS databases can be spread among multiple availability zones for increased fault tolerance and data availability. These so-called "Multi-AZ Deployments" can be automatically replicated and maintain a standby replica in another availability zone, with automatic failover when a database disruption is detected. The conversion of a single location RDS database to a Multi-DB deployment may be accomplished with a single API call. Other API calls support instance creation and maintenance, snapshots, and restores.

Choosing a database for AWS

In choosing a database solution for your AWS solutions, consider the following factors in making your selection:

- Choose SimpleDB when index and query functions do not require relational database support.
- Use SimpleDB for the lowest administrative overhead.
- Select SimpleDB if you want a solution that autoscales on demand.
- Choose SimpleDB for a solution that has a very high availability.
- Use RDS when you have an existing MySQL database that could be ported and you want to minimize the amount of infrastructure and administrative management required.
- Use RDS when your database queries require relation between data objects.
- Chose RDS when you want a database that scales based on an API call and has a pay-as-you-use-it pricing model.
- Select Amazon EC2/Relational Database AMI when you want access to an enterprise rela-tional database or have an existing investment in that particular application.
- Use Amazon EC2/Relational Database AMI to retain complete administrative control over your database server.

Summary

In this chapter, Amazon Web Services (AWS) were described. AWS is the most successful example of a Service Oriented Architecture (SOA) that provides an Infrastructure as a Service (IaaS) cloud computing solution. With AWS, you can create virtual private servers using the Elastic Cloud Compute (EC2) service, dynamically size your servers and distribute them throughout the world, and create an application infrastructure that allows for very sophisticated and scalable applications.

The process for creating an Amazon Machine Instance (AMI) was described, as was the provision-ing of various resources such as storage and databases for those instances. The range of services that AWS supports is wide and includes Content Delivery Networks (CDNs), messaging and notifi-cation systems, load balancing and replication, and many other services as well. AWS is among the most important developer platforms for building cloud computing applications, and the range of services and their point of development was described here.

Chapter 10, "Using Microsoft Web Services," describes the range of developer and user tools based in large part on Microsoft proprietary technologies such as .NET Framework, ASP.NET, and the Azure platform. Microsoft's cloud computing effort is considerable and falls midway between the approach taken by Google, which delivers applications that impact users, and Amazon.com, which delivers services that provide cloud computing infrastructure. In short, Amazon Web Services pro-vide something for everyone.

Using Microsoft Cloud Services

Microsoft has a very extensive cloud computing portfolio under active development. Efforts to extend Microsoft products and third-party applications into the cloud are centered around adding more capabilities to existing Microsoft tools. Microsoft's approach is to view cloud applications as software plus service. In this model, the cloud is another platform and applications can run locally and access cloud services or run entirely in the cloud and be accessed by browsers using standard Service Oriented Architecture (SOA) protocols.

Microsoft calls their cloud operating system the Windows Azure Platform. You can think of Azure as a combination of virtualized infrastructure to which the .NET Framework has been added as a set of .NET Services. The Windows Azure service itself is a hosted environment of virtual machines enabled by a fabric called Windows Azure AppFabric. You can host your application on Azure and provision it with storage, growing it as you need it. Windows Azure service is an Infrastructure as a Service offering.

A number of services interoperate with Windows Azure, including SQL Azure (a version of SQL Server), SharePoint Services, Azure Dynamic CRM, and many of Windows Live Services comprising what is the Windows Azure Platform, which is a Platform as a Service cloud computing model. Eventually, many more services will be added, encompassing the whole range of Microsoft's offerings. This architecture positions Microsoft to either extend its product into the Web or to license its products, whichever way the cloud computing marketplace develops. From Microsoft's position and that of its developers, Windows Azure has lots of advantages.

Windows Live Services is a collection of applications and services that run on the Web. Some of these applications

---

Live Services are standalone Web applications viewable in a browser. An important subset of these Windows Live Services is available to Windows Azure applications through the Windows Live Messenger Connect API. A set of Windows Live for Mobile applications also exists. These applica-tions and services are more fully described in this chapter.

Exploring Microsoft Cloud Services

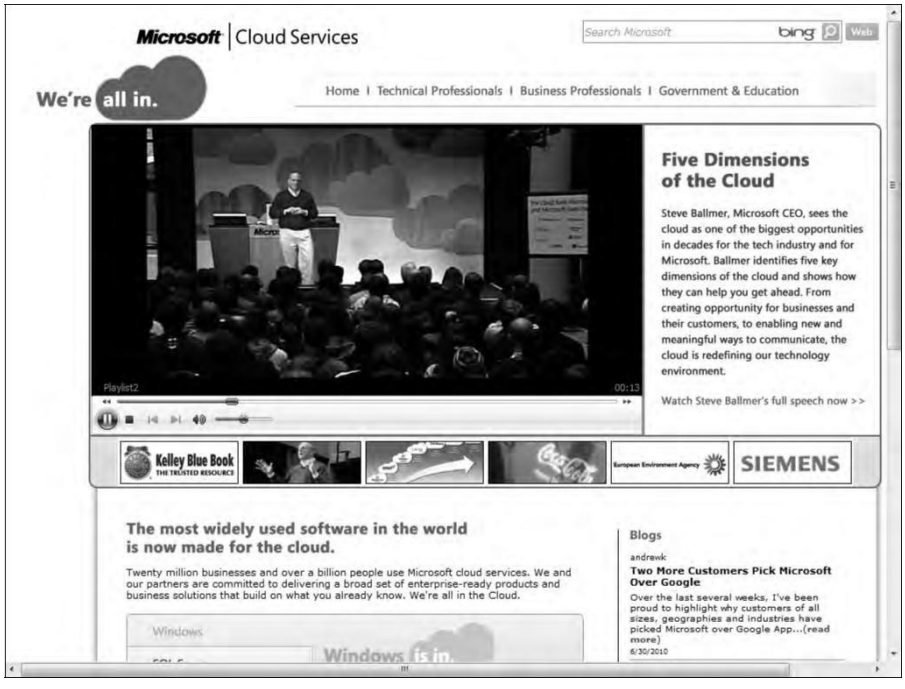
Microsoft CEO Steve Balmer recently said at a University of Washington speech that Microsoft was "betting our company" on the cloud. Balmer also claimed that about 70 percent of Microsoft employees were currently working on cloud-related projects and that the number was expected to rise to about 90 percent within a year. Plans to integrate cloud-based applications and services into the Microsoft product portfolio dominates the thinking at Microsoft and is playing a central role in the company's ongoing product development. The starting place for Microsoft's cloud computing efforts may be found at Microsoft.com/cloud, shown in Figure 10.1.

Microsoft has a vast array of cloud computing products and initiatives, and a number of industry-leading Web applications. Although services like America Online Instant Messenger (AIM) may garner mindshare in the United States, surprisingly Microsoft Messenger is the market leader in many other countries. Product by product in any category you can name—calendars, event managers, photo galleries, image editors, movie making, and so on—Microsoft has a Web application for it. Some of these products are also-rans, some are good, some are category leaders, and a few of them are really unique. What is also true is that Web apps are under very active development. Microsoft sees its on-line application portfolio as a way of extending its desktop applications to make the company pervasive and to extend its products' lives well into the future.

Going forward, Microsoft sees its future as providing the best Web experience for any type of device, which means that it structures its development environment so the application alters its behavior depending upon the device. For a mobile device, that would mean adjusting the user interface to accommodate the small screen, while for a PC the Web application would take advantage of the PC hardware to accelerate the application and add richer graphics and other features. That means Microsoft is pushing cloud development in terms of applications serving as both a service and an application. This duality—like light, both a particle and a wave—manifests itself in the way Microsoft is currently structuring its Windows Live Web products. Eventually, the company intends to create a Microsoft app store to sell cloud applications to users.

Microsoft Live is only one part of the Microsoft cloud strategy. The second part of the strategy is the extension of the .NET Framework and related development tools to the cloud. To enable .NET developers to extend their applications into the cloud, or to build .NET style applications that run completely in the cloud, Microsoft has created a set of .NET services, which it now refers to as the Windows Azure Platform. .NET Services itself had as its origin the work Microsoft did to create its BizTalk products.

Microsoft maintains a home page for cloud computing at http://www.microsoft.com/cloud.



Azure and its related services were built to allow developers to extend their applications into the cloud. Azure is a virtualized infrastructure to which a set of additional enterprise services has been layered on top, including:

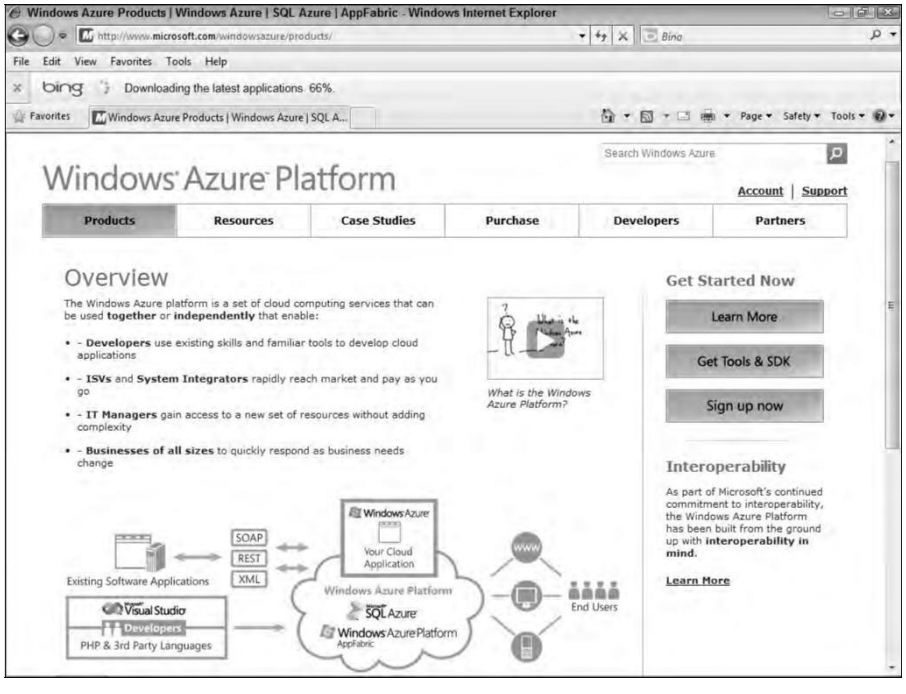
- A virtualization service called Azure **AppFabric** that creates an application hosting envi-ronment. AppFabric (formerly .NET Services) is a cloud-enabled version of the .NET Framework.
- A high capacity non-relational storage facility called Storage.
- A set of virtual machine instances called Compute.
- A cloud-enabled version of SQL Server called SQL Azure Database.
- A database marketplace based on SQL Azure Database code-named "Dallas."
- An xRM (Anything Relations Management) service called Dynamics CRM based on Microsoft Dynamics.
- A document and collaboration service based on SharePoint called SharePoint Services.
- Windows Live Services, a collection of services that runs on Windows Live, which can be used in applications that run in the Azure cloud.

Eventually the entire Microsoft server portfolio will be available as a cloud-based application or service, including Exchange. So the Windows Azure Platform can be viewed in a sense as the next Microsoft operating system, the first one that is a cloud OS. The Microsoft vision for the Windows Azure Platform is shown in Figure 10.2, where the company sees applications developed in Visual Studio or through PHP and other languages deployed to the cloud, existing local (on-premises) applications interacting with Azure with standard SOA protocols (SOAP, REST, and XML), all run-ning on the Windows Azure virtualized infrastructure.

The end result is pervasive computing available to users on the device of their choice. Just how Microsoft intends to integrate all these technologies into a unified offering is the story of this chapter.

FIGURE 10.2

The integrated vision for application development and deployment with Azure is illustrated in this over-view page of the Azure platform (http://www.microsoft.com/windowsazure/products/).



Defining the Windows Azure Platform

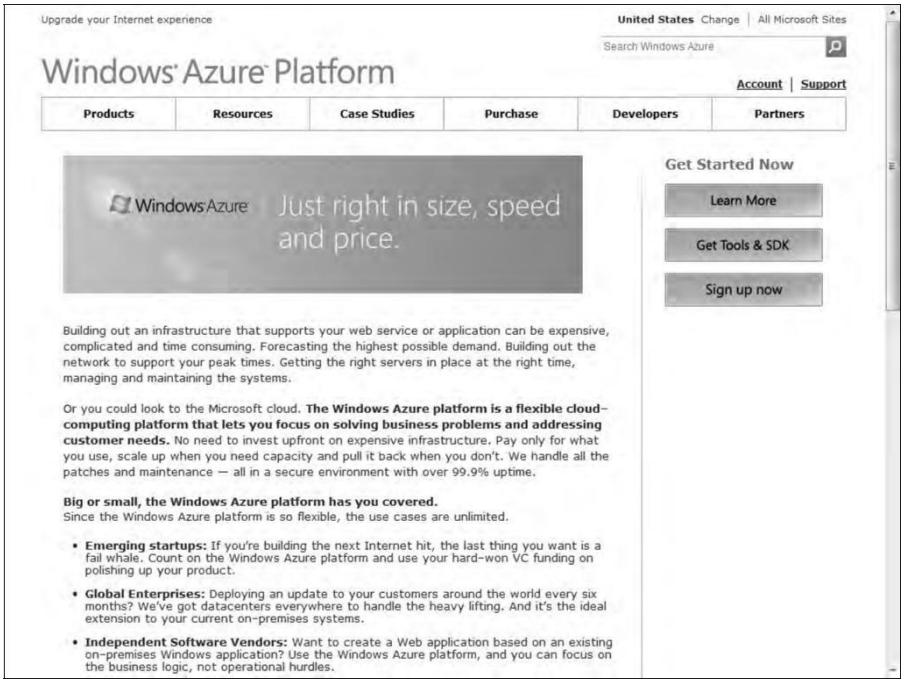
Azure is Microsoft's Infrastructure as a Service (IaaS) Web hosting service. Azure is a deep blue color, the color of the clear sky onto which you can paint clouds. Taken together as a unit, Windows Azure Platform becomes a Platform as a Service (PaaS) offering. Hence, you may run into some people calling Azure an infrastructure service and others calling it a platform; in context, both are correct. Compared to Amazon's and Google's cloud services, Azure (the service) is a com-petitor to AWS. Windows Azure Platform is a competitor to Google's App Engine.

Figure 10.3 shows the home page of the Windows Azure Platform found at http://www.microsoft.com/windowsazure.

A developer creates an Azure application by first logging onto the Azure portal from the Sign up now button shown in Figure 10.3, supplying a Windows Live ID, creating a hosted account, and provisioning a storage account. The completed application can then be made available to users as a hosted application or service.

FIGURE 10.3

Window Azure Platform's home page may be found at http://www.microsoft.com/windowsazure, and is shown in this figure.



The software plus services approach

Microsoft has a very different vision for cloud services than either Amazon or Google does. In Amazon's case, AWS is a pure infrastructure play. AWS essentially rents you a (virtual) computer on which to run your application. An Amazon Machine Image can be provisioned with an operating system, an enterprise application, or application stack, but that provisioning is not a prerequisite. An AMI is your machine, and you can configure it as you choose. AWS is a deployment enabler.

Google's approach with its Google App Engine (GAE) is to offer a cloud-based development platform on which you can add your program, provided that the program speaks the Google App Engine API and uses objects and properties from the App Engine framework. Google makes it pos-sible to program in a number of languages, but you must write your applications to conform to Google's infrastructure. Google Apps lets you create a saleable cloud-based application, but that application can only work within the Google infrastructure, and the application is not easily ported to other environments.

Microsoft sees the cloud as being a complimentary platform to its other platforms. The company envisages a scenario where a Microsoft developer with an investment in an application wants to extend that application's availability to the cloud. Perhaps the application runs on a server, desktop, or mobile device running some form of Windows. Microsoft calls this approach *software plus services.*

The Windows Azure Platform allows a developer to modify his application so it can run in the cloud on virtual machines hosted in Microsoft datacenters. Windows Azure serves as a cloud oper-ating system, and the suitably modified application can be hosted on Azure as a runtime applica-tion where it can make use of the various Azure Services. Additionally, local applications running on a server, desktop, or mobile device can access Windows Azure Services through the Windows Services Platform API.

Given that Microsoft owns the Office application market as well as the desktop OS market, this approach makes lots of sense. It is also quite possible that a hybrid application that can reside either locally or in the cloud will have lots of appeal not only to developers but to users who would prefer more control over their data and more security than the cloud might offer.

The Azure Platform

With Azure's architecture (shown in Figure 10.4), an application can run locally, run in the cloud, or some combination of both. Applications on Azure can be run as applications, as background pro-cesses or services, or as both. The Windows Azure service itself is shown as the oval in Figure 10.4 and is a cloud-based operating system with a fabric infrastructure of virtual machines hosted in Microsoft datacenters.

The Azure Windows Services Platform API uses the industry standard REST, HTTP, and XML pro-tocols that are part of any Service Oriented Architecture cloud infrastructure to allow applications to talk to Azure. Developers can install a client-side managed class library that contains functions that can make calls to the Azure Windows Services Platform API as part of their applications. These

API functions have been added to Microsoft Visual Studio as part of Microsoft's Integrated Development Environment (IDE). There are plans to add IPsec connectivity to Azure in the near future. *IPsec* refers to the Internet Protocol Security protocol suite for creating a secure Internet connection between two endpoints. IPsec provides for authenticated communication using session-based negotiation and the exchange of cryptographic keys to enable encrypted communication to be sent and decrypted. IPsec is an IETF standard that is in wide use.

The Azure Service Platform hosts runtime versions of .NET Framework applications written in any of the languages in common use, such as Visual Basic, C++, C#, Java, and any application that has been compiled for .NET's Common Language Runtime (CLR). Azure also can deploy Web-based applications built with ASP.NET, the Windows Communication Foundation (WCF), and PHP, and it supports Microsoft's automated deployment technologies. Microsoft also has released SDKs for both Java and Ruby to allow applications written in those languages to place calls to the Azure Service Platform API to the AppFabric Service.

The Windows Azure service

Windows Azure is a virtualized Windows infrastructure run by Microsoft on a set of datacenters around the world. In Figure 10.4, the dashed oval encloses the portion of the Windows Azure Platform that is Azure itself—that is, the portion of the platform that is the IaaS part, which is shown in more detail in Figure 10.5.

Six main elements are part of Windows Azure:

- **Application:** This is the runtime of the application that is running in the cloud.
- **Compute:** This is the load-balanced Windows server computation and policy engine that allows you to create and manage virtual machines that serve either in a Web role and a Worker role.

  A Web role is a virtual machine instance running Microsoft IIS Web server that can accept and respond to HTTP or HTTPS requests. A Worker role can accept and respond to requests, but doesn't run IIS in that virtual machine. Worker roles can communicate with Azure Storage or through direct connections to clients.
- **Storage:** This is a non-relational storage system for large-scale storage.

  Azure Storage Service lets you create drives, manage queues, and store BLOBs (Binary Large Objects). You manipulate content in Azure Storage using the REST API, which is based on standard HTTP requests and is therefore platform-independent. Stored data can be read using GETs, written with PUTs, modified with POSTs, and removed with DELETE requests.

  Azure Storage plays the same role in Azure that Amazon Simple Storage Service (S3) plays in Amazon Web Services. For relational database services, SQL Azure may be used.
- **Fabric:** This is the Windows Azure Hypervisor, which is a version of Hyper-V that runs on Windows Server 2008.
- **Config:** This is a management service.
- **Virtual machines:** These are instances of Windows that run the applications and services that are part of a particular deployment.

Windows Azure AppFabric

Azure AppFabric (http://msdn.microsoft.com/en-us/windowsazure/netservices. aspx) is a Service Bus and Access Control facility based on .NET technology for client requests to Web services on Azure. Previously, these services were called Microsoft .NET Services. Azure AppFabric supports the standard Service Oriented Architecture (SOA) protocols such as REST and SOAP and the WS-protocols.

The function of a service bus in an SOA is to expose distributed services as an endpoint with a spe-cific URI that clients can request services from, as shown in Figure 10.6. A particular set of end-points and its associated Access Control rules for an application is referred to as the service namespace. Each namespace is assigned a management key that is part of the security mechanism. The Service Bus service registry makes endpoints discoverable, if so configured.

Azure AppFabric manages requests by locating the service, communicating the request, and mak-ing the necessary connection possible by performing network address translation, opening appro-priate ports in any intervening firewalls. AppFabric manages the transaction to ensure that it is completed and that a response is sent to the client. A service bus also can serve to negotiate the exchange of information between a client and the service.

Azure AppFabric acts as an SOA service bus, as shown in Figure 10.6. AppFabric can provide a negotiated traversal of services through firewalls and NATs as a relay service using the Service Bus' rendezvous address. A rendezvous address not only includes the service URI, but also includes the namespace of the service bus. Alternatively, if both applications comply to .NET Services a direct connection between the applications can be used instead with the required NAT traversal informa-tion for the direct connection provided by the relay service of the Service Bus. NAT (Network Address Traversal) is a system for creating and maintaining Internet connections for TCP or UDP traffic where the connection point is hidden behind a router or a firewall and routing is performed by one of several possible mechanisms.

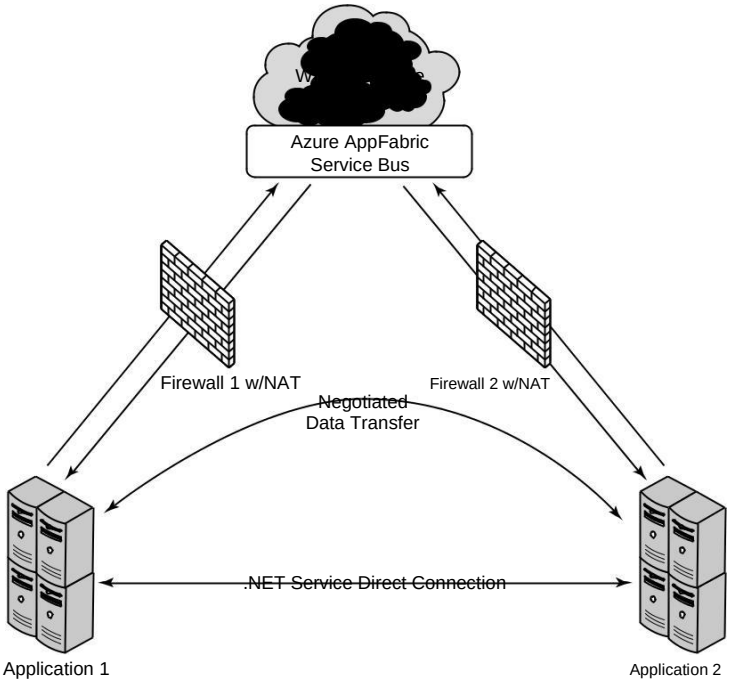
Cross-Ref

In Chapter 13, "Understanding Service Oriented Architecture," the role of a service bus in a Service Oriented Architecture is more fully explored. ■

The Access Control portion of Azure AppFabric is a claims access control system that provides a token-based trust mechanism for identity management. An application or user, as shown on the right of Figure 10.7, presents a claim for a service from an application on the left. The Access Control examines the request, and if it finds it to be valid, it grants a security token to the client.

FIGURE 10.6

Azure AppFabric service pathways



These steps are associated with Access Control:

1. The client requests authentication from Access Control.
2. Access Control creates a token based on the stored rules for server application.
3. A token is signed and returned to the client application.
4. The client presents the token to the service application.
5. The server application verifies the signature and uses the token to decide what the client application is allowed to do.

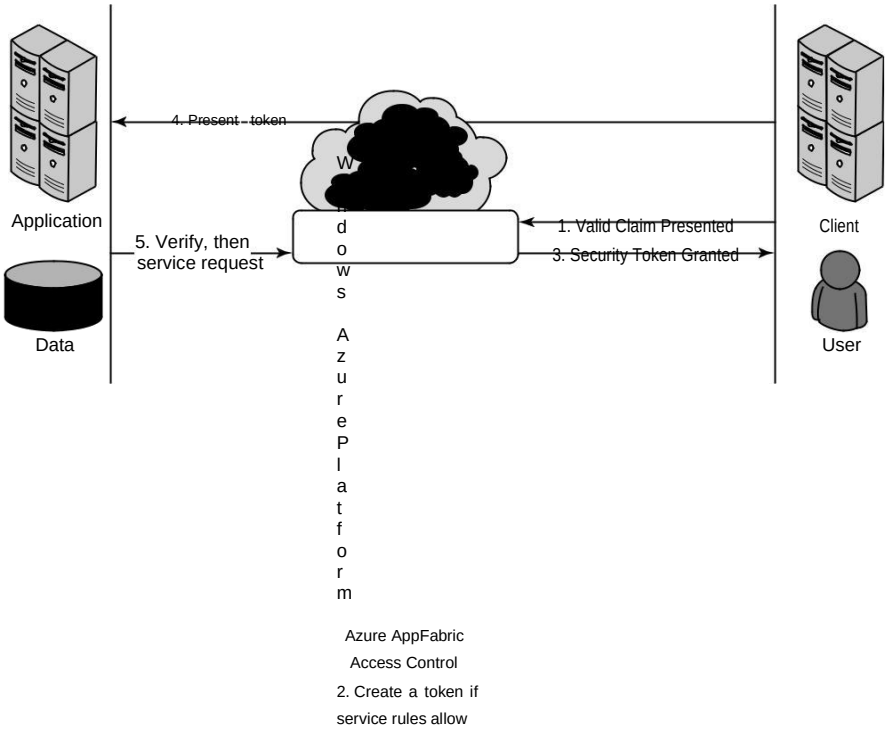
Access Control allows one application to trust the identity of another application. This mechanism can federate with identity providers such as Active Directory Federation Services (ADFS v2) to cre-ate distributed systems based on SOA.

Note

"AppFabric" is also used by Microsoft for the name of its local server deployment technology called Windows Server AppFabric (http://msdn.microsoft.com/en-us/windowsserver/ee695849.aspx**).**
Windows Server AppFabric enables Web data caching for application data and provides managed services using WindowsWorkflow Foundation and the Windows Communication Foundation. There are plans to inte-grate Windows Server AppFabric into the Azure platform AppFabic. ■

FIGURE 10.7

Azure AppFabric Access Control enables secure application requests through a token mechanism.



Microsoft likes to refer to the Azure AppFabric as an "Internet Service Bus" to differentiate it from the standard Enterprise Service Bus (ESB) that you find in SOA architectures. AppFabric has all the same components of an ESB, namely service orchestration, federated identity, access control, a namespace, service registry, and a messaging fabric, but it locates these components in the cloud. Often ESBs are located on LANs. According to Microsoft, this approach abstracts away from appli-cation developers the challenges related to NAT traversal, DDNS (Dynamic DNS), and UPnP. ESBs are described in Chapter 13; please refer to that chapter for further discussion on this topic.

Azure Content Delivery Network

The Windows Azure Content Delivery Network (CDN) is a worldwide content caching and deliv-ery system for Windows Azure blob content. Currently, more than 18 Microsoft datacenters are hosting this service in Australia, Asia, Europe, South America, and the United States, referred to as endpoints. CDN is an edge network service that lowers latency and maximizes bandwidth by deliv-ering content to users who are nearby.

Any storage account can be enabled for CDN. In order to share information stored in an Azure blob, you need to place the blob in a public blob container that is accessible to anyone using an anonymous sign-in. The Azure portal lists the domain name of the blob container in the form http://<*guid*>.vo.msecnd.net/. You also can register a custom domain name for a Windows Azure CDN endpoint.

For content in a public container named "Box" in the storage account named "MyAccount," a user would access the content with one of the following URLs:

- Windows Azure Blob services URL: http://<*MyAccount*>.blob.core.windows.net/<*Box*>/
- Windows Azure CDN URL: http://<*guid*>.vo.msecnd.net/<*Box*>/

When the Blob service URL is used, the request is redirected to the closest CDN endpoint to the client. The CDN service searches that location and serves the content; if the content isn't found, the CDN retrieves the Blob from the Blob service, caches the content, and then serves it to the user. Parameters can be set that determine how long content is cached (Time-To-Live, TTL), with the default being 72 hours.

SQL Azure

SQL Azure is a cloud-based relational database service that is based on Microsoft SQL Server. Initially, this service was called SQL Server Data Service. An application that uses SQL Azure Database can run locally on a server, PC, or mobile device, in a datacenter, or on Windows Azure. Data stored in an SQL Azure database is accessed using the Tabular Data Stream (TDS) protocol, the same protocol used for a local SQL Server database. SQL Azure Database supports Transact-SQL statements.

Azure data is replicated three times for data protection and writes are checked for consistency. SQL Azure eventually will support the Microsoft Sync Framework providing a facility for SQL Azure Databases to synchronize their data with local databases.

There is a current limit of 10GB for each SQL Azure Database. Queries against a single database are unified. However, if the storage size exceeds the limit, then data must be partitioned into logical sets and queries need to be structured to account for this partitioning. For example, names in a database might have to be partitioned A-K, L-R, and S-Z. SQL Azure Database is a shared database environment, and limitations are placed on how long a query can run or how many resources a query can use.

Note

Microsoft hopes to create a cloud-based global data marketplace using SQL Azure for stored information, a project that it has codenamed "Dallas." Applications will then be able to access both private and public domain data such as imagery, census data, statistical data, and other premium content using REST protocols. You can read about Microsoft's plans for Dallas at http://www.microsoft.com/windowsazure/ dallas/**.** ■

From the standpoint of any application, an SQL Azure Database looks like and behaves like a local database with a few exceptions. The current exceptions are that the SQL Common Language Runtime (CLR) and support for spatial data were not included, although support will be added later for them. The biggest difference is that because SQL Azure is managed in the cloud, there are no administrative controls over the SQL engine. You can't shut the system down, nor can you directly interact with the SQL Servers.

Windows Azure pricing

Prices for working with the Windows Azure Platform are based either on a “consumption” (pay-as-you-go) model or through various contracts for levels of monthly service that Microsoft calls “com-mitments.” When you exceed the subscription level of your commitment, the additional usage is charged on the consumption model.

Current pricing for Windows Azure is as follows:

- Compute: $0.12 / hour
- Storage: $0.15 / GB stored / month
- Storage transactions: $0.01 / 10K
- Data transfers (excluding CDN): $0.10 in / $0.15 out / GB ($0.30 in / $0.45 out / GB in Asia)
- CDN data transfers: $0.15 GB for North America and Europe ($0.20 GB elsewhere)
- CDN transactions: $0.01 / 10K

  A transaction is an application request.

In the Windows Azure Service Level Agreement, Microsoft states that it guarantees an external con-nectivity between two or more role instances that are located in different Azure domains of at least 99.95 percent uptime. The connection between storage and Microsoft's Content Delivery Network (CDN, described below) is stated to be at least 99.9 percent uptime.

SQL Azure charges are based on two different programs:

- Web Editions: Up to 1GB database = $9.99 / month; up to 5GB database = $49.95 / month
- Business Edition: Up to 10GB database = $99.99 / month; up to 20GB database = $199.98 / month; up to 30GB database = $299.97 / month; up to 40GB database = $399.96 / month; up to 50GB database = $499.95 / month
- Data transfers: $0.10 in / $0.15 out / GB ($0.30 in / $0.45 out / GB in Asia)

Note

The various consumption and subscription offers for the Windows Azure platform are summarized on the Pricing page at http://www.microsoft.com/windowsazure/pricing/**.** ■

These are the charges for Windows Azure Platform AppFabric:

- Access Control transactions of $1.99 / 100K transactions
- Service Bus connections: $3.99 per connection on a "pay-as-you-go" basis, $9.95 for a pack of 5 connections, $49.75 for a pack of 25 connections, $199 for a pack of 100 connections, and $995 for a pack of 500 connections
- Data transfers: $0.10 in / $0.15 out / GB ($0.30 in / $0.45 out / GB in Asia)

Given that Windows Azure is a relatively new service and that IaaS likely will become a very com-petitive market, pricing is sure to change over time. You should definitely check the pricing page for current pricing if you are thinking of deploying on Azure.

Microsoft offers a TCO calculator for the Windows Azure Platform that you may find useful in determining your costs and savings. To access the calculator use the following link: http://www.microsoft.com/windowsazure/economics/#tcoCompare-LB. The calculator was described in brief in Chapter 2, "Computing the Total Cost of Ownership."

Windows Live services

Windows Live is a collection of cloud-based applications and services, some of which can be used inside applications that run on Windows Azure Platform. Some Windows Live applications run as standalone applications and are available to users directly through a browser. Others are services that add capabilities to the Windows Azure Platform as part of Microsoft's software plus services strategy.

Microsoft has rolled out Windows Live in sets of releases they describe as four waves. The first wave was a rebranding of several Microsoft MSN applications and services in late 2005. More applications including Windows Mail, Windows Photo Gallery, and Windows Movie Maker were unbundled from Vista and rolled into a downloadable software suite called Windows Live Essentials. There has been continuous development, branding, marketing, and rebranding of the Windows Live portfolio that has had many people scratching their heads. Many Windows Live applications have been rolled into other services or discontinued entirely.

Here's what I believe the current situation is with Windows Live. If an application is bundled as part of an additional download for desktop users, it is part of the Windows Live Essentials pack-age. Some applications that are part of Windows Live are standalone products, while others are extensions of existing Microsoft commercial software. An example of a standalone product would be Windows Live Calendar. An example of a cloud-based line extension is Windows Live Office, described more fully in Chapter 16.

Some parts of the Windows Live portfolio are shared applications and services that are accessible to developers, and those services are the Windows Live Services that are one component of the Windows Azure Platform. Developers access the services for Windows Live Services through a col-lection of APIs and controls called Windows Live Messenger Connect (previously called Live Services and Windows Live Dev). Using these APIs and controls, developers can add Windows Live Services capabilities and data to their application.

Note

To learn more about Windows Live Messenger Connect, visit the MSDN site's documentation found at http://msdn.microsoft.com/en-us/library/ff749458.aspx**.** ■

Messenger Connect was released as part of the Windows Live Wave 4 at the end of June 2010, and it unites APIs such as Windows Live ID, Windows Live Contacts, and Windows Live Messenger Web Toolkit into a single API. Messenger Connect works with ASP.NET, Windows Presentation Foundation (WPF), Java, Adobe Flash, PHP, and Microsoft's Silverlight graphics rendering technol-ogy through four different methods:

- Messenger Connect REST API Service
- Messenger Connect .NET and Silverlight Libraries
- Messenger Connection JavaScript Libraries and Controls
- Web activity feeds, either RSS 2.0 or ATOM

Using Windows Live

Windows Live includes several popular cloud-based services. The two best known and most widely used are Windows Live Hotmail and Windows Live Messenger, with more than 300 million users worldwide. Windows Live is based around five core services:

- E-mail
- Instant Messaging
- Photos
- Social Networking
- Online Storage

A user or application can consume Windows Live in a number of ways. Some Windows Live applications are entirely cloud-based Web services, so users can use these applications from within any browser. The Office Live applications described more fully in Chapter 16, "Microsoft Office Web Apps," is an example of this sort of service. Some of these services are aimed at mobile devices and are referred to as Windows Live for Mobile (described below), and they are consumed on conform-ing mobile devices. Some of these applications are client-side applications that you download from Windows Live for use on your desktop, of which Windows Live Essentials is the primary example.

You can access Windows Live services in one of the following ways:

- By navigating to the service from the command on the navigation bar at the top of Windows Live
- By directly entering the URL of the service
- By selecting the application from the Windows Live Essentials folder on the Start menu

Summary

In this chapter, I described Microsoft cloud computing strategy. Microsoft seeks to extend its products into the cloud using a software plus service approach. In this model, the cloud is yet another platform, and applications can run locally and access cloud services, run in the cloud and be made available through SOA standard protocols, or some combination of both.

Microsoft's cloud operating system is called Windows Azure. Windows Azure is a hosted environ-ment of virtualized systems tied together in a fabric using a service called AppFabric. This is offered to developers in the form of an Infrastructure as a Service model similar to Amazon Web Services. To Windows Azure is added a cloud-enabled version of the .NET Framework originally called .NET Services, which are now part of Azure AppFabric. This approach lets developers extend their applications into the cloud using development tools that they already possess with the minimum amount of reconfiguration. Microsoft has added a number of additional services and the entire offering is now a Platform as a Service cloud model that Microsoft calls the Windows Azure Platform.

The other major component of Microsoft's cloud computing strategy is a collection of user applica-tions and related services called Windows Live. Some Windows Live applications are client-side applications, many others are Web-based applications, some are mobile apps, and an important subset of these services is available to developers through the Windows Live Messenger Connect APIs. The various offering in Windows Live were discussed in this chapter.

In Chapter 11, "Managing the Cloud," you learn about some of the management tools used to work with cloud applications and methods used for application deployment.

Questions

1. Explain Abstraction and Virtualization. Explain the types of virtualization. Give the basic concept of load balancing.
2. Explain the network resources for load balancing. Give the classifications of virtualization environment.
3. Briefly explain the different types of virtualization environment.
4. Explain the virtual machine technology and its types. Write short notes on VMware, vSphere and Machine imaging.
5. Explain the porting of applications in the cloud by throwing light on simple Cloud API and AppZero Virtual Application appliance.
6. Define the services as used in PaaS. Compare between SaaS and PaaS. What are the uses of PaaS Application frameworks?
7. Explain the Google Applications Portfolio in terms of Indexed search, Dark Web, Aggregation and disintermediation, Productivity applications and service, Adwords, Google Analytics, Google Translate.
8. Briefly discuss the Google Toolkit. Give the major features of Google App Engine service.
9. Explain the Amazon Web Services by throwing light on Amazon Elastic Cloud, Amazon Simple Storage system, Amazon Elastic Block Store, Amazon SimpleDB and Relational Database Service.
10. Explain the Microsoft Cloud Services in terms of Microsoft's approach, architecture, and main elements, overview of Windows Azure AppFabric, Content Delivery Network, SQL Azure, and Windows Live services.

Module – III

Cloud Infrastructure

Managing the Cloud

Cloud computing deployments must be monitored and managed in order to be optimized for best performance. To the problems associ-ated with analyzing distributed network applications, the cloud adds the complexity of virtual infrastructure. This is one of the most active areas of product development in the entire cloud computing industry, and this chapter introduces you to the different products in this nascent area.

Cloud management software provides capabilities for managing faults, configuration, accounting, performance, and security; this is referred to as FCAPS. Many products address one or more of these areas, and through network frameworks, you can access all five areas. Framework products are being repositioned to work with cloud systems.

Your management responsibilities depend on the particular service model for your cloud deployment. Cloud management includes not only managing resources in the cloud, but managing resources on-premises. The management of resources in the cloud requires new technology, but management of resources on-premises allows vendors to use well-established network management technologies.

The lifecycle of a cloud application includes six defined parts, and each must be managed. In this chapter, the tasks associated with each stage are described.

Efforts are underway to develop cloud management interoperability standards. One effort you learn about in this chapter is the DMTF's (Distributed Management Task Force) Open Cloud Standards Incubator. The goal of these efforts is to develop management tools that work with any cloud type. Another group called the Cloud Commons is developing a technology called the Service Measurement Index (SMI). SMI aims to deploy methods for mea-suring various aspects of cloud performance in a standard way.

Administrating the Clouds

The explosive growth in cloud computing services has led many vendors to rename their products and reposition them to get in on the gold rush in the clouds. What was once a network management product is now a cloud management product. Nevertheless, this is one area of technology that is very actively funded, comes replete with interesting startups, has been the focus of several recent strategic acquisitions, and has resulted in some interesting product alliances. Let's join the party and see what all the fuss is about.

These fundamental features are offered by traditional network management systems:

- Administration of resources
- Configuring resources
- Enforcing security
- Monitoring operations
- Optimizing performance
- Policy management
- Performing maintenance
- Provisioning of resources

Network management systems are often described in terms of the acronym FCAPS, which stands for these features:

- **F**ault
- **C**onfiguration
- **A**ccounting
- **P**erformance
- **S**ecurity

Most network management packages have one or more of these characteristics; no single package provides all five elements of FCAPS.

To get the complete set of all five of these management areas from a single vendor, you would need to adopt a network management framework. These large network management frameworks were industry leaders several years back: BMC PATROL, CA Unicenter, IBM Tivoli, HP OpenView, and Microsoft System Center. Network framework products have been sliced and diced in many differ-ent ways over the years, and they are rebranded from time to time. Today, for example, BMC PATROL is now part of BMC ProactiveNet Performance Management (http://www.bmc.com/ products/product-listing/ProactiveNet-Performance-Management.html), HP OpenView has been split (https://h10078.www1.hp.com/cda/hpms/display/main/ hpms_content.jsp?zn=bto&cp=1-10^36657_4000_100) into a set of HP Manager products.

The impact that cloud computing is having on network frameworks is profound. These five vendors have (or soon will have) products for cloud management. Computer Associates, for example, has completely repositioned its network management portfolio as an IT Management

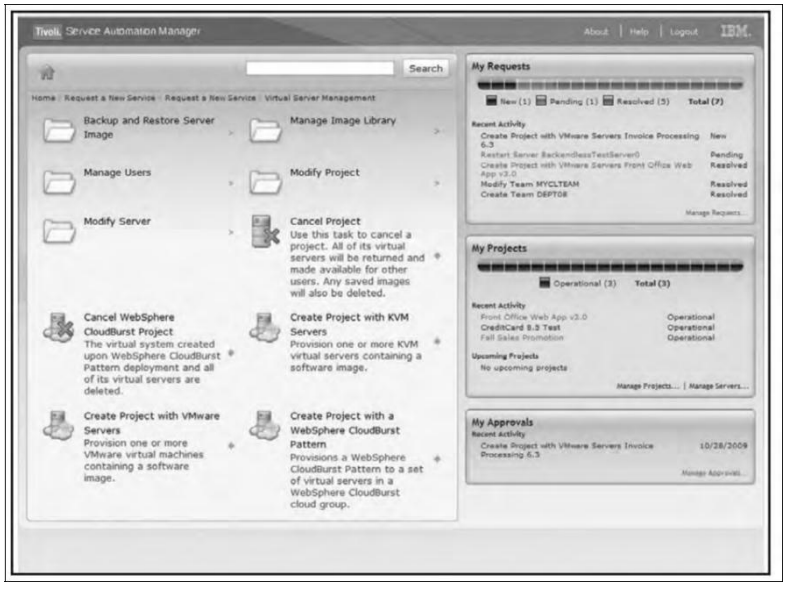
Software as a Service. Find the cloud products for these five large cloud vendors at the following URLs:

- BMC Cloud Computing (http://www.bmc.com/solutions/esm-initiative/ cloud-computing.html)
- Computer Associates Cloud Solutions (http://www.ca.com/us/cloud-computing.aspx)
- HP Cloud Computing (http://h20338.www2.hp.com/enterprise/w1/en/ technologies/cloud-computing-overview.html)
- IBM Cloud Computing (http://www.ibm.com/ibm/cloud/)
- Microsoft Cloud Services (http://www.microsoft.com/cloud/)

Figure 11.1 shows IBM Tivoli Service Automation Manager, a framework tool for managing cloud infrastructure.

FIGURE 11.1

Tivoli Service Automation Manager lets you create and stage cloud-based servers.



Management responsibilities

What separates a network management package from a cloud computing management package is the "cloudly" characteristics that cloud management service must have:

- Billing is on a pay-as-you-go basis.
- The management service is extremely scalable.
- The management service is ubiquitous.
- Communication between the cloud and other systems uses cloud networking standards.

To monitor an entire cloud computing deployment stack, you monitor six different categories:

1. End-user services such as HTTP, TCP, POP3/SMTP, and others
2. Browser performance on the client
3. Application monitoring in the cloud, such as Apache, MySQL, and so on
4. Cloud infrastructure monitoring of services such as Amazon Web Services, GoGrid, Rackspace, and others
5. Machine instance monitoring where the service measures processor utilization, memory usage, disk consumption, queue lengths, and other important parameters
6. Network monitoring and discovery using standard protocols like the Simple Network Management Protocol (SNMP), Configuration Management Database (CMDB), Windows Management Instrumentation (WMI), and the like

It's important to note that there are really two aspects to cloud management:

Managing resources *in the cloud*

Using the cloud to manage resources *on-premises*

When you move to a cloud computing architecture from a traditional networked model like client/server or a three-tier architecture, many of the old management tasks for processes going on in the cloud become irrelevant or nearly impossible to manage because the tools to effectively manage resources of various kinds fall outside of your own purview. In the cloud, the particular service model you are using directly affects the type of monitoring you are responsible for.

Consider the case of an Infrastructure as a Service vendor such as Amazon Web Services or Rackspace. You can monitor your usage of resources either through their native monitoring tools like Amazon CloudWatch or Rackspace Control Panel or through the numerous third-party tools that work with these sites' APIs. In IaaS, you can alter aspects of your deployment, such as the number of machine instances you are running or the amount of storage you have, but you have very limited control over many important aspects of the operation. For example, your network bandwidth is locked into the type of instance you deploy. Even if you can provision more bandwidth, you likely have no control over how network traffic flows into and out of the system, whether there is packet prioritization, how routing is done, and other important characteristics.

The situation—as you move first to Platform as a Service (PaaS) like Windows Azure or Google App Engine and then onto Software as a Service (SaaS) for which Salesforce.com is a prime exam-ple—becomes even more restrictive. When you deploy an application on Google's PaaS App Engine cloud service, the Administration Console provides you with the following monitoring capabilities:

- Create a new application, and set it up in your domain.
- Invite other people to be part of developing your application.
- View data and error logs.
- Analyze your network traffic.
- Browse the application datastore, and manage its indexes.
- View the application's scheduled tasks.
- Test the application, and swap out versions.

However, you have almost no operational control. Essentially, Google App Engine lets you deploy the application and monitor it, and that's about it. All the management of devices, networks, and other aspects of the platform are managed by Google. You have even less control when you are selling software in the cloud, as you would with Salesforce.com.

Figure 11.2 graphically summarizes the management responsibilities by service model type.

The second aspect of cloud management is the role that cloud-based services can play in managing on-premises resources. From the standpoint of the client, a cloud service provider is no different than any other networked service. The full range of network management capabilities may be brought to bear to solve mobile, desktop, and local server issues, and the same sets of tools can be used for measurement.

Microsoft System Center is an example of how management products are being adapted for the cloud. System Center provides tools for managing Windows servers and desktops. The manage-ment services include an Operations Manager, the Windows Service Update Service (WSUS), a Configuration Manager for asset management, a Data Protection Manager, and a Virtual Machine Manager, among other components.

One of these service sets was called the System Center Online Desktop Manager (SCODM). Microsoft has taken SCODM and repositioned it as a cloud-based service for managing updates, monitoring PCs for license compliance and health, enforcing security policies, and using Forefront protect systems from malware, and the company has branded it as Windows Intune (http:// www.microsoft.com/windows/windowsintune/default.aspx). From the client's standpoint, it makes little difference whether the service is in the cloud or on a set of servers in a datacenter. The benefit of a cloud management service accrues to the organization responsible for managing the desktops or mobile devices. Figure 11.3 shows an Overview screen from the beta ver-sion of Windows Intune. The product is due to be released in the first or second quarter of 2011.

FIGURE 11.2

Management responsibilities by service model type

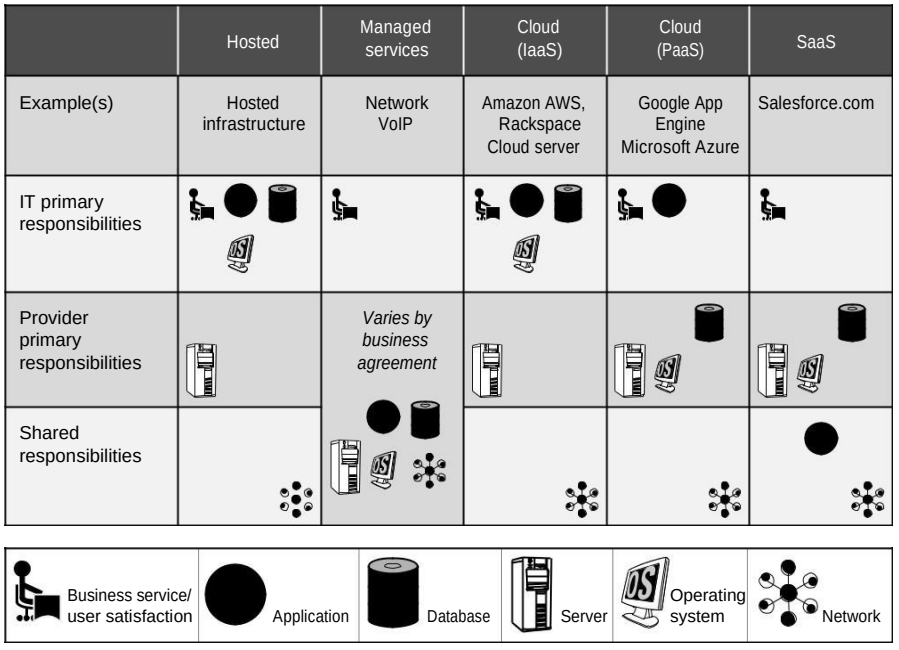
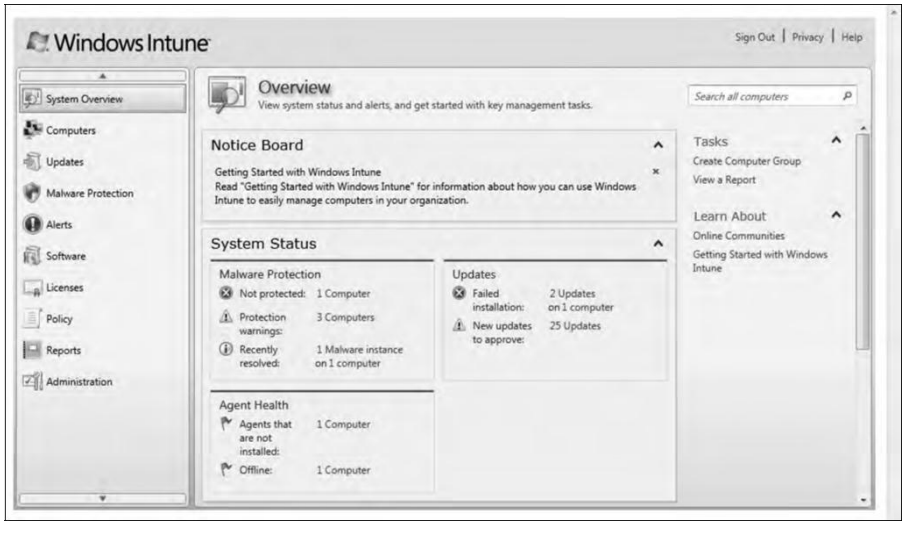


FIGURE 11.3

Intune is Microsoft's cloud-based management service for Windows systems.



Lifecycle management

Cloud services have a defined lifecycle, just like any other system deployment. A management program has to touch on each of the six different stages in that lifecycle:

1. The definition of the service as a template for creating instances

   Tasks performed in Phase 1 include the creation, updating, and deletion of service templates.

2. Client interactions with the service, usually through an SLA (Service Level Agreement) contract

   This phase manages client relationships and creates and manages service contracts.

3. The deployment of an instance to the cloud and the runtime management of instances Tasks

   performed in Phase 3 include the creation, updating, and deletion of service offerings.

4. The definition of the attributes of the service while in operation and performance of modifications of its properties

   The chief task during this management phase is to perform service optimization and customization.

5. Management of the operation of instances and routine maintenance

   During Phase 5, you must monitor resources, track and respond to events, and perform reporting and billing functions.

6. Retirement of the service

   End of life tasks include data protection and system migration, archiving, and service contract termination.

Understanding Cloud Security

Cloud computing has lots of unique properties that make it very valu-able. Unfortunately, many of those properties make security a singu-lar concern. Many of the tools and techniques that you would use to protect your data, comply with regulations, and maintain the integrity of your systems are complicated by the fact that you are sharing your systems with others and many times outsourcing their operations as well. Cloud computing service providers are well aware of these concerns and have developed new technologies to address them.

Different types of cloud computing service models provide different levels of security services. You get the least amount of built in security with an Infrastructure as a Service provider, and the most with a Software as a Service provider. This chapter presents the concept of a security boundary separating the client's and vendor's responsibilities.

Adapting your on-premises systems to a cloud model requires that you deter-mine what security mechanisms are required and mapping those to controls that exist in your chosen cloud service provider. When you identify missing security elements in the cloud, you can use that mapping to work to close the gap.

Storing data in the cloud is of particular concern. Data should be transferred and stored in an encrypted format. You can use proxy and brokerage services to separate clients from direct access to shared cloud storage.

Logging, auditing, and regulatory compliance are all features that require planning in cloud computing systems. They are among the services that need to be negotiated in Service Level Agreements.

Also in this chapter, you learn about identity and related protocols from a security standpoint. The concept of presence as it relates to identity is also introduced.

IN THIS CHAPTER

Reviewing cloud security concerns

Understanding how cloud data can be secured

Planning for security in your system

Learning how identity is used to allow secure cloud access

Securing the Cloud

The Internet was designed primarily to be resilient; it was not designed to be secure. Any distrib-uted application has a much greater attack surface than an application that is closely held on a Local Area Network. Cloud computing has all the vulnerabilities associated with Internet applica-tions, and additional vulnerabilities arise from pooled, virtualized, and outsourced resources.

In the report "Assessing the Security Risks of Cloud Computing," Jay Heiser and Mark Nicolett of the Gartner Group (http://www.gartner.com/DisplayDocument?id=685308) high-lighted the following areas of cloud computing that they felt were uniquely troublesome:

- Auditing
- Data integrity
- e-Discovery for legal compliance
- Privacy
- Recovery
- Regulatory compliance

Your risks in any cloud deployment are dependent upon the particular cloud service model chosen and the type of cloud on which you deploy your applications. In order to evaluate your risks, you need to perform the following analysis:

1. Determine which resources (data, services, or applications) you are planning to move to the cloud.
2. Determine the sensitivity of the resource to risk.

   Risks that need to be evaluated are loss of privacy, unauthorized access by others, loss of data, and interruptions in availability.
3. Determine the risk associated with the particular cloud type for a resource.

   Cloud types include public, private (both external and internal), hybrid, and shared community types. With each type, you need to consider where data and functionality will be maintained.
4. Take into account the particular cloud service model that you will be using.

   Different models such as IaaS, SaaS, and PaaS require their customers to be responsible for security at different levels of the service stack.
5. If you have selected a particular cloud service provider, you need to evaluate its system to understand how data is transferred, where it is stored, and how to move data both in and out of the cloud.

   You may want to consider building a flowchart that shows the overall mechanism of the system you are intending to use or are currently using.

One technique for maintaining security is to have "golden" system image references that you can return to when needed. The ability to take a system image off-line and analyze the image for

vulnerabilities or compromise is invaluable. The compromised image is a primary forensics tool. Many cloud providers offer a snapshot feature that can create a copy of the client's entire environ-ment; this includes not only machine images, but applications and data, network interfaces, fire-walls, and switch access. If you feel that a system has been compromised, you can replace that image with a known good version and contain the problem.

Many vendors maintain a security page where they list their various resources, certifications, and credentials. One of the more developed offerings is the AWS Security Center, shown in Figure 12.1, where you can download some backgrounders, white papers, and case studies related to the Amazon Web Service's security controls and mechanisms.

FIGURE 12.1

The AWS Security Center (http://aws.amazon.com/security/) is a good place to start learning about how Amazon Web Services protects users of its IaaS service.

The security boundary

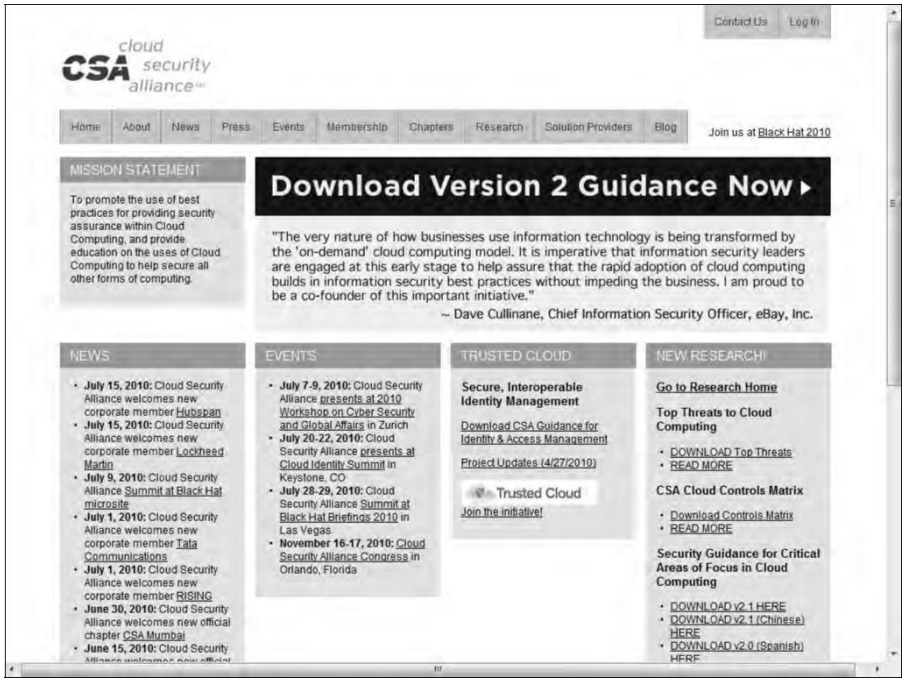
In order to concisely discuss security in cloud computing, you need to define the particular model of cloud computing that applies. This nomenclature provides a framework for understanding what security is already built into the system, who has responsibility for a particular security mechanism, and where the boundary between the responsibility of the service provider is separate from the responsibility of the customer.

All of Chapter 1 was concerned with defining what cloud computing is and defining the lexicon of cloud computing. There are many definitions and acronyms in the area of cloud computing that will probably not survive long. The most commonly used model based on U.S. National Institute of Standards and Technology (NIST; http://www.csrc.nist.gov/groups/SNS/cloud-computing/index.html) separates deployment models from service models and assigns those models a set of service attributes. Deployment models are cloud types: community, hybrid, private, and public clouds. Service models follow the SPI Model for three forms of service delivery: Software, Platform, and Infrastructure as a Service. In the NIST model, as you may recall, it was not required that a cloud use virtualization to pool resources, nor did that model require that a cloud support multi-tenancy. It is just these factors that make security such a complicated proposi-tion in cloud computing.

Chapter 1 also presented the Cloud Security Alliance (CSA; http://www.cloudsecurity alliance.org/) cloud computing stack model, which shows how different functional units in a net-work stack relate to one another. As you may recall from Chapter 1, this model can be used to separate the different service models from one another. CSA is an industry working group that studies security issues in cloud computing and offers recommendations to its members. The work of the group is open and available, and you can download its guidance from its home page, shown in Figure 12.2.

FIGURE 12.2

The Cloud Security Alliance (CSA) home page at http://www.cloudsecurityalliance.org/ offers a number of resources to anyone concerned with securing his cloud deployment.



The CSA partitions its guidance into a set of operational domains:

- Governance and enterprise risk management
- Legal and electronic discovery
- Compliance and audit
- Information lifecycle management
- Portability and interoperability
- Traditional security, business continuity, and disaster recovery
- Datacenter operations
- Incidence response, notification, and remediation
- Application security
- Encryption and key management
- Identity and access management
- Virtualization

You can download the group's current work in these areas from the different sections of its Web site.

One key difference between the NIST model and the CSA is that the CSA considers multi-tenancy to be an essential element in cloud computing. Multi-tenancy adds a number of additional security concerns to cloud computing that need to be accounted for. In multi-tenancy, different customers must be isolated, their data segmented, and their service accounted for. To provide these features, the cloud service provider must provide a policy-based environment that is capable of supporting different levels and quality of service, usually using different pricing models. Multi-tenancy expresses itself in different ways in the different cloud deployment models and imposes security concerns in different places.

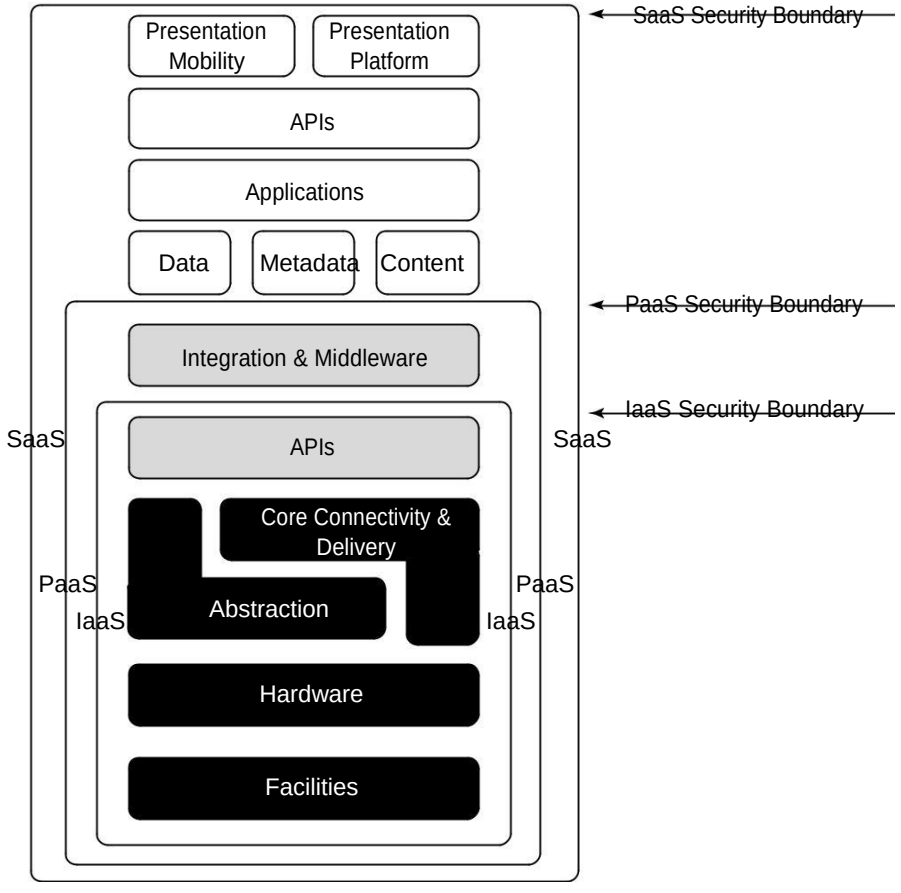
Security service boundary

The CSA functional cloud computing hardware/software stack is the Cloud Reference Model. This model, which was discussed in Chapter 1, is reproduced in Figure 12.3. IaaS is the lowest level service, with PaaS and SaaS the next two services above. As you move upward in the stack, each service model inherits the capabilities of the model beneath it, as well as all the inherent security concerns and risk factors. IaaS supplies the infrastructure; PaaS adds application development frameworks, transactions, and control structures; and SaaS is an operating environment with applications, management, and the user interface. As you ascend the stack, IaaS has the least levels of integrated functionality and the lowest levels of integrated security, and SaaS has the most.

The most important lesson from this discussion of architecture is that each different type of cloud service delivery model creates a security boundary at which the cloud service provider's responsibilities end and the customer's responsibilities begin. Any security mechanism below the security boundary must be built into the system, and any security mechanism above must be maintained by the customer. As you move up the stack, it becomes more important to make sure that the type and level of security is part of your Service Level Agreement.

FIGURE 12.3

The CSA Cloud Reference Model with security boundaries shown



In the SaaS model, the vendor provides security as part of the Service Level Agreement, with the compliance, governance, and liability levels stipulated under the contract for the entire stack. For the PaaS model, the security boundary may be defined for the vendor to include the software framework and middleware layer. In the PaaS model, the customer would be responsible for the security of the application and UI at the top of the stack. The model with the least built-in security is IaaS, where everything that involves software of any kind is the customer's problem. Numerous definitions of services tend to muddy this picture by adding or removing elements of the various functions from any particular offering, thus blurring which party has responsibility for which features, but the overall analysis is still useful.

In thinking about the Cloud Security Reference Model in relationship to security needs, a fundamental distinction may be made between the nature of how services are provided versus where those services are located. A private cloud may be internal or external to an organization, and although a public cloud is most often external, there is no requirement that this mapping be made so. Cloud computing has a tendency to blur the location of the defined security perimeter in such a way that the previous notions of network firewalls and edge defenses often no longer apply.

This makes the location of trust boundaries in cloud computing rather ill defined, dynamic, and sub-ject to change depending upon a number of factors. Establishing trust boundaries and creating a new perimeter defense that is consistent with your cloud computing network is an important consider-ation. The key to understanding where to place security mechanisms is to understand where physi-cally in the cloud resources are deployed and consumed, what those resources are, who manages the resources, and what mechanisms are used to control them. Those factors help you gauge where sys-tems are located and what areas of compliance you need to build into your system.

Table 12.1 lists some of the different service models and lists the parties responsible for security in the different instances.

TABLE 12.1

Security Responsibilities by Service Model

Model Type	Infrastructure Security Management	Infrastructure Owner	Infrastructure Location	Trust Condition
Hybrid	Both vendor and customer	Both vendor and customer	Both on- and off-premises	Both trusted and untrusted
Private/Community	Customer	Customer	On- or off-premises	Trusted
Private/Community	Customer	Vendor	Off- or on-premises	Trusted
Private/Community	Vendor	Customer	On- or off-premises	Trusted
Private/Community	Vendor	Vendor	Off- or on-premises	Trusted
Public	Vendor	Vendor	Off-premises	Untrusted

Security mapping

The cloud service model you choose determines where in the proposed deployment the variety of security features, compliance auditing, and other requirements must be placed. To determine the particular security mechanisms you need, you must perform a mapping of the particular cloud service model to the particular application you are deploying. These mechanisms must be supported by the various controls that are provided by your service provider, your organization, or a third party. It's unlikely that you will be able to duplicate security routines that are possible on-premises, but this analysis allows you to determine what coverage you need.

A security control model includes the security that you normally use for your applications, data, management, network, and physical hardware. You may also need to account for any compliance standards that are required for your industry. A compliance standard can be any government regulatory framework such as Payment Card Industry Data Security Standards (PCI-DSS), Health Insurance Portability and Accountability Act (HIPPA), Gramm–Leach–Bliley Act (GLBA), or the Sarbanes–Oxley Act (SOX) that requires you operate in a certain way and keep records.

Essentially, you are looking to identify the missing features that would be required for an on-premises deployment and seek to find their replacements in the cloud computing model. As you assign accountability for different aspects of security and contract away the operational responsibility to others, you want to make sure they remain accountable for the security you need.

Securing Data

Securing data sent to, received from, and stored in the cloud is the single largest security concern that most organizations should have with cloud computing. As with any WAN traffic, you must assume that any data can be intercepted and modified. That's why, as a matter of course, traffic to a cloud service provider and stored off-premises is encrypted. This is as true for general data as it is for any passwords or account IDs.

These are the key mechanisms for protecting data mechanisms:

- Access control
- Auditing
- Authentication
- Authorization

Whatever service model you choose should have mechanisms operating in all four areas that meet your security requirements, whether they are operating through the cloud service provider or your own local infrastructure.

Brokered cloud storage access

The problem with the data you store in the cloud is that it can be located anywhere in the cloud service provider's system: in another datacenter, another state or province, and in many cases even in another country. With other types of system architectures, such as client/server, you could count on a firewall to serve as your network's security perimeter; cloud computing has no physical system that serves this purpose. Therefore, to protect your cloud storage assets, you want to find a way to isolate data from direct client access.

One approach to isolating storage in the cloud from direct client access is to create layered access to the data. In one scheme, two services are created: a broker with full access to storage but no access to the client, and a proxy with no access to storage but access to both the client and broker. The location of the proxy and the broker is not important (they can be local or in the cloud); what is important is that these two services are in the direct data path between the client and data stored in the cloud.

Under this system, when a client makes a request for data, here's what happens:

1. The request goes to the external service interface (or endpoint) of the proxy, which has only a partial trust.
2. The proxy, using its internal interface, forwards the request to the broker.
3. The broker requests the data from the cloud storage system.
4. The storage system returns the results to the broker.
5. The broker returns the results to the proxy.
6. The proxy completes the response by sending the data requested to the client.

Figure 12.4 shows this storage "proxy" system graphically.

Note

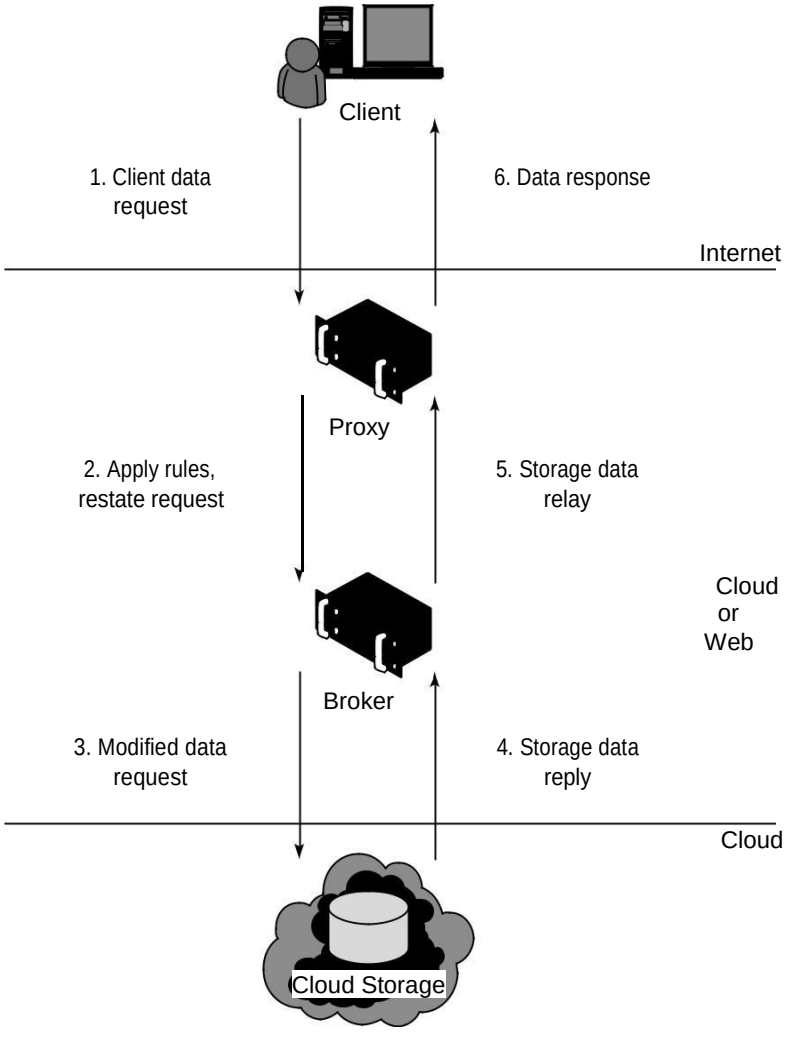
This discussion is based on a white paper called "Security Best Practices For Developing Windows Azure Applications," by Andrew Marshall, Michael Howard, Grant Bugher, and Brian Harden that you can find at http://download.microsoft.com/download/7/3/E/73E4EE93-559F-4D0F-A6FC-7FEC5F1542D1/SecurityBestPracticesWindowsAzureApps.docx**. In their presentation, the proxy service is called the Gatekeeper and assigned a Windows Server Web Role, and the broker is called the KeyMaster and assigned a Worker Role.** ■

This design relies on the proxy service to impose some rules that allow it to safely request data that is appropriate to that particular client based on the client's identity and relay that request to the broker. The broker does not need full access to the cloud storage, but it may be configured to grant READ and QUERY operations, while not allowing APPEND or DELETE. The proxy has a limited trust role, while the broker can run with higher privileges or even as native code.

The use of multiple encryption keys can further separate the proxy service from the storage account. If you use two separate keys to create two different data zones—one for the untrusted communication between the proxy and broker services, and another a trusted zone between the broker and the cloud storage—you create a situation where there is further separation between the different service roles.

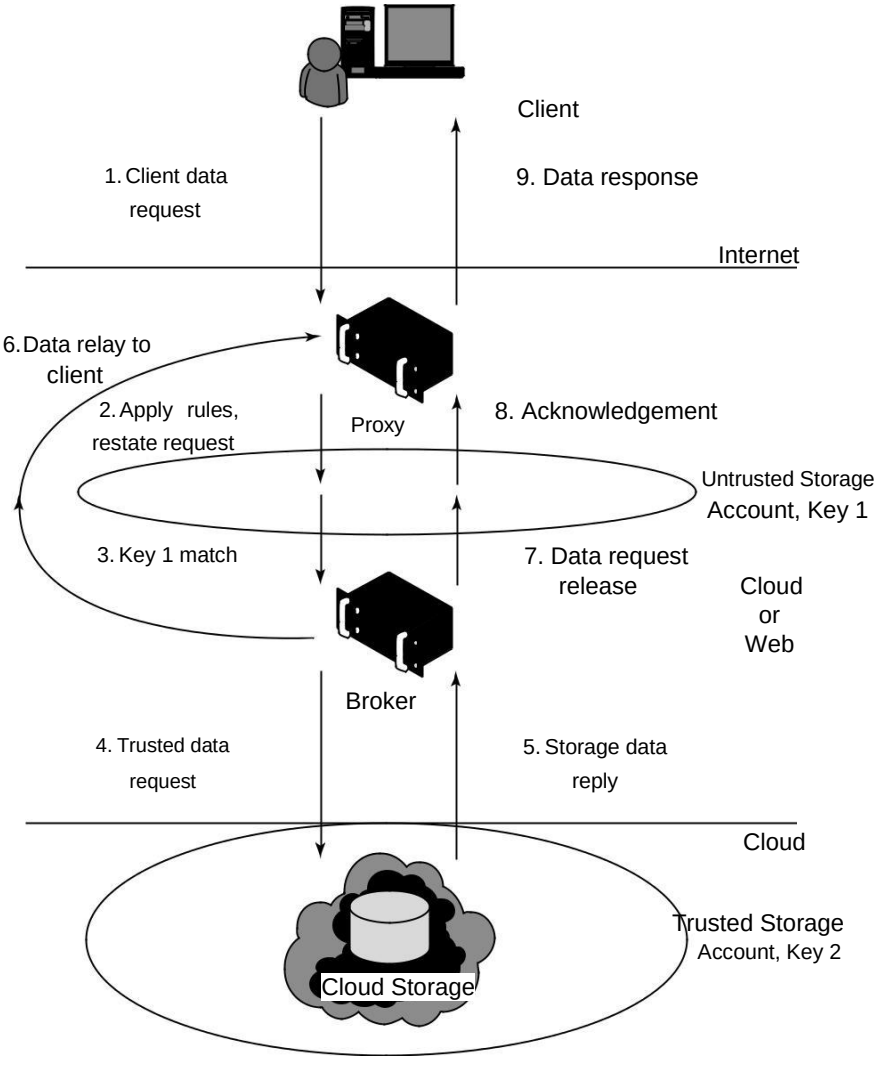
FIGURE 12.4

In this design, direct access to cloud storage is eliminated in favor of a proxy/broker service.



Even if the proxy service is compromised, that service does not have access to the trusted key nec-essary to access the cloud storage account. In the multi-key solution, shown in Figure 12.5, you have not only eliminated all internal service endpoints, but you also have eliminated the need to have the proxy service run at a reduced trust level.

The creation of storage zones with associated encryption keys can further protect cloud storage from unauthorized access.



Storage location and tenancy

Some cloud service providers negotiate as part of their Service Level Agreements to contractually store and process data in locations that are predetermined by their contract. Not all do. If you can

get the commitment for specific data site storage, then you also should make sure the cloud vendor is under contract to conform to local privacy laws.

Because data stored in the cloud is usually stored from multiple tenants, each vendor has its own unique method for segregating one customer's data from another. It's important to have some understanding of how your specific service provider maintains data segregation.

Another question to ask a cloud storage provider is who is provided privileged access to storage. The more you know about how the vendor hires its IT staff and the security mechanism put into place to protect storage, the better.

Most cloud service providers store data in an encrypted form. While encryption is important and effective, it does present its own set of problems. When there is a problem with encrypted data, the result is that the data may not be recoverable. It is worth considering what type of encryption the cloud provider uses and to check that the system has been planned and tested by security experts.

Regardless of where your data is located, you should know what impact a disaster or interruption will have on your service and your data. Any cloud provider that doesn't offer the ability to replicate data and application infrastructure across multiple sites cannot recover your information in a timely manner. You should know how disaster recovery affects your data and how long it takes to do a complete restoration.

Encryption

Strong encryption technology is a core technology for protecting data in transit to and from the cloud as well as data stored in the cloud. It is or will be required by law. The goal of encrypted cloud storage is to create a virtual private storage system that maintains confidentiality and data integrity while maintaining the benefits of cloud storage: ubiquitous, reliable, shared data storage. Encryption should separate stored data (data at rest) from data in transit.

Depending upon the particular cloud provider, you can create multiple accounts with different keys as you saw in the example with Windows Azure Platform in the previous section. Microsoft allows up to five security accounts per client, and you can use these different accounts to create different zones. On Amazon Web Service, you can create multiple keys and rotate those keys dur-ing different sessions.

Although encryption protects your data from unauthorized access, it does nothing to prevent data loss. Indeed, a common means for losing encrypted data is to lose the keys that provide access to the data. Therefore, you need to approach key management seriously. Keys should have a defined lifecycle. Among the schemes used to protect keys are the creation of secure key stores that have restricted role-based access, automated key stores backup, and recovery techniques. It's a good idea to separate key management from the cloud provider that hosts your data.

One standard for interoperable cloud-based key management is the OASIS Key Management Interoperability Protocol (KMIP; http://www.oasis-open.org/committees/kmip/). IEEE 1619.3 (https://siswg.net/index.php?option=com_docman) also covers both storage encryption and key management for shared storage.

Auditing and compliance

Logging is the recording of events into a repository; auditing is the ability to monitor the events to understand performance. Logging and auditing is an important function because it is not only necessary for evaluation performance, but it is also used to investigate security and when illegal activity has been perpetrated. Logs should record system, application, and security events, at the very minimum.

Logging and auditing are unfortunately one of the weaker aspects of early cloud computing service offerings.

Cloud service providers often have proprietary log formats that you need to be aware of. Whatever monitoring and analysis tools you use need to be aware of these logs and able to work with them. Often, providers offer monitoring tools of their own, many in the form of a dashboard with the potential to customize the information you see through either the interface or programmatically using the vendor's API. You want to make full use of those built-in services.

Because cloud services are both multitenant and multisite operations, the logging activity and data for different clients may not only be co-located, they may also be moving across a landscape of different hosts and sites. You can't simply expect that an investigation will be provided with the necessary information at the time of discovery unless it is part of your Service Level Agreement. Even an SLA with the appropriate obligations contained in it may not be enough to guarantee you will get the information you need when the time comes. It is wise to determine whether the cloud ser-vice provider has been able to successfully support investigations in the past.

As it stands now, nearly all regulations were written without keeping cloud computing in mind. A regulator or auditor isn't likely to be familiar with the nature of running applications and storing data in the cloud. Even so, laws are written to ensure compliance, and the client is held responsible for compliance under the laws of the governing bodies that apply to the location where the pro-cessing or storage takes place.

Therefore, you must understand the following:

- Which regulations apply to your use of a particular cloud computing service
- Which regulations apply to the cloud service provider and where the demarcation line falls for responsibilities
- How your cloud service provider will support your need for information associated with regulation
- How to work with the regulator to provide the information necessary regardless of who had the responsibility to collect the data

Traditional service providers are much more likely to be the subject of security certifications and external audits of their facilities and procedures than cloud service providers. That makes the willingness for a cloud service provider to subject its service to regulatory compliance scrutiny an important factor in your selection of that provider over another. In the case of a cloud service provider who shows reluctance to or limits the scrutiny of its operations, it is probably wise to use the

service in ways that limit your exposure to risk. For example, although encrypting stored data is always a good policy, you also might want to consider not storing any sensitive information on that provider's system.

As it stands now, clients must guarantee their own regulatory compliance, even when their data is in the care of the service provider. You must ensure that your data is secure and that its integrity has not been compromised. When multiple regulatory entities are involved, as there surely are between site locations and different countries, then that burden to satisfy the laws of those governments is also your responsibility.

For any company with clients in multiple countries, the burden of regulatory compliance is onerous. While organizations such as the EEC (European Economic Community) or Common Market provide some relief for European regulation, countries such as the United States, Japan, China, and others each have their own sets of requirements. This makes regulatory compliance one of the most actively developing and important areas of cloud computing technology.

This situation is likely to change. On March 1, 2010, Massachusetts passed a law that requires companies that provide sensitive personal information on Massachusetts residents to encrypt data transmitted and stored on their systems. Businesses are required to limit the amount of personal data collected, monitor data usage, keep a data inventory, and be able to present a security plan on how they will keep the data safe. The steps require that companies verify that any third-party services they use conform to these requirements and that there be language in all SLAs that enforce these protections. The law takes full effect in March 2012.

Going forward, you want to ensure the following:

- You have contracts reviewed by your legal staff.
- You have a right-to-audit clause in your SLA.
- You review any third parties who are service providers and assess their impact on security and regulatory compliance.
- You understand the scope of the regulations that apply to your cloud computing applications and services.
- You consider what steps you must take to comply with the demands of regulations that apply.
- You consider adjusting your procedures to comply with regulations.
- You collect and maintain the evidence of your compliance with regulations.
- You determine whether your cloud service provider can provide an audit statement that is SAS 70 Type II-compliant.

The ISO/IEC 27001/27002 standard for information security management systems has a roadmap for mission-critical services that you may want to discuss with your cloud service provider. Amazon Web Services supports SAS70 Type II Audits.

Becoming a cloud service provider requires a large investment, but as we all know, even large companies can fail. When a cloud service provider fails, it may close or more likely be acquired by another company. You likely wouldn't use a service provider that you suspected of being in difficulty, but problems develop over years and cloud computing has a certain degree of vendor lock-in to it. That is, when you have created a cloud-based service, it can be difficult or often impossible to move it to another service provider. You should be aware of what happens to your data if the cloud service provider fails. At the very least, you would want to make sure your data could be obtained in a format that could be accessed by on-premise applications.

The various attributes of cloud computing make it difficult to respond to incidents, but that doesn't mean you should consider drawing up security incidence response policies. Although cloud computing creates shared responsibilities, it is often up to the client to initiate the inquiry that gets the ball rolling. You should be prepared to provide clear information to your cloud ser-vice provider about what you consider to be an incident or a breach in security and what are sim-ply suspicious events.

Questions

1. Explain the types of services required in implementing the cloud infrastructure in terms of Consulting, Configuration, Customization and Support in cloud infrastructure.
2. Explain the features of network management systems. Briefly introduce the related products from large cloud vendors.
3. Discuss briefly the monitoring of an entire cloud computing deployment stack. Mention some products used in cloud computing deployment stack.
4. Explain the lifecycle management of cloud services (six stages of lifecycle).
5. Explain the need of live migration of virtual machine. Design a process of live migration.
6. Discuss the security issues during live migration.

7.

8. Discuss the infrastructure security by throwing light on the following:
    i. Network Level
    ii. Host Level
    iii. Application Level
    iv. Data Security and Storage
    v. Aspects of Data Security
    vi. Data Security Mitigation Provider Data and Its Security
    vii. Identity and Access Management
9. Discuss the auditing and compliance in cloud environment in terms of following:
    i. Data Security in Cloud Computing Environment
    ii. Need for Auditing in Cloud Computing Environment
    iii. Third Party Service Provider
    iv. Cloud Auditing Outsourcing Lifecycle Phases
    v. Auditing Classification

Module – IV

Concepts of Services and Applications

Understanding Service Oriented Architecture

Service Oriented Architecture (SOA) describes a standard method for requesting services from distributed components and managing the results. Because the clients requesting services, the components providing the services, the protocols used to deliver messages, and the responses can vary widely, SOA provides the translation and management layer in an architecture that removes the barrier for a client obtaining desired services. With SOA, clients and components can be written in different languages and can use multiple messaging protocols and networking protocols to commu-nicate with one another. SOA provides the standards that transport the mes-sages and makes the infrastructure to support it possible. SOA provides access to reusable Web services over a TCP/IP network, which makes this an important topic to cloud computing going forward.

You don't need SOA if you are creating a monolithic cloud application that performs a specific function such as backup, e-mail, Web page access, or instant messaging. Many of the large and familiar cloud computing applica-tions are monolithic and were built with proprietary technologies—albeit often on top of open source software and hardware. However, as cloud computing applications expand their capability to provide additional and diverse services, SOA offers access to ready-made, modular, highly optimized, and widely shareable components that can minimize developer and infrastructure costs.

For over a decade now Service Oriented Architecture has been part of a col-laborative effort on the part of both large and small vendors to come up with a common solution to architecting complex business software processes effi-ciently. As cloud computing matures and the applications offered become more capable, the key to being competitive and offering users the capability to customize their environments lays in the standardization that Service Oriented Architecture offers. The influence of SOA in cloud computing is therefore likely to grow.

This chapter provides the basis for understanding what SOA is, how SOA operates, what limitations and capabilities are part of the architecture, and the vocabulary that you need to know in this subject. SOA is an architecture, first and foremost, so in essence it is a blueprint for creating a system conforming to this standard. The environment it creates is a virtual message-passing system with a loose coupling between clients and services. The products that support SOA are a diverse lot. SOA components are meant to be modular and easily added to a business process, and this modularity makes them good candidates for Computer Aided Software Engineering (CASE) model-ing tools. This chapter describes the software used to support SOA and some of the important ways in which the architecture is interpreted.

Cloud computing is not the next evolutionary step beyond SOA, as some think because of SOA's longer history. The two technologies are complementary. Whereas SOA can be used to construct large and complex applications that scale both horizontally and vertically, cloud computing appli-cations tend to be scaled vertically. Horizontal scaling refers to applications with a large number of different business processes operating. Vertical scaling refers to large applications with a limited number of business processes operating. SOA techniques may be applied in both instances.

Introducing Service Oriented Architecture

Service Oriented Architecture (SOA) is a specification and a methodology for providing platform- and language-independent services for use in distributed applications. A service is a repeatable task within a business process, and a business task is a composition of services. SOA describes a mes-sage-passing taxonomy for a component-based architecture that provides services to clients upon demand. Clients access a component that complies with SOA by passing a message containing metadata to be acted upon in a standard format. The component acts on that message and returns a response that the client then uses for its own purpose. A common example of a message is an XML file transported over a network protocol such as SOAP.

Usually service providers and service consumers do not pass messages directly to each other. Implementations of SOA employ middleware software to play the role of transaction manager (or broker) and translator. That middleware can discover and list available services, as well as potential service consumers, often in the form of a registry, because SOA describes a distributed architecture security and trust services are built directly into many of these products to protect communication. Middleware products also can be where the logic of business processes reside; they can be general-purpose applications, industry-specific, private, or public services.

Middleware services manage lookup requests. The Universal Description Discovery and Integration (UDDI) protocol is the one most commonly used to broadcast and discover available Web services, often passing data in the form of an Electronic Business using eXtensible Markup Language (ebXML) documents. Service consumers find a Web service in a broker registry and bind their ser-vice requests to that specific service; if the broker supports several Web services, it can bind to any of the ones that are useful.

This architecture does not contain executable links that require access to a specific API. The mes-sage presents data to the service, and the service responds. It is up to the client to determine if the service returned an appropriate result. An SOA is then seen as a method for creating an integrated process as a set of linked services. The component exposes itself as an "endpoint" (a term of art in SOA) to the client.

The most commonly used message-passing format is an Extensible Markup Language (XML) docu-ment using Simple Object Access Protocol (SOAP), but many more are possible, including Web Services Description Language (WSDL), Web Services Security (WSS), and Business Process Execution Language for Web Services (WS-BPEL). WSDL is commonly used to describe the service interface, how to bind information, and the nature of the component's service or endpoint. The Service Component Definition Language (SCDL) is used to define the service component that per-forms the service, providing the component service information that is not part of the Web service and that therefore wouldn't be part of WSDL.

Note

Whatever protocol is used to negotiate a transaction, the formal definition of the transaction is referred to as the "contract." Indeed, the notion of a contract implies a certain level of service that is available to clients and that may be part of any paid service in SOA. ■

Figure 13.1 shows a protocol stack for an SOA architecture and how those different protocols execute the functions required in the Service Oriented Architecture. In the figure, the box labeled Other Services could include Common Object Request Broker Architecture (CORBA), Representational State Transfer (REST), Remote Procedure Calls (RPC), Distributed Common Object Model (DCOM), Jini, Data Distribution Service (DDS), Windows Communication Foundation (WCF), and other technologies and protocols. It is this flexibility and neutrality that makes SOA so singularly useful in designing complex applications. These services and the manner in which they interact in regards to SOA have been codified by a number of standards organiza-tions, and some of the more prominent efforts are described later in this chapter.

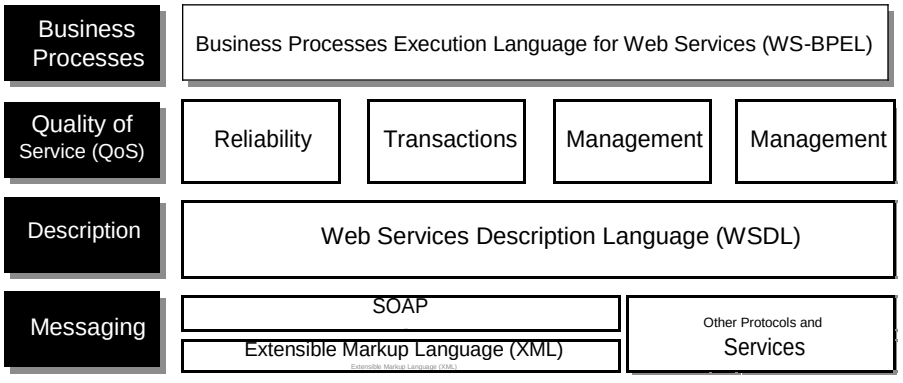
Tip

To read the IBMs SOA Foundation White Paper, see http://download.boulder.ibm.com/ibmdl/pub/software/dw/webservices/ws-soa-whitepaper.pdf**. ■**

SOA provides the framework needed to allow clients of any type to engage in a request-response mechanism with a service. The specification of the manner in which messages are passed in SOA, or in which events are handled, are referred to as their *contract.* The term is meant to imply that the client engages the service in a task that must be managed in a specified manner. In real systems, contracts may specifically be stated with a Quality of Service parameter in a real paper contract. Typically, SOA requires the use of an orchestrator or broker service to ensure that messages are correctly transacted. SOA makes no other demands on either the client (consumer) or the compo-nents (provider) of the service; it is concerned only with the interface or action boundary between the two. This is the earliest definition of SOA architecture.

FIGURE 13.1

A protocol stack for SOA showing the relationship of each protocol to its function



Components are often written to comply with the Service Component Architecture (SCA), a language- and technology-agnostic design specification that has wide, but not universal, industry support. SCA can use the services of components that are written in the Business Process Execution Language (BPEL), Java, C#/.NET, XML, or Cobol, and can apply to C++ and Fortran, as well as to the dynamic languages Python, Ruby, PHP, and others. This allows components to be written in the easiest form that supports the business process that the component is meant to service. By wrapping data from legacy clients written in languages such as COBOL, SOA has greatly extended the life of many legacy applications.

Tip

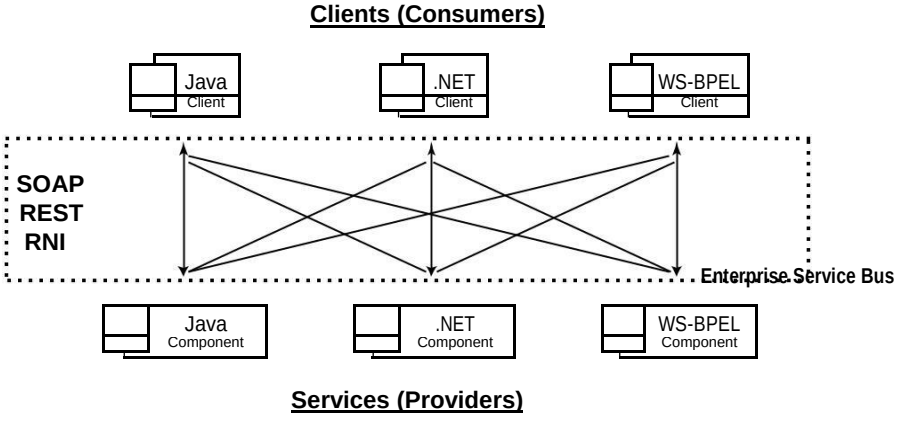
To read David Chappel's white paper on SCA, go to http://www.davidchappell.com/articles/Introducing_SCA.pdf**.** ■

Components are coded with their service logic and their dependencies, QoS is established, and the service is instantiated. In the SCA model, data and messages are exchanged in a Service Data Object (SDO). This system of messaging using objects and services is sometimes referred to as a Data Access Service (DAS). Figure 13.2 shows how components of different types can communi-cate using different protocols as part of SOA.

When you combine Web services to create business processes, the integration must be managed. Two main methods are used to combine Web services: orchestration and choreography. In orchestra-tion, a middleware service centrally coordinates all the different Web service operations, and all ser-vices send messages and receive messages from the orchestrator. The logic of the compound business process is found at the orchestrator alone. Figure 13.3 shows how orchestration is managed.

FIGURE 13.2

SOA allows for different component and client construction, as well as access to each using different protocols.



By contrast, a compound business process that uses choreography has no central coordination function. In choreography, each Web service that is part of a business process is aware of when to process a message and with what client or component it needs to interact with. Choreography is a collaborative effort where the logic of the business process is pushed out to the members who are responsible for determining which operations to execute and when to execute them, the structure of the messages to be passed and their timing, and other factors. Figure 13.4 illustrates the nature of choreography.

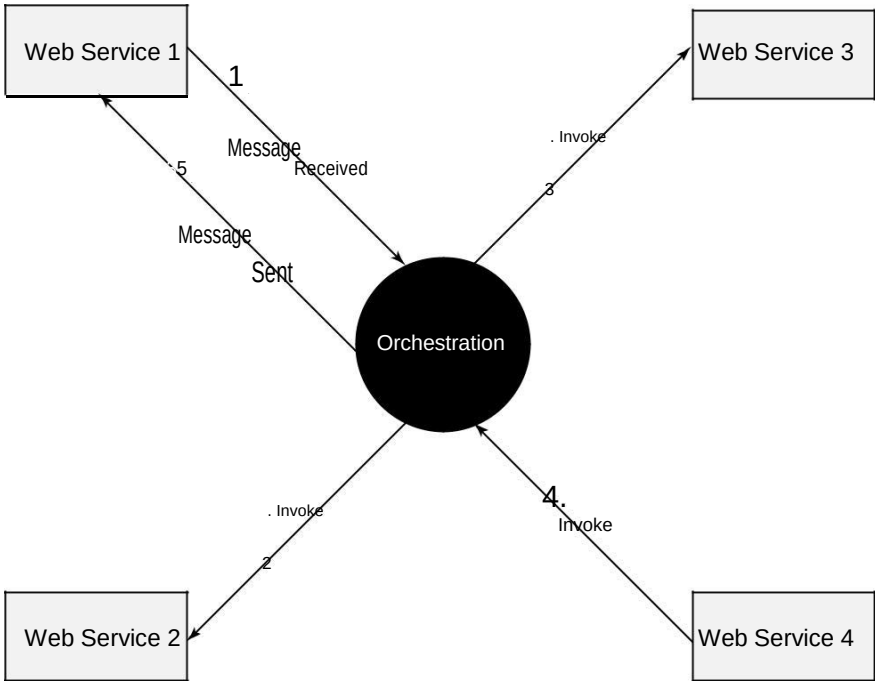
What isn't clear from Figure 13.2, but is shown in Figure 13.3 (orchestration) and Figure 13.4 (choreography) is that business processes are conducted using a sequence, in parallel, or simply by being invoked (called to). An execution language like WS-BPEL provides commands for defining logic using conditional statements, loops, variables, fault handlers, and other constructs. Because a business process is a collection of activity graphs, complex processes are often shown as part of Unified Modeling Language (UML) diagrams. UML is the modeling language of the Object Management Group that provides a method for creating visual models for software in the form of 14 types of diagrams. Some of the diagram types are structure, behavior, class, component, object, interaction, state, and sequence.

Tip

You can find a primer on BPEL by Matjaz Juric on Oracle's Web site at http://www.oracle.com/technology/pub/articles/matjaz_bpel1.html**. For information on SOA modeling, refer to *Service-Oriented Modeling: Analysis, Design, and Architecture,* by Michael Bell, Wiley, 2008, and Bell's later book *SOA Modeling Patterns for Service-Oriented Discovery and Analysis,* Wiley, 2010.** ■

FIGURE 13.3

An orchestrated business process uses a central controlling service or element, referred to as the orchestrator, conductor, or less frequently, the coordinator.

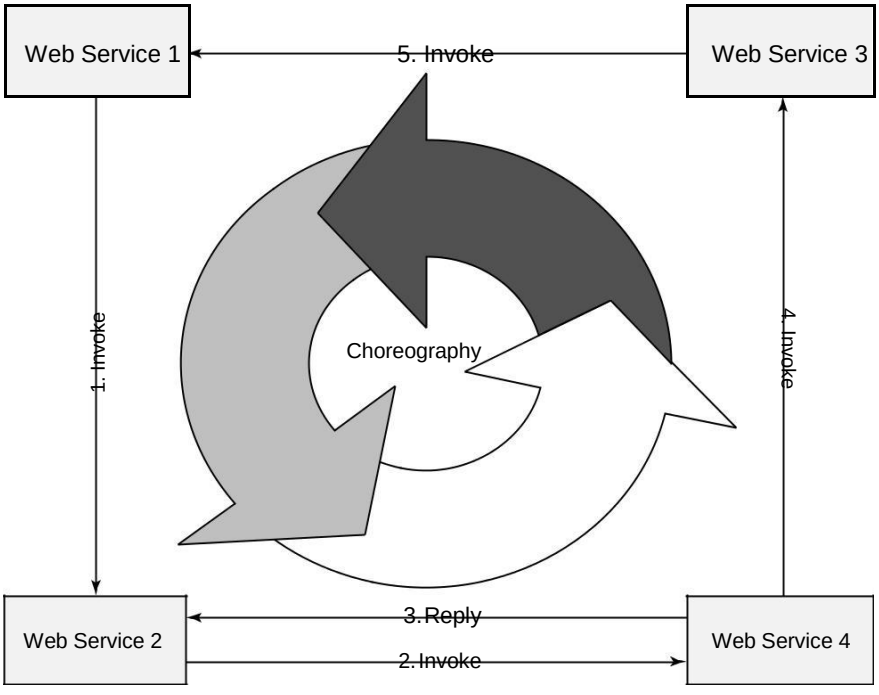


Most mature SOA implementations favor orchestration over choreography for a number of reasons. With orchestration a single central service manages the various processes, and changes to the busi-ness logic can be made in that one location. The integration of Web services into the architecture is easier than with choreography because these services don't need to know anything about the busi-ness process. Centralizing the business logic also makes it easier to put error handling mechanisms in place and to account for, manage, and analyze events that occur outside the business process that relate to a part of the process. Event handling is part of event-driven SOA or SOA 2.0, which extends Service Oriented Architecture to include both random and scheduled events that are trig-gered by a business process outside of a business process.

One way of performing orchestration is through the use of an Enterprise Service Bus or ESB. An ESB provides a middleware software layer for event management with a messaging infrastructure. ESBs are described later in the section called "The Enterprise Service Bus." An ESB isn't required by SOA, but it is often used to create a compliant and efficient service architecture.

FIGURE 13.4

With choreography, business process execution is a cooperative affair.



Event-driven SOA or SOA 2.0

Event-driven SOA or SOA 2.0 is an extension of the Service Oriented Architecture to respond to events that occur as a result of business processes or perhaps cause and influence a business process. For example, in a business process, sales at a certain Web site are processed. If the business process recognizes the rate at which sales are occurring, it could perform an analysis to determine what events might influence the buying decision. This is the sort of analysis that event-driven SOA is meant to address. SOA 2.0 can allow low-level events to trigger a business process, correlate events with information contained in the SOA design, inhibit a business process if the appropriate events don't appear, or invoke a reaction or response based on a trigger.

To perform these tasks in SOA 2.0, a Causal Vector Engine (CVE) with some built-in artificial intelligence must be added to the SOA design. Events are analyzed in terms of event sequences, event relationships, and event timing to establish whether a certain condition has occurred. The CVE then determines how to react to the condition using a set of rules that are built into the system. Many CVE systems display events in a console in different contexts so that an observer can

analyze the display and take appropriate actions. A CVE application may include the ability to query event data in the same way that a stock ticker or trading application can query trading data. The CVE application provides the same kind of heartbeat and correlation functionality that a stock trading application does.

From the standpoint of the service requestor or consumer (client), the client simply needs to know the form required to initiate the action of the provider (service) and how to interpret the results returned from the service provider. The nature of the component's processing is unknown, the location where the processing is done is unknown, and the various operating systems and applications involved are unknown. The client is responsible for validating that the service returned the results that were expected. The SOA component is essentially a black box to the client. That is, SOA makes no demands of the component other than to conform to the rules of a standard endpoint. This level of abstraction offers operational advantages to Web service providers in that com-ponents can be continually upgraded, replaced, or moved to improve efficiencies without disrupting the clients that depend on those services, and the Quality of Service for that service can be accurately measured and delivered. In SOA, the service has been virtualized.

Cross-Ref

Communication protocols are discussed in detail in Chapter 3, "Understanding Cloud Architecture." ■

Any network service that spans different application types is a candidate for componentization. Consider a logon or authentication system. It would be wasteful to implement the same authenti-cation functionality in several different applications when a single unified module could serve the same purpose. A single sign-on system for an accounting package, payroll module, or production database replaces three separate modules with attendant efficiencies in the amount of code that needs to be written and managed and the level of system overhead that is involved. SOA provides the rules so each application can access the authentication module in its own way, as required. What you gain with SOA is the ability to add significant capabilities with a fraction of the cost or effort and to federate applications if you desire. What you lose with SOA is the ability to perform fundamental customization of the service itself when that service is provided by a third party.

The Enterprise Service Bus

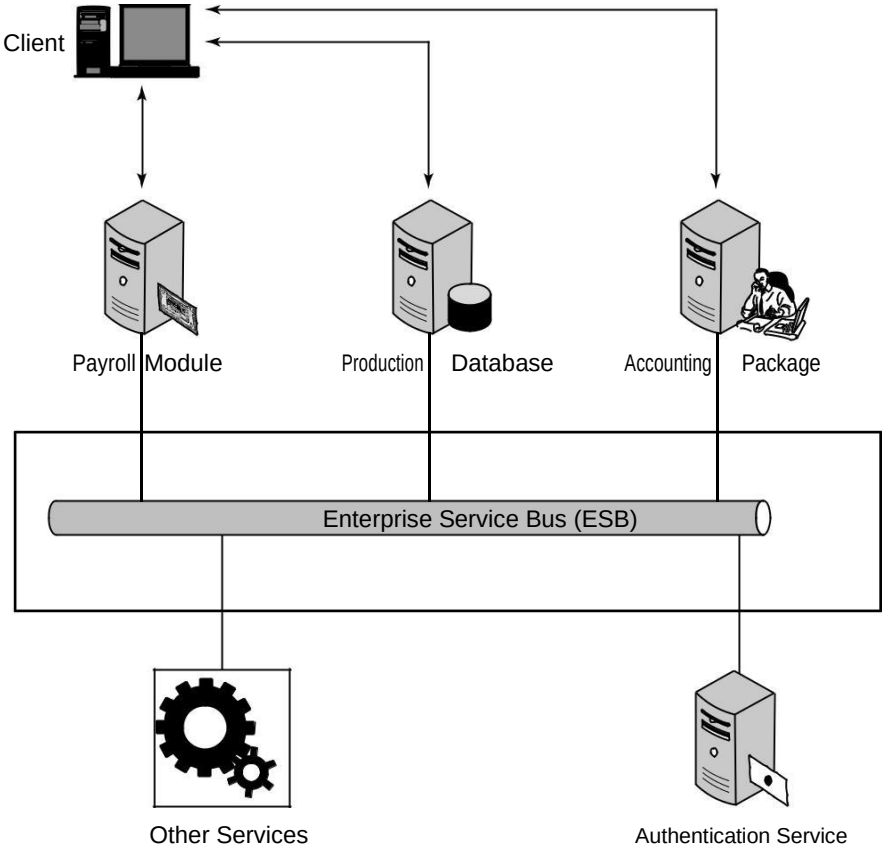
In Figure 13.5, those aforementioned hypothetical three different applications are shown interfaced with an authentication module through what has come to be called an Enterprise Service Bus (ESB). An ESB is not a physical bus in the sense of a network; rather, it is an architectural pattern comprised of a set of network services that manage transactions in a Service Oriented Architecture.

You may prefer to think of an ESB as a set of services that separate clients from components on a transactional basis and that the use of the word *bus* in the name indicates a high degree of connectivity or fabric quality to the system; that is, the system is *loosely coupled.* Messages flow from client to component through the ESB, which manages these transactions, even though the location of the services comprising the ESB may vary widely.

An ESB is necessary but not essential to a Service Oriented Architecture because typical business processes can span a vast number of messages and events, and distributed processing is an inherently unreliable method of transport. An ESB therefore plays the role of a transaction broker in SOA, ensuring that messages get to where they where supposed to go and are acted upon properly. The service bus performs the function of mediation: message translation, registration, routing, logging, auditing, and managing transactional integrity. Transactional integrity is similar to ACID in a database system—atomicity, consistency, isolation, and durability, the essence of which is that transactions succeed or they fail and are rolled back.

FIGURE 13.5

An SOA application of a shared logon or Authentication module



An ESB may be part of a network operating system or may be implemented using a set of middle-ware products. An ESB creates a virtual environment layered on top of an enterprise messaging system where services are advertised and accessed. Think of an ESB as a message transaction system. IBM's WebSphere ESB 7.0 is an ESB based on open standards such as Java EE, EJB, WS-Addressing, WS-Policy, and Kerberos security, and it runs on the WebSphere Application Server. It is interoperable with Open SCA. WebSphere ESB contains both a Service Federation Management tool and an integrated Registry and Repository function.

These typical features are found in ESBs, among others:

- **Monitoring services** aid in managing events.
- **Process management services** manage message transactions.
- **Data repositories or registries** store business logic and aid in governance of business processes.
- **Data services** pass messages between clients and services.
- **Data abstraction services** translate messages from one format to another, as required.
- **Governance** is a service that monitors compliance of your operations with governmental regulation, which can vary from state to state and from country to country.
- **Security services** validate clients and services and allow messages to pass from one to the other.

Figure 13.6 shows how these different services in an SOA relate to one another.

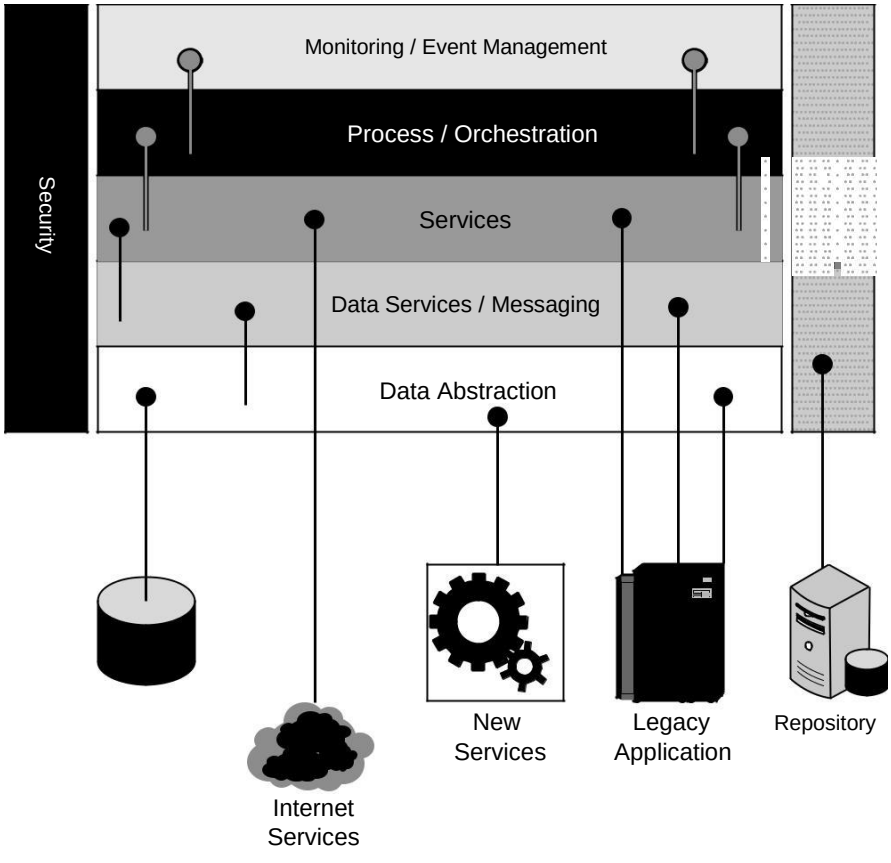
The difference between a repository and a registry in the context of a Service Oriented Architecture is subtle. Repositories and registries are both data stores, but a repository stores references to the components of the SOA, their source code, and linking information that are used to provide SOA services. An SOA registry contains references to rules, descriptions, and definitions of the ser-vices—that is, the metadata of the components.

A repository serves the role that a name server does in a network operating system infrastructure, while the registry plays the role of a directory service (domain). The service broker uses the rules contained in the SOA registry to perform its function as translator and delivery agent. For develop-ers, the registry serves as the central location to store component descriptions that allow composite applications to be created and the place in which services may be published for general use.

These services in an SOA also include the provider interfaces and standard sets of network proto-cols that were mentioned previously. Developers may also choose to create a Business Process Orchestration module to coordinate the access and transactional integrity of multiple business applications that integrate into a larger platform, described in the next section in more detail.

FIGURE 13.6

This figure shows a network services model infrastructure for an SOA, which is based on the SOA meta-model of the Linthicum Group, 2007. A slightly different version of this diagram appears in *Networking Bible* by Barrie Sosinsky, Wiley, 2009.



Service catalogs

Finding any particular service and locating the service's requirement in a large SOA implementa-tion can involve a large amount of network system overhead. To aid in locating services, SOA infrastructure often includes a catalog service. This service stores information on the following, among other things:

- What services are available, both internal and external
- How to use a service
- Which applications are related to a particular service (dependencies)
- How services relate to one another
- Who owns the service and how a service is modified
- The event history of a service, including service levels, outages, and so on
- The nature of service contracts

Service catalogs are dynamic and under constant modification. Catalog servers have these features:

- They can be **standalone catalog servers** serving a single site.
- They serve the role of a **global catalog service** where two or more catalog servers are merged to include several sites. A global service usually requires some sort of synchroniza-tion or update to maintain a unified data store across the servers involved.
- They can be part of a **federated catalog service** where two or more global catalog servers have access to one another's information through a trusted query relationship.

Catalog services have an enormous impact on large system performance and eventually become essential as a SOA internetwork system grows. An internetwork is a network that is constructed through the consolidation of separate networks, in the same manner that the Internet has been built.

Note

The Information Technology Infrastructure Library (ITIL) developed by the United Kingdom's Office of Government Commerce (OGC) has developed a Service Catalog Management design specification as part of the ITIL v2 Service Design Package (SDP). Service Designs present best practice guidance for planned services. The ITIL Web site may be found at http://www.itil-officialsite.com/home/home.asp**.** ■

Defining SOA Communications

Message passing in SOA requires the use of two different protocol types: the data interchange for-mat and the network protocol that carries the message. A client (or customer) connected to an ESB communicates over a network protocol such as HTTP, Representational State Transfer (REST), or Java Message Service (JMS) to a component (or service). Messages are most often in the form of the eXtensible Markup Language (XML) or in a variant such as the Simple Object Access Protocol (SOAP). SOAP is a messaging format used in Web services that use XML as the message format while relying on Application layer protocols such as HTTP and Remote Procedure Calls (RPC) for message negotiation and transmission.

The software used to write clients and components can be written in Java, .NET, Web Service Business Process Execution Language (WS-BPEL), or another form of executable code; the services that they message can be written in the same or another language. What is required is the ability to transport and translate a message into a form that both parties can understand.

An ESB may require a variety of combinations in order to support communications between a service consumer and a service provider. For example, in WebSphere ESB, you might see the following combinations:

- XML/JMS (Java Message Service)
- SOAP/JMS
- SOAP/HTTP
- Text/JMS
- Bytes/JMS

The Web Service Description Language (WSDL) is one of the most commonly used XML protocols for messaging in Web services, and it finds use in Service Oriented Architectures. Version 1.1 of WSDL is a W3C standard, but the current version WSDL 2.0 (formerly version 1.2) has yet to be ratified by the W3C. The significant difference between 1.1 and 2.0 is that version 2.0 has more support for RESTful (e.g. Web 2.0) application, but much less support in the current set of soft-ware development tools. The most common transport for WSDL is SOAP, and the WSDL file usu-ally contains both XML data and an XML schema.

REST offers some very different capabilities than SOAP. With REST, each URL is an object that you can query and manipulate. You use HTML commands such as GET, POST, PUT, and DELETE to work with REST objects. SOAP uses a different approach to working with Web data, exposing Web objects through an API and transferring data using XML. The REST approach offers light-weight access using standard HTTP command, is easier to implement than SOAP, and comes with less overhead. SOAP is often more precise and provides a more error-free consumption model. SOAP often comes with more sophisticated development tools. All major Web services use REST, but many Web services, especially newer ones, combine REST with SOAP to derive the benefits that both offer.

Contained within WSDL are essential objects to support message transfer, including these:

- The **service** object, a container where the service resides.
- The **port or endpoint,** which is the unique address of the service.
- The **binding**, which is the description of the interface (e.g. RPC) and the transport (e.g. SOAP).
- The **portType**, or interface that defines the capabilities of the Web service, and what operations are to be performed, as well as the messages that must be sent to support the operation.
- The **operation** that is to be performed on the message.

- The **message** content, which is the data and metadata that the service operation is performed on. Each message may consist of one or more parts, and each part must include typing information.
- The **types** used to describe the data, usually as part of the XML schema that accompanies the WSDL.

Business Process Execution Language

If a message represents an atomic transaction in a Service Oriented Architecture, the next level of abstraction up is the grouping and managing of sets of transactions to form useful work and to execute a business process. An example of an execution language is the Business Process Execution Language (BPEL) or alternatively as the Web Service Business Process Execution Language

(WS-BPEL), a language standard for Web service interactions. The standard is maintained by the **O**rganization for the **A**dvancement of **S**tructured **I**nformation **S**tandards (OASIS) through their Web Services Business Process Execution Language Technical Committee (WSBPEL-TC; see http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wsbpel.)

BPEL is a meta-language comprised of two functions: executable commands for Web services and clients, and internal or abstract code for executing the internal business logic that processes require. A meta-language is any language whose statements refer to statements in another language referred to as the object language. BPEL is often used to compose, orchestrate, and coordinate business processes with Web services in the SOA model, and it has commands to manage asyn-chronous communications.

BPEL uses XML with specific support for messaging protocols such as SOAP, WSDL, UDDI, WS-Reliable Messaging, WS-Addressing, WS-Coordination, and WS-Transactions. BPEL also builds on IBM's Web Services Flow Language (WSFL) and Microsoft's XLANG for data transport; the former is a system of directed graphs, while the latter is a block-structured language adding to additional verbs and nouns specific for business processes to BPEL, which were combined to form BPEL4WS and are being merged with BPEL. A version of BPEL to support human interaction is called BPEL4People, and it falls under the WS-HumanTask specifications of OASIS.

BPEL was designed to interact with WSDL and define business processes using an XML language. BPEL does not have a graphical component. A business process has an internal or executable view and an external or abstract view in BPEL. One process may interact with other processes, but the goal is to minimize the number of specific extensions added to BPEL to support any particular business process. Data functions in BPEL support process data and control flow, manage process instances, provide for logic and branching structures, and allow for process orchestration. Because transactions are long-lived and asynchronous, BPEL includes techniques for error handling and scopes transactions. As much as possible, BPEL uses Web services for standards and to assemble and decompose processes.

Business process modeling

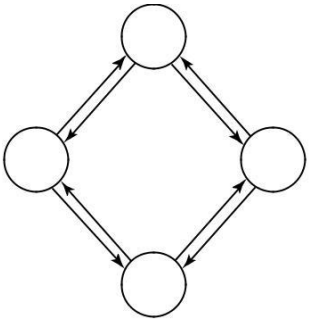
SOA was created by the industry to solve a problem: how to make disparate, diverse, and distrib-uted services talk to disparate and diverse clients. The final result of an SOA project isn't the access of services, per se; it is the creation of a business process. In a complex business project, the devel-opers juggle many clients and many services, which can make visualization of the overall system difficult. To address this problem, various modeling tools have been developed to support SOA development and optimization, system and process management, change and life-cycle management.

Several methodologies have been developed to model SOAs. Working in a software package to model your business processes is similar in approach to designing and optimizing a relational data-base in entity-relationship, object-role modeling package, or another Computer Aided System Engineering (CASE) tool for data storage—and equally as valuable.

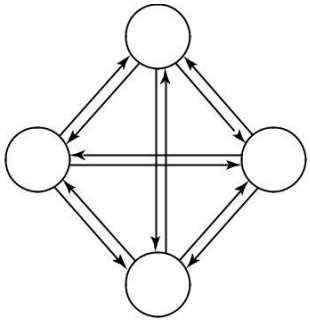
Commonly encountered system models include the following:

- **Unified Modeling Language (UML):** The UML standard is the work of the Object Management Group (http://www.omg.org/). UML creates graphical representations of software systems in the form of a set of diagram types. Elements in a UML architectural blueprint include actors, business processes, logic modules (components), program routines, database schemas, software components, and activities. UML diagrams are separated into seven structural types and four behavior types; structure types model the components of the system, while behavior types model states, actions, and events. UML is widely used in the industry for software system modeling. A developed system model can be reduced automatically to code.
- **XML Metadata Interchange (XMI):** XMI is another standard of the Object Management Group (OMG) and is used to exchange metadata using the Extensible Markup Language (XML). Metadata is structured into a metamodel that fits into the OMG's Meta-Object Facility. UML models often use XMI as their interchange format, although they can be used by other languages. XMI files are not generally interchangeable between the different modeling languages that can use them. XMI has been codified as an international standard by ISO, as IS0/IEC 19503:2005.
- **Systems Modeling Language (SysML)** is an open-source extension of the part of the UML system dealing with profiles. It is smaller, more focused, and easier to learn and work with than UML itself. SysML reuses 7 of UML 2.0's 13 diagrams. The effort to develop SysML was rolled into OMG in 2008, but remains open source. SysML can use XMI and is developing toward support of ISO 10303, which is the Standard for the Exchange of Product Model Data (STEP) AP-233. STEP aims to create the mechanisms for sharing information between the different software engineering tools described in this sec-tion (and others).
- **Business Process Modeling Notation (BPMN)** is a methodology for representing busi-ness processes as a set of connected visual objects that illustrate workflow in a Business Process Diagram (BPD). It is similar to UML. Originally developed in the Business Process Management Initiative (BPMI), it was incorporated into the Open Management Group in 2005. A BPD can be reduced to the OASIS standard WS-Business Process Execution Language, which is an executable language for information transfer between different Web services. However, this mapping is tool specific and standardized at this point.
- **Service-Oriented Modeling Framework (SOMF):** This framework was proposed by Michael Bell and combines a modeling language with a graphical display of the various SOA components so the system can be viewed as a map of objects and associated relation-ships. SOMF software allows developers to create an action plan to implement their busi-ness processes and can be valuable in system and architecture optimization, tracing message pathways, positioning software assets correctly, and providing a language for describing through abstraction and generalization how the processes operate. SOMF soft-ware not only allows you to determine what needs to be done, but also allows you to run "what-if" scenarios to see how changes will impact your SOA system.

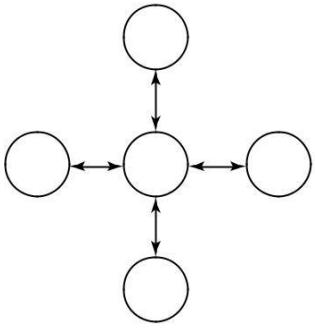
The four different message-passing topologies used in SOMF are shown above. The lines and arrows indi-cate message-passing pathways and relationships.



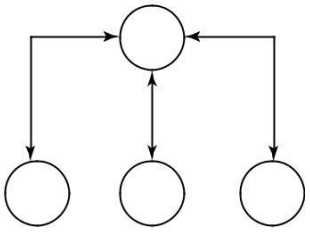
Circular topology



Network topology



Star topology



Hierarchical topology

Rarely do atomic services stand alone. A service that provides ID, name, and address might be part of a set of services that describe a bank account, customer purchase record, or any number of services. A collection of services that work together is referred to as a composite service. The Enterprise Service Bus described previously in this chapter is another example of a composite ser-vice; it contains functions for message routing, message interchange and translation, and process orchestration. A composite service is usually organized as a hierarchical topology and is multi-functional or a coarse-grained entity.

SOA describes a distributed collection of services performing business process functions. A collec-tion of composite services that would form a process module is referred to as a *service cluster*. Service clusters may be composed of both atomic services and composite services. Large functions such as payroll modules would normally be composite SOA services.

Tip

To view a structured presentation on how to build an SOMF diagram, go to http://www.modeling concepts.com/pdf/SOMF_ANALYSIS_MODELING.pdf**. The white paper "Enacting the Service Oriented Modeling Framework (SOMF) Using Enterprise Architect" by Frank Truyen (**http://www.sparxsystems. com/downloads/whitepapers/EA-SOMF_Introduction.pdf**) shows different SOMF diagram types.** ■

In an SOMF model, each of these three service types (atomic, composite, and clusters) appears as a specific shape, and connections are made between them that generalize, specify, expand, or contract the services they provide. Services are typed, granular services are identified, and then services may be aggregated, decomposed, unified, intersected, or subtracted from other services to suit the needs of the business process being modeled. The SOMF modeling notation has a symbol for each analysis that relates one service to another. As you build a business process, you add services to the model and connect them in ways that make sense for your workflow. When the model is complete and optimized, it is reduced to a conceptualized service that relates the business process to the specific implementation chosen. Some modeling technologies allow for the reduction of the model to executable code.

Managing and Monitoring SOA

Software for monitoring and managing an SOA infrastructure plays an important role in large SOA deployments. While SOA offers a logical design and reusable components, it does not make the task of network management any easier. If anything, SOA management requires proactive over-sight because you can't wait for a particular application to fail before taking corrective action. Therefore, tools for managing SOAs tend to be multifaceted and run constantly.

SOA management tools

There are a number of network management frameworks products and suites, notably these:

- HP Software and Solutions OpenView SOA Manager (https://h10078.www1. hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto &cp=1-10^36657_4000_100)
- IBM Tivoli Framework Composite Application Manager for SOA (ITCAM; see http:// www-01.ibm.com/software/tivoli/solutions/),
- Oracle BPEL Process Manager (http://www.oracle.com/technology/bpel/ index.html)

These products have SOA tools for network management. IBM's product specializes in change management and SOA lifecycle development, and it integrates with a WebSphere and other Tivoli systems. HP SOA Manager provides dynamic mapping, monitoring, and optimization of SOA ser-vices such as Web services, software assets, and virtual services. These framework products create a central console with a variety of management views. Oracle's BPEL Process Manager and WebSphere are process managers for creating an Enterprise Service Bus.

The SOA management software technology is dynamic, with many small vendors' products some of which have been purchased and rolled into (or are being rolled into) larger systems. Oracle's recent acquisition of AmberPoint's SOA Management System is an example of this trend. BMC Software's

AppSight (http://www.bmc.com/products/product-listing/BMC-AppSight.html) is an automated SOA problem-resolution package, as is Tidal Software's Intersperse package, which has root cause analysis services. The CA Wily SOA Solution (http:// www.ca.com/us/eitm/solution.aspx?id=8254) is a monitoring and discovery service that can map SOA transactions and dependencies and discover components such as ESBs, Web portals, and various Web services. iTKO's LISA (http://www.itko.com/products/index. jsp). Enterprise SOA Testing platform specializes in testing Web service components that are used in SOA. Another example of an SOA transaction manager is OpTier's CoreFirst (http://www. optier.com/corefirst_overview.aspx).

Configuration and change management present a particular challenge in the area of SOA (and cloud computing in general). In addition to the fact that elements of an SOA infrastructure can be highly distributed and therefore require good discovery mechanisms, these environments also are highly virtualized. As workloads vary, solutions often provision virtual servers as needed and move these virtual servers' processing across physical servers. Virtualization will continue to challenge SOA management software well into the future.

SOA security

Any system that sends hundreds or thousands of messages across an internetwork as SOA does is subject to attack in all the traditional ways that network traffic is hijacked, spoofed, redirected, or blocked. Because SOA eliminates the use of application boundaries, the traditional methods where security is at the application level aren't likely to be effective.

Cisco has a family of products that enforce rules and policies for the transmission of XML messaging that they have named Application Oriented Networking (AON; http://www.cisco.com/en/US/products/ps6480/). A similar policy based XML security service may be found in Citrix's NetScaler 9.0 (http://www.citrix.com/English/ps2/products/product.asp?contentID=21679) Web application delivery appliance.

To address SOA security, a set of OASIS standards (http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security) was created, which includes the following:

- **Security Assertion Markup Language (SAML)** is an XML standard that provides for data authentication and authorization between client and service. The SAML technology is used as part of Single Sign-on Systems (SSO) and allows a user logging into a system from a Web browser to have access to distributed SOA resources.
- **WS-Security (WSS)** is an extension of SOA that enforces security by applying tokens such as Kerberos, SAML, or X.509 to messages. Through the use of XML Signature and XML Encryption, WSS aims to offer client/service security.

- **WS-SecureConversion** is a Web services protocol for creating and sharing security context. WS-SecureConversion is meant to operate in systems where WS-Security, WS-Trust, and WS-Policy are in use, and it attaches a security context token to communications such as SOAP used to transport messages in an SOA enterprise.
- **WS-SecurityPolicy** provides a set of network policies that extend WS-Security, WS-Trust, and WS-SecureConversion so messages complying to a policy must be signed and encrypted. The SecurityPolicy is part of a general WS-Policy framework.
- **WS-Trust** extends WS-Security to provide a mechanism to issue, renew, and validate security tokens. A Web service using WS-Trust can implement this system through the use of a Security Token Service (STS), a mechanism for attaching security tokens to messages and a set of mechanisms for key exchanges that are used to validate tokens and messages.

Another approach to enforcing security in SOA is to use an XML gateway that intercepts XML mes-sages transported by SOAP or REST, identifies the source of the message, and verifies that the mes-sage was securely received. Providing XML Gateway SOA security requires a Public Key Infrastructure (PKI) so that encryption is enforced by digital signatures. Progress Software's Actional 8.0 (http://web.progress.com/en/actional/index.html) now includes Mindreef's SOAPscope Server and has an XML middleware service that performs diagnostic testing and Web services governance, adding that component to Actional's ability to monitor and map XML appliances and application servers.

The Open Cloud Consortium

The Open Cloud Consortium (OCC; see http://opencloudconsortium.org/) is an organi-zation comprised of several universities and interested companies that supports the development of standards for cloud computing and for interoperating with the various frameworks.

OCC working groups perform these functions:

- They develop benchmarks for measuring cloud computing performance. Their benchmark and data generator for measuring large data clouds is called MalStone (http://code.google.com/p/malgen/).
- They provide testbeds that vendors can use to test their applications, including the Open Cloud Testbed and the Intercloud Testbed that are part of the work of the Open Cloud Testbed and Intercloud working groups.
- They support the development of open-source reference implementations for cloud computing. The Working Group on Standards and Interoperability For Large Data Clouds extends the architecture for data storage with a distributed file system, table services, and computing using MapReduce following the model that is part of Google's offering.

MapReduce is Google's patented software framework that supports distributed large data sets organized by the Google File System (GFS) accessed by clusters of computers. The Apache Hadoop (http://hadoop.apache.org/) open-source system is based on MapReduce and GFS.

- They support the management of cloud computing infrastructure for scientific research as part of the Open Science Data Cloud (OSDCP) Working Group's initiative.

Relating SOA and Cloud Computing

Cloud computing is still in its infancy, and although Web services can implement a Service Oriented Architecture, it is not a requirement. Most of the large implementations of cloud comput-ing described in this book are single-purpose applications that have been optimized on a grand scale: Carbonite's backup, Google's Gmail e-mail, and Twitter's Instant Messaging (IM) are several examples. Applications of those types have less of a need for the flexibility and loose coupling that SOA provides. As cloud applications become more diverse in scope, SOA offers an architectural blueprint for accessing diverse optimized services through a loosely coupled standardized method that provides an ability to evolve that is difficult to implement in any other way.

SOA is loosely coupled because the service is separated from the messaging. If a component doesn't provide the capabilities required, it is an easy task to switch to a different component, and switching requires almost no programming. Developers lose some of the ability to customize mod-ules, but gain a significant advantage in simplifying their applications. Taken as a whole, applica-tions that rely on SOA components can be very complex and appear to be tightly coupled, when in reality they are not.

SOA components are often best-of-breed service providers that can provide a measured service level and can play a role in Business Process Management (BPM) systems. The separation of ser-vices from their design allows for much easier system upgrades and maintenance.

Many Web 2.0 applications use SOA components, and SOA will become increasingly useful in larger applications that require many Web services. Web 2.0 is an acronym coined by Tim O'Reilly to describe Web services that allow for user input and modification. These applications often rely on REST and feature AJAX components in a user interface that supports Web syndication (think of the Google customizable user page), blogs, and wikis. Some people regard mashups as Web 2.0 applications as well. A *mashup* is the combination of data from two or more sources that creates a unique service. The layers added to Google maps are examples of mashups.

AJAX stands for Asynchronous JavaScipt and XM. AJAX is a set of development tools that allow for client input into Web applications; it is not a standard. Rather AJAX describes a group of technolo-gies that leverage HTML and CSS for styling, Web objects in the Document Object Model (DOM)

for data, XML and XSLT for data interchange, the XMLHttpRequest for asynchronous communica-tion, and JavaScript commands to request data from data sources.

The challenge SOA faces in designing systems to support Web 2.0 is the lack of standardization in how components in Web 2.0 are used. However, many people believe that SOA will play a role in creating what has been dubbed an "Internet of Services" where complex services will be available for use as a set of building blocks based on the convergence of SOA and Web 2.0. The Gartner Group refers to this trend as the development of "Advanced SOA," but features of SOA that are event-driven have been part of many vendors' middleware offerings for several years now.

Summary

This chapter described Service Oriented Architecture (SOA). SOA offers a design methodology for creating distributed applications using diverse clients and components. SOA defines a message-passing infrastructure from clients or consumers to and from service providers. Making SOA work correctly requires a certain set of middleware products in your infrastructure. These servers may aid in transaction management or brokering, message translation, or other services. Taken as a whole, these services are referred to as an Enterprise Service Bus (ESB).

Most message-passing protocols are based on a version structured XML, although that is not required in SOA. A variety of transport protocols are used, but SOAP and RPC are the most com-mon ones. The nature of SOA messaging was explored. In a complex system of message passing and services, system management and security is an important consideration. Tools for setting up and running an SOA infrastructure were described.

Finally, this chapter described the relationship between SOA and cloud computing. You don't need to use SOA to build a massively scaled cloud computing application, but as cloud computing applications become more capable and user configurable, the logic and structure that SOA design imposes on infrastructure will prove to be invaluable to cloud applications. The two areas of tech-nology benefit from their mutual convergence.

In Chapter 14, I consider the topic of transactional Web applications in cloud computing systems. That subject builds on what you have learned about SOA in this chapter, extending the discussion into the command Web APIs that are in use today.

Moving Applications to the Cloud

In this chapter, you learn about some of the important considerations involved in moving an application from a local or on-premises installa-tion to one that is either fully or partly in the cloud. Some applications benefit from cloud deployment, and the cloud enhances some features.

The process for determining whether, what, and when to move your applica-tions to the cloud involves an analysis of what critical features of the applica-tion need to be supported. After those critical features are understood, you can determine the features supported by your particular cloud service pro-vider to see whether the cloud can support the application's critical features. Factors such as access to data, latencies, data security, and so on often limit what applications are good candidates for porting.

Two examples of application porting to the cloud are discussed in this chap-ter. In one application, physical hardware is eliminated by moving the entire application to the cloud. In the second example, a system is essentially cloned to the cloud to provide an overflow capability, an example of a hybrid application technique called cloud bursting.

When you move an application to the cloud, you must use the APIs of your particular cloud service provider. There are APIs for each of the types of cloud services: infrastructure, software services, and applications in the case of platform providers. These APIs are generally not interoperable. So although the situation may change in the future, an application developer must make an informed choice to select the vendor that both best suits his needs and allows him to have the greatest flexibility.

Applications in the Clouds

When you deploy an application to the cloud, you start with the advantages and disadvantages of a distributed system that is the Internet and add to that mix the fundamental characteristics that clouds offer. In the cloud, your applications must account for system abstraction and redirection, scalability, a whole new set of application and system APIs, LAN/WAN latencies, and other factors that are specific to one cloud platform or another. In theory, any application can run either com-pletely or partially in the cloud.

The question a developer needs to ask is whether his application's function is best served by cloud or local deployment. That answer depends upon the attributes of the application that the developer is trying to preserve or enhance, and how locating those services in the cloud impacts those attri-butes. This chapter takes a broad look at cloud computing from an application-specific viewpoint and attempts to highlight the factors that make cloud-based applications successful.

The location of an application or service plays a fundamental role in how the application must be written. An application or process that runs on a desktop or server is executed coherently, as a unit, under the control of an integrated program. An action triggers a program call, code executes, and a result is returned and may be acted upon.

Taken as a unit, "Request => Process => Response" is an atomic transaction. Because the transac-tion is executing locally within the purview of a monolithic application, the process is stateful and transaction is consistent. That is, the condition of the transaction is always known and the result is always accounted for. A coherent transaction either succeeds and is enacted, or fails and is rolled back. When rollback is not possible due to optimistic transaction commitment in a multiuser application, atomicity requires correcting the condition or performs some other compensating action at some later time.

The properties necessary to guarantee a reliable transaction in databases and other applications and the technologies necessary to achieve them have been called the ACID principle. The acronym stands for:

- **Atomicity:** The atomic property defines a transaction as something that cannot be subdi-vided and must be completed or abandoned as a unit.
- **Consistency:** The consistency property states that the system must go from one known state to another and that the system integrity must be maintained.
- **Isolation:** The isolation property states that the system cannot have other transactions operate on data that is currently being processed by a transaction.
- **Durability:** The durability property states that the system must have a mechanism to recover from committed transactions should that be necessary.

The ACID rules were developed by Jim Gray to apply to database technology in the late 1970s. The ACID principle is used today by any application that is reading and writing to a stored data set, which includes just about any application type you can think of.

An application that runs as a service on the Internet has a client portion that makes a request and a server portion that responds to that request. The request has been decoupled from the response because the transaction is executing in two or more places. In a distributed system, the transaction is stateless. In order to create a stateful system in a distributed architecture, a transaction manager or broker must be added so that the intermediary service can account for transactions and react accordingly when they succeed or fail.

When applications get moved to the cloud, they retain the features of a three-layered architecture, but now physical systems become virtualized systems. Virtual machines are not only stateless, but the place where program execution occurs is likely to be different every time the process runs. These fundamental properties must be accounted for in any cloud-based application.

Functionality mapping

Some applications can be successfully ported to the cloud, while others suffer from the translation. Understanding whether your particular application can benefit from cloud deployment requires that you deconstruct your application's functionality into its basic components and identify which functions are critical and can be supported by the cloud.

For example, any application that requires access to a data store quickly runs up against some of the limits that cloud computing imposes. Order transaction systems require that data in a database maintain the transactional integrity implied by the ACID model. For many non-relational cloud storage systems, such as the Amazon Simple Storage Service (S3), the newly announced Google Storage for Developers, and the Windows Azure Storage Service, the ability of the system to main-tain transactional integrity through record locking isn't part of those systems. These types of stor-age systems are secure and store large amounts of data, but they have very slow access to that data and do not support query and retrieval well. These limitations are why all these vendors offer alter-native relational cloud database systems such as SQL Azure.

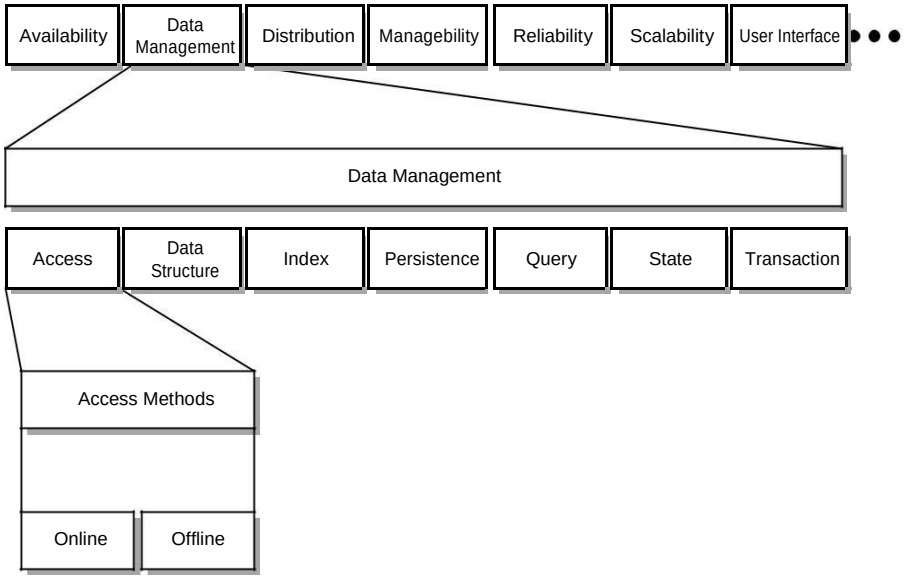
In Figure 14.1, an attribute tree is constructed for an order transaction system where the function-ality is decomposed into different functional areas. At the top are high-level attributes; some of these functions are essential to the operation of the application while others are not. Drilling down on the data management attribute, the second level explores data access and then access methods. A critical attribute for the application is the need to be able to access data when the client is both online and offline.

Note

This exercise isn't required for all attributes, just the critical ones, which should result in a manageable list. ■

FIGURE 14.1

An attribute map is created to expose critical functionality.



The choice to allow both online and offline data access determines the nature of your application's interaction with both cloud and local data stores. If the application needed to access data only when the client was online, then access to cloud-based storage would be the only data store your application would required. Perhaps the application could be entirely in the cloud and browser-based. The decision to allow both online and local data access means that you must create a hybrid application with a cloud component and a local component. Even if the access to data on the local system is a simple caching system, client-side support is needed. To support the application's data access, you may also be faced with building a synchronization or replication feature, which adds more overhead to the application.

This type of mapping exercise leads to some conclusions about the value of cloud computing to this particular application. You could safely conclude that an application that gets the most value from a cloud deployment is one that uses online storage without the need for offline storage. An application that needed offline storage alone might not benefit from a cloud deployment at all. In the case of a hybrid application, other factors such as scalability, costs, or ubiquitous access might offset the cost of offline access and make the cloud more attractive.

Application attributes

Table 14.1 lists some of the first- and second-level application attributes that you might want to consider in your analysis of an application's suitability to be ported to the cloud.

Application Attributes

First Level	Second Level	First Level	Second Level
Application	Abstraction		Implementation
	Architecture		Language/locale
	Configuration		Monitoring
	Interoperability		Operations
	Modularity		Staffing
	Object model		Startup/recovery
	Reusability		Tools
Availability	Caching	Scalability	Caching
	Fault management		Expertise
	Geographic location		Licensing
	Pooling		Lifecycle management
	Resource access		Load balancing
	Reliability		Replication
	Uptime		Scale up or out
Costs	Development		
	Resources		Staging
Data Management	Application needs		
	Data exchange	Security	Access
	Database needs		Auditing
	Index		Authentication
	Online/offline access		Authorization
	Portability		Cryptography
	Query		Encryption
	State		Identity
	Store type		Regulations
	Structure		Remote access
	Transactions		Security rules
			Trust relationships
Maintenance	APIs	User Interface	Ease of use
	Configuration		Interface features
	Deployment		User interaction

Deconstructing an application's critical functionality is only half the process. Each cloud platform also has its own set of attributes that need to be mapped. In considering the needs of any feature, the key drivers for applications that benefit from deployment to the cloud are those that meet these criteria:

- Are not mission critical
- Are not core business functions
- Do not have sensitive data to protect
- Tolerate high network latencies or low network bandwidth
- Are legacy applications with no particular competitive advantage
- Are based on industry standard technologies
- Do not need to be customized
- Are mature enough and understood well enough to be successfully ported to the cloud

Cloud service attributes

You then want to match up application attributes to these key cloud service attributes:

- Applications
- Core services
- Infrastructure
- Platform features
- Storage

At the current stage in its development, it is impracticable to match the needs of an application to a set of cloud service providers. Each provider has a unique solution, uses its own APIs, and provides unique services. Therefore, each cloud provider needs separate developer skills, and integration between clouds would be a major chore. Perhaps someday this situation may change as more standards are developed, but at the moment application developers need to match their application to the single best vendor.

Table 14.2 lists some of the first- and second-level cloud features that you might want to consider in your analysis. You should map your critical application features to the features of a particular cloud platform to get the best match.

Cloud Service Attributes

First Level	Second Level	First Level	Second Level
Application	Accounting		Operating system support (platform)
	Database		Resource pooling
	Event management		Scale up or out
	Messaging		Site location
	Location service		Redundancy and replication
	Relation management		Virtual machine types
	Web server		API
Core Services	Accounting	Platform Features	Application support
	Application support		Deployment technology
	Auditing		Development environment
	Data access		Language and locale
	Identity		Programming language support
	Index		Testing
	Query		API and commands
	Transaction management	Storage	Query
	Workflow		Non-relational
Infrastructure	Application support		Relational
	Availability		Reliability
	I/O (network) characteristics		Replication
	Load balancing		SQL support

System abstraction

The cloud turns physical systems into virtual systems. Organizations choose to deploy systems to the cloud entirely when they can recreate the essential part of their process and eliminate infra-structure. As an example, consider a service that does medical imaging. In the past, this service created patient scans and then rendered the image on a local computer. After the image was ren-dered, it was posted to the hospital LAN and made available to the people who read the scans. When the people reading the scans were outside the hospital, across the country, or around the world, those people would have to log into the hospital server via VPN to download the file.

The scanning service decided to eliminate infrastructure and streamline the process. The service began its redeployment by first moving the stored images off the hospital's LAN and onto shared storage in the cloud. This feature eliminated the need to maintain a great deal of managed storage locally. As the service began to outsource the reading of scans to other countries, it enabled a content delivery network feature that the cloud service provider had. CDN (Content Delivery Network) placed copies of recently used and created scans in locations that were closer to the readers and made the system faster.

The second stage in the redeployment was to eliminate the local processing associated with the scanning machines themselves. Most of the time the scanning machine was operating, it was collecting data, and an economic analysis revealed that it was significantly cheaper to process the files in the cloud.

In the new system, shown in Figure 14.2, the files are created locally and transmitted to the cloud. Virtual machines are provisioned to process the scans. The system leverages a message queuing server to create a steady stream of execution for the application server to process. At times of peak load, the system creates new machine instances to handle the load. As the application server com-pletes the scan processing, it notifies the message queue, records the result in a database, and dis-plays it on a Web page on a Web server, all of which are in the cloud.

This new system results in greater system efficiencies because the system is always processing at its optimum load. The rendered scans are available from anywhere viewed inside a browser. Also, because the system is scalable, the scanning service can expand to other sites and bring on new capacity to handle additional load. As the service loses sites, it can also release resources as well. When it is decided that the scans need to be converted into a different format, this can be done in a central location and doesn't need to be rolled out to the computers attached to individual scan systems.

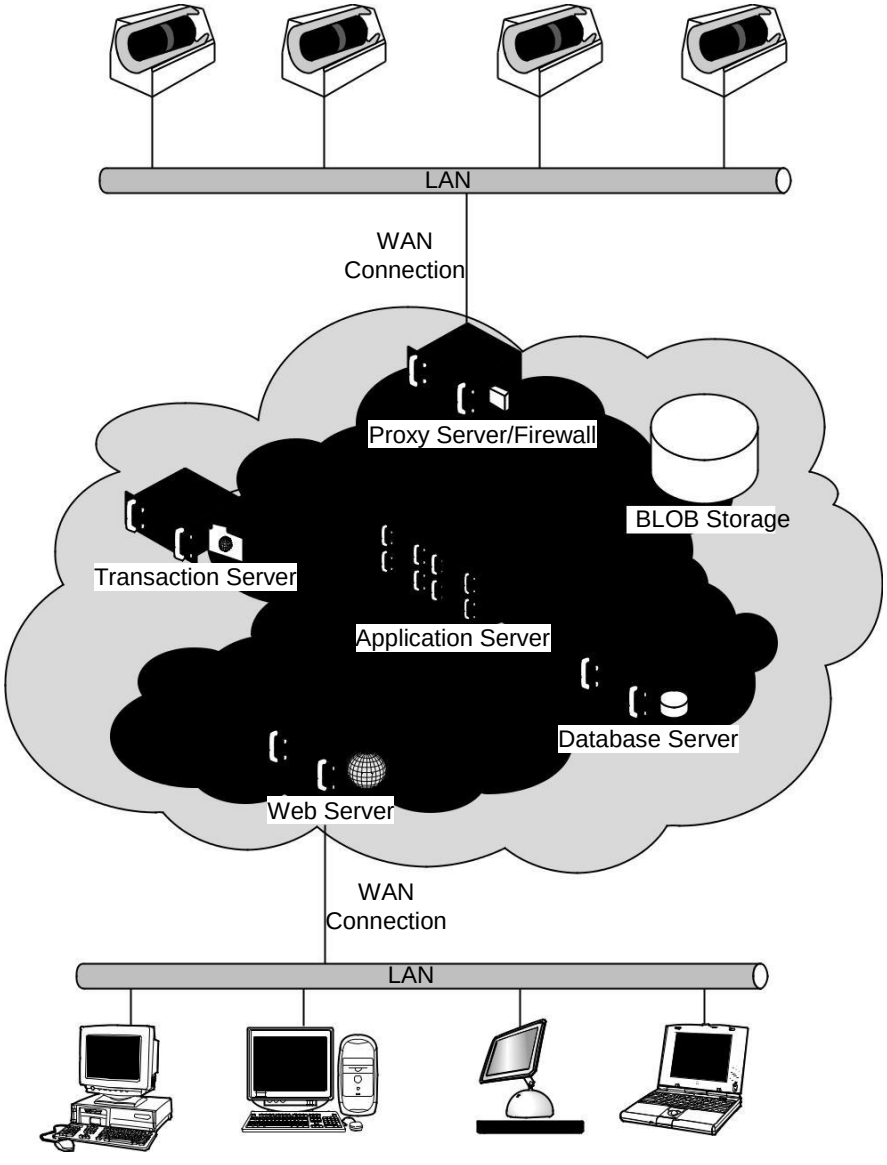
Infrastructure, storage, and the queuing system all come together to eliminate a great deal of cost and operational complexity. This is a pure cloud play.

Cloud bursting

Many cloud deployments are hybrid applications: Part of the application is on a local system, and part is in the cloud. Often, this is the first stop on the path for many organizations migrating their applications to the cloud. There are many reasons why this is desirable, but one of the most common reasons is that the cloud can serve as excess capacity at times of high volume. This type of hybrid has been called *cloud bursting.* Examples of systems where there is high volume over short periods of time are transaction processing systems such as reservations systems.

In a reservation system, there is a certain low level of background transactions occurring at any time. At certain times, events trigger high demand. If the system builds infrastructure to accommodate peak demand, then that infrastructure is wasted.

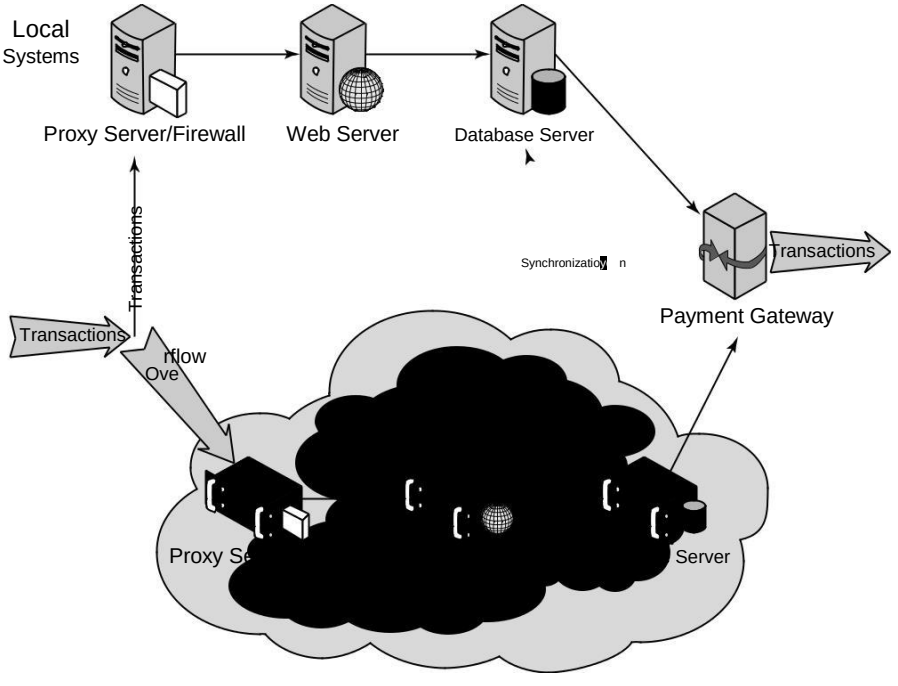
An application deployed entirely to the cloud



Most systems built to perform cloud bursting have a simple underlying design: clone the local system in the cloud. Often, there may be little activity in the cloud portion of the system, but when the activity grows, the copy of the system in the cloud picks up the extra activity and, when necessary, provisions extra resources. Figure 14.3 shows a simple reservations system set up for cloud bursting.

FIGURE 14.3

An application that provides for transaction overflow in a reservation system is an example of cloud bursting.



Reservation systems often require that transactions not only are atomic, but that when there is a pool of items being reserved, the system is consistent. When a transaction enters the local branch in Figure 14.3 and another transaction enters the cloud platform branch, they can't both reserve the same item. So there must be a transaction manager in this system to manage the pool. This is shown as a dotted line between the two database servers, labeled "Synchronization." The underlying mechanism is to perform record locking on a set of database records and when the transaction or a batch of transactions completes, the system performs a commit operation.

In most reservation systems, the actual transaction commitment is a small part of the traffic and processing. Most of the traffic is generated on the Web site as users browse the content. So it makes sense in this scenario to recreate the company Web site and create additional load-balanced Web server instances as needed. You also can optimize that Web site for faster transactions with less

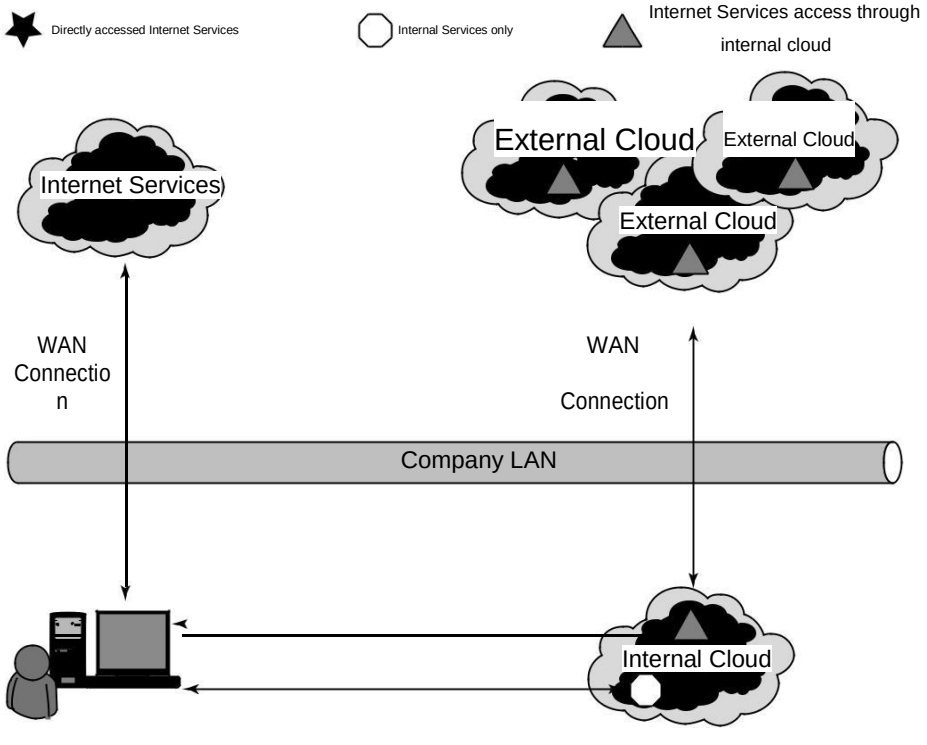
customization. If your Web site relies on dynamic data driven content, you can speed up its operation by switching more of the content over to static content. You'll need to synchronize changes between your on-premises and cloud-based Web servers in order to keep the information current.

The other step in a reservation system that is often a bottleneck is the payments gateway to credit card companies and financial institutions. It may make sense to move the payment portion completely to the cloud so that the processing of payments doesn't affect the other parts of the system. Because the commitment of the payment either is effective or not, this portion of the process does not need to be tracked. The fact that a virtual server is executing the payments and that the process is stateless has no impact in this point.

Eventually, developers will want to create composite applications that are built from the best-of-breed cloud services on multiple platforms. This offers the benefit of redundant suppliers, access to additional services and features, more data sources, and a whole host of other advantages. Cloud architectures offer enough advantages that over time large organizations will want to adopt them as a core architectural design. For example, in Figure 14.4, an internal cloud provides high-speed transactional services on the LAN, an external cloud services other needs of users, and the com-pany cloud is replicated to multiple sites. Where services get located then depends upon factors such as cost, latency, and convenience.

FIGURE 14.4

For the users in large organizations, it is literally clouds everywhere in their future.



Applications and Cloud APIs

The nature of a cloud provider's Cloud API will impact your ability to move an application to the cloud and affect the way many of your application features operate. Cloud APIs are the Application Programming Interface to functions that exchange information in and with the cloud, request sup-ported operations, and provide management and monitoring functions for applications running in the cloud.

At this stage in cloud computing's development, Amazon Web Service's Cloud API dominates the conversation, but that is probably subject to change. Each cloud vendor has its own specific API; most are exposed as REST, a few are exposed as SOAP, or some are both. Each API provides spe-cific calls required by that vendor's infrastructure and service.

Most importantly, the cloud API contains the authentication and authorization mechanisms needed to access cloud services. When a vendor like Google allows other cloud application provid-ers to access its ID mechanism (as is the case now), there is the same flexibility in using those ser-vices on other platforms.

Although efforts are underway to create more standardized cloud APIs, the situation limits the portability of any application developed for the cloud. The two cross-platform API initiatives are the Simple Cloud API (http://www.simplecloud.org/) and the work arising out of the Cloud Computing Interoperability Forum (http://www.cloudforum.org/). Several cross-platform cloud API projects are underway, including the Design Cloud, Deltacloud, jclouds, and libcloud APIs. These cross-platform APIs are based on generalizing the major cloud vendors APIs.

Each layer of a cloud application has its own specific API as well. So at the infrastructure level in addition to AWS EC2, you have the following:

- Windows Azure (http://www.microsoft.com/windowsazure/windows azure/)
- VMWare vCloud (https://www.vmware.com/products/vcloud/)
- Racksapce Cloud Servers (http://www.rackspacecloud.com/cloud_hosting_ products/servers/api)
- RimHosting (http://rimuhosting.com/)

All these present the developer with their own APIs. Individual services such as Windows Azure SQL, Flickr, and Google Maps present a service cloud API. If your application is developed in a platform such as Facebook, LinkedIn, or the Salesforce Force APIs, each platform has its own spe-cific application API.

The point is that the decision to move an application to the cloud rapidly funnels you into a specific solution that provides a measure of vendor lock-in that, depending upon the nature of your application, can be anywhere from very easy to nearly impossible to port to any other cloud technology. This may not always be so, but it currently is true and it should give a developer some moments to pause.

Summary

In this chapter, you learned about some of the factors involved in deciding to move an application to the cloud. Cloud computing supports some application features better than others. To deter-mine whether your application will port successfully, you should perform a functionality mapping exercise. This process involves determining the critical application features and then matching them to the cloud provider's offering to see if those features can be supported.

Examples of applications that were ported were presented. One application virtualized the entire application in the cloud and presented users with a browser-based service. The second scenario, called cloud bursting, is an overflow solution that clones the application to the cloud and directs traffic to the cloud during times of high traffic.

The role of a cloud vendor's specific API and the impact that it has on porting an application was also considered. This aspect of a migration isn't given enough thought beforehand and can cause problems later on should you wish to move to other solutions.

In Chapter 15, "Working with Cloud-Based Storage," various storage, backup, and disaster recov-ery solutions offered by cloud vendors are discussed.

Working with Cloud-Based Storage

The world is creating massive amounts of data. A large percentage of that data either is already stored in the cloud, will be stored in the cloud, or will pass through the cloud during the data's lifecycle.

Cloud storage systems are among the most successful cloud computing applications in use today. This chapter surveys the area of cloud storage systems, categorizes the different cloud storage system types, discusses file-sharing and backup software and systems, and describes the methods being used to get cloud storage systems to interoperate.

Cloud storage can be either unmanaged or managed. *Unmanaged storage* is presented to a user as if it is a ready-to-use disk drive. The user has little control over the nature of how the disk is used. Most user-oriented software such as file-sharing and backup consume unmanaged cloud storage. Applications using unmanaged cloud storage are Software as a Service (SaaS) Web services.

Managed storage involves the provisioning of raw virtualized disk and the use of that disk to support applications that use cloud-based storage. Storage options involved in formatting, partitioning, replicating data, and other options are available for managed storage. Applications using managed cloud storage are Infrastructure as a Service (IaaS) Web services.

Developing cloud storage interoperability standards are described in this chapter, notably those from the Storage Networking Industry Association (SNIA) and the Open Grid Foundation (OGF). The Cloud Data Management

Interface (CDMI) interoperability storage object protocol is described. This interface can store data objects, discover stored data objects, and supply these data objects to subscribing applications. The Open Cloud Computing Interface (OCCI) is another storage data interchange interface for stored data objects. The two protocols interoperate with one another.

Measuring the Digital Universe

The world has an insatiable hunger for storage. This hunger is driven by the capture of rich media, digital communications, the Web, and myriad other factors. When you send an e-mail with a 1GB attachment to three people, this generates an estimated 50GB of stored managed data. Only 25 per-cent of the data stored is unique; 75 percent of stored data is duplicated. You may be surprised to learn that 70 percent of the data stored in the world is user initiated; the remainder is enterprise-generated content.

Video cameras and surveillance photos, financial transaction event logs, performance data, and so on create what IDC (International Data Corporation; http://www.idc.com/), the research analysis arm of International Data Group has called the "digital shadow"—data that is automatically generated. Shadow data represents more than 50 percent of the data created every day. However, lots of shadow data does get retained, having never been touched by a human being.

Much of the data produced is temporal, stored briefly, and then deleted. That's a good thing, because there is a growing divide between the amount of data that is being produced and the amount of storage available.

The storage giant EMC has an interest in knowing just how much data is being stored worldwide. EMC has funded some studies over the past decade to assess the size of what it calls "The Digital Universe." The latest study done by IDC in 2007-2008 predicted that by 2011 the world will store 1800 exabytes (EB) or 1.8 zettabytes (ZB) of data. By the year 2020, stored data will reach an astonishing 35ZB. The number of managed objects stored in containers—files, images, packets, records, signals, and so on—is estimated to be roughly 25 quintillion (1018) containers. A *container* is a term of art in cloud storage.

These numbers are astronomical, and wrapping your mind around them can be hard. Even more astonishing is the fact that the amount of stored data is doubling roughly every five years. In 2007, the last year before the recession of 2008, the amount of stored data grew even faster, by 60 percent

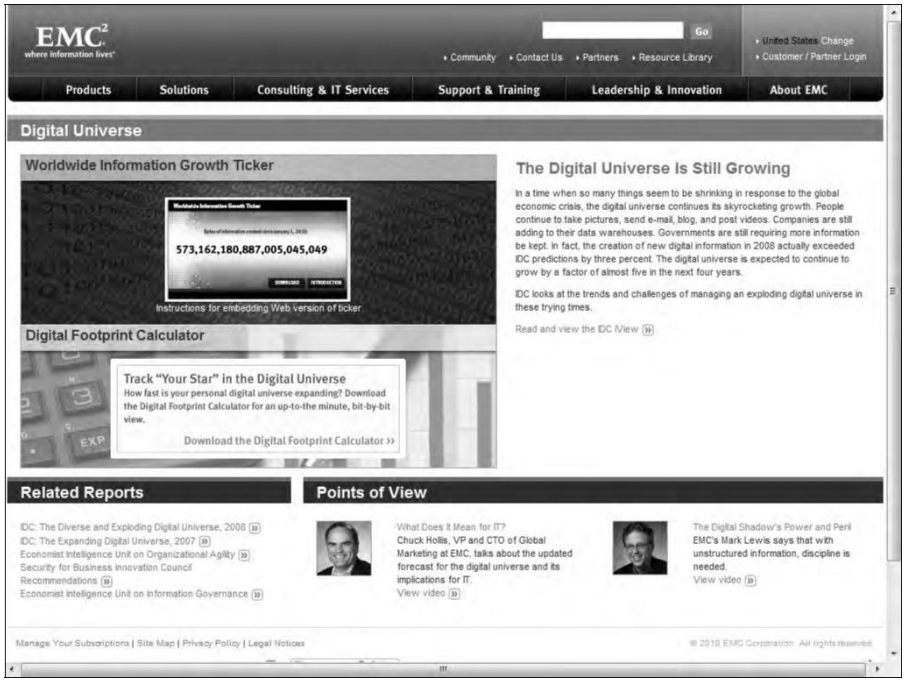
annually. You can visit EMC's Digital Universe home page, shown in Figure 15.1, to link to the IDC study, view the Digital Data Consumption Ticker, and get EMC's take on the problems associ-ated with managing vast data sets.

Note

Here are some definitions of scale: a gigabyte is 10^9 bytes, a petabyte is 10^{15} bytes; an exabyte is equal to one billion gigabytes or 10^{18} bytes; a zettabyte is equal to one trillion gigabytes or 10^{21} bytes; and a yottabyte (YB) is 10^{24} bytes. ■

FIGURE 15.1

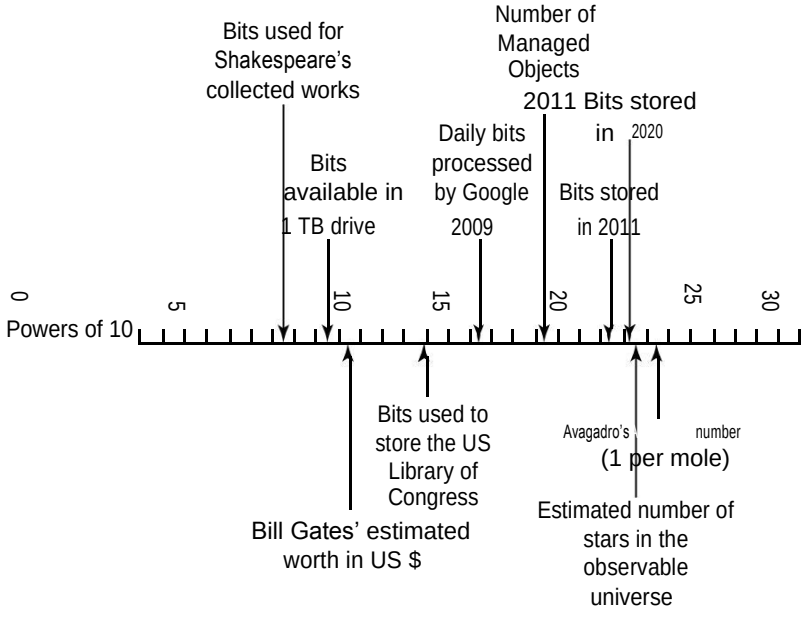
EMC's Digital Universe Web page located at http://www.emc.com/leadership/digital-universe/expanding-digital-universe.htm



To provide some measure of scale, the size of William Shakespeare's complete works downloaded as text from Gutenburg.org (http://www.gutenberg.org/etext/100) is 5.1MB. The amount of stored information in the United States Library of Congress is about 10 terabytes of data (10,000 gigabytes), and in 2009 Google was processing around 24 petabytes of data per day. Figure 15.2 shows a logarithmic scale with different data storage sizes.

FIGURE 15.2

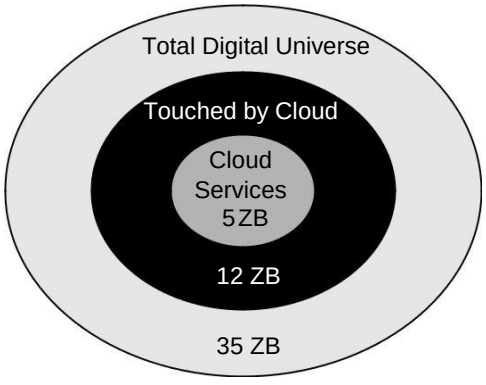
Data storage plotted on a logarithmic scale



Cloud storage in the Digital Universe

A very significant fraction of this data is now or will be residing in cloud storage. Even more will pass through cloud storage in its use. IDC's 2010 study attempted to estimate the percentage of data that will be stored in the cloud or passed through the cloud in the year 2020. There will be a steady growth of cloud storage at the expense of online storage over the next decade. Figure 15.3 shows a graphical illustration of the impact of cloud storage systems on the overall Digital Universe in 2020.

Cloud storage data usage in the year 2020 is estimated to be 14 percent resident and 34 percent passing through the cloud by IDC. Source: IDC Digital Universe, May 2010.



Cloud storage definition

Think of cloud storage as storage accessed by a Web service API. The characteristics that separate cloud storage include network access most often through a browser, on-demand provisioning, user control, and most often adherence to open standards so that cloud storage may be operating-system-neutral and file-system-neutral. These characteristics, taken as a whole define an offering that is best described as an Infrastructure as a Service model. However, most users do not provision storage under IaaS systems such as Amazon S3 (described in Chapter 9). Instead, most users interact with cloud storage using backup, synchronization, archiving, staging, caching, or some other sort of software. The addition of a software package on top of a cloud storage volume makes most cloud storage offerings conform to a Software as a Service model.

Storage devices may be broadly categorized as either block storage devices or file storage devices. A block storage device exposes its storage to clients as Raw storage that can be partitioned to create volumes. It is up to the operating system to create and manage the file system; from the standpoint of the storage device, data is transferred in blocks. The alternative type of storage is a file server, most often in the form of a Network Attached Storage (NAS) device. NAS exposes its storage to clients in the form of files, maintaining its own file system. Block storage devices offer faster data transfers, but impose additional overhead on clients. File-oriented storage devices are generally slower (with the exception of large file-streaming applications), but require less overhead from attached clients. Cloud storage devices can be either block or file storage devices.

Provisioning Cloud Storage

Cloud storage may be broadly categorized into two major classes of storage: unmanaged and man-aged storage. In unmanaged storage, the storage service provider makes storage capacity available to users, but defines the nature of the storage, how it may be used, and by what applications. The options a user has to manage this category of storage are severely limited. However, unmanaged storage is reliable, relatively cheap to use, and particularly easy to work with. Most of the user-based applications that work with cloud storage are of this type.

Managed cloud storage is mainly meant for developers and to support applications built using Web services. Managed cloud storage is provisioned and provided as a raw disk. It is up to the user to partition and format the disk, attach or mount the disk, and make the storage assets available to applications and other users.

The sections that follow describe these two storage types and their uses in more detail.

Unmanaged cloud storage

With the development of high-capacity disk storage starting in the mid- to late-1990s, a new class of service provider appeared called a Storage Service Provider (SSP). Fueled by venture capital and the dot.com boom, dozens of companies created datacenters around the world with the intent of doing for online storage what Internet service providers (ISP) did for communications.

With so much excess capacity in place, companies with names like iDrive (now at http://www.driveway.com/), FreeDrive (http://www.freedrive.com/) MyVirtualDrive (defunct), OmniDrive (gonzo), XDrive (kaput), and others were formed to offer file-hosting services in the form of unmanaged cloud storage. It is unmanaged storage in the sense that the storage is precon-figured for you, you can't format as you like, nor can you install your own file system, or change drive properties such as compression or encryption.

Storage was offered to users by these file-hosting services as fixed online volumes. These volumes were first accessible using FTP, then from a utility, and then from within a browser. Often the ser-vice offered a certain capacity for free, with the opportunity to purchase more online storage as needed. FreeDrive is an example of an unmanaged storage utility set up to do automated backups, a class of Web services that is discussed in the section "Exploring Cloud Backup Solutions" later in this chapter.

Three factors led to the demise of many of the early SSPs and to many hosted file services:

- The Dot.com bust in 2000
- The inability of file-hosting companies to successfully monetize online storage
- The continued commoditization of large disk drives, which led to free online storage from large vendors such as Google

These SSPs and file-hosting services were ahead of their time, but in many cases—through acquisi-tions and offspring ventures—their legacy remains.

The simplest of these unmanaged cloud storage services falls into the category of a file transfer utility. You can upload files to the service where that file is stored (for a while) and made available to you for downloading from another location. Some of these services allow the transfer of only a single file. File transfer services may be shared by other users you allow, and the files that are uploaded may be discoverable for some services. That is, you can query the system for a file or information that meets the criteria you set. The service FreeDrive is storage that allows Facebook users to view the content of others.

Dropbox, shown in Figure 15.4, is an example of a file transfer utility. You install the Dropbox utility on your system and create an account, and the Dropbox folder appears. Dropbox also installs a System Tray icon in Windows for you. You can then drag and drop files and folders to your Dropbox. When a remote user logs into a Dropbox account, he installs the Dropbox folder for that account on his system, creating what is in effect a shared folder over the Web.

Managed cloud storage

[redacted] basic service that online storage can serve is to provide disk space on demand. In the previous section, you saw examples of services where the service provider prepares and conditions the disk space for use by the user, provides the applications that the user can use with that disk space, and assigns disk space to the user with a persistent connection between the two. The user may be able to purchase additional space, but often that requires action by the service provider to provision the storage prior to use. That type of storage is considered unmanaged cloud storage because the user can't proactively manage his storage.

The second class of cloud storage is what I call managed cloud storage. You saw an example of a managed cloud storage system in Chapter 9 where Amazon's Simple Storage System (S3) was described. In a managed cloud storage system, the user provisions storage on demand and pays for the storage using a pay-as-you-go model. The system presents what appears to the user to be a raw disk that the user must partition and format. This type of system is meant to support virtual cloud computing as the virtualized storage component of that system.

Note

SNIA (Storage Networking Industry Association; http://www.snia.org/**) has coined the term Data Storage as a Service (DaaS) to describe the delivery of storage on demand to clients over a distributed system. Others have called these types of system services Storage as a Service (STaaS). ■**

Managed cloud storage providers include the following:

- **Amazon.com Simple Storage Service** (S3; http://aws.amazon.com/s3/): This hosting service is described in Chapter 9.
- **EMC Atmos** (http://www.emc.com/products/family/atmos.htm): With Atmos, you can create your own cloud storage system or leverage a public cloud service with Atmos online.
- **Google Storage for Developers** (http://code.google.com/apis/storage/docs/overview.html): Code named "Platypus," this service currently in beta allows developers to store their data in Google's cloud storage infrastructure. It will share Google's authentication and data sharing platforms.

- **IBM Smart Business Storage Cloud** (http://www-935.ibm.com/services/us/index.wss/offering/its/a1031610): IBM has both infrastructure and software offerings that allow businesses to create and manage a private storage cloud.

  IBM is a major player in cloud computing (http://www.ibm.com/ibm/cloud/), particularly for businesses. The company offers a hardware platform called CloudBurst, as well as a portfolio of software that leverages cloud infrastructure such as IBM Smart Analytics Cloud, IBM Information Archive, IBM LotusLive, and LotusLive iNotes.

- **Iron Mountain** (http://www.ironmountain.com/storage/storage-as-a-service.html): Iron Mountain's service is mainly focused on backup and digital archiving, not on storage hosting.

- **Nirvanix** (formerly Streamload; http://www.nirvanix.com/): The company's MossoFS offers a managed cloud service.
- **Rackspace Cloud** (http://www.rackspace.com/index.php): Rackspace is a direct competitor to Amazon's S3 service.

Creating cloud storage systems

The Internet was designed to be a fault-tolerant network that could survive a nuclear attack. Paths between endpoints are redundant, message transfer is packetized, and dropped or lost packets can be retransmitted and travel different paths. Networks are redundant, name servers are redundant, and overall the system is highly fault tolerant. These features help make cloud-based storage sys-tems highly reliable, particularly when multiple copies of data are stored on multiple servers and in multiple locations. Failover can involve a system simply changing the pointers to the stored object's location.

In Chapter 9, you saw how Amazon Web Services (AWS) adds redundancy to its IaaS systems by allowing EC2 virtual machine instances and S3 storage containers (bucket) to be created in any one of its four datacenters or regions. AWS S3 essentially lets you create your own cloud storage, provided you distribute your provisioned storage appropriately on Amazon's system.

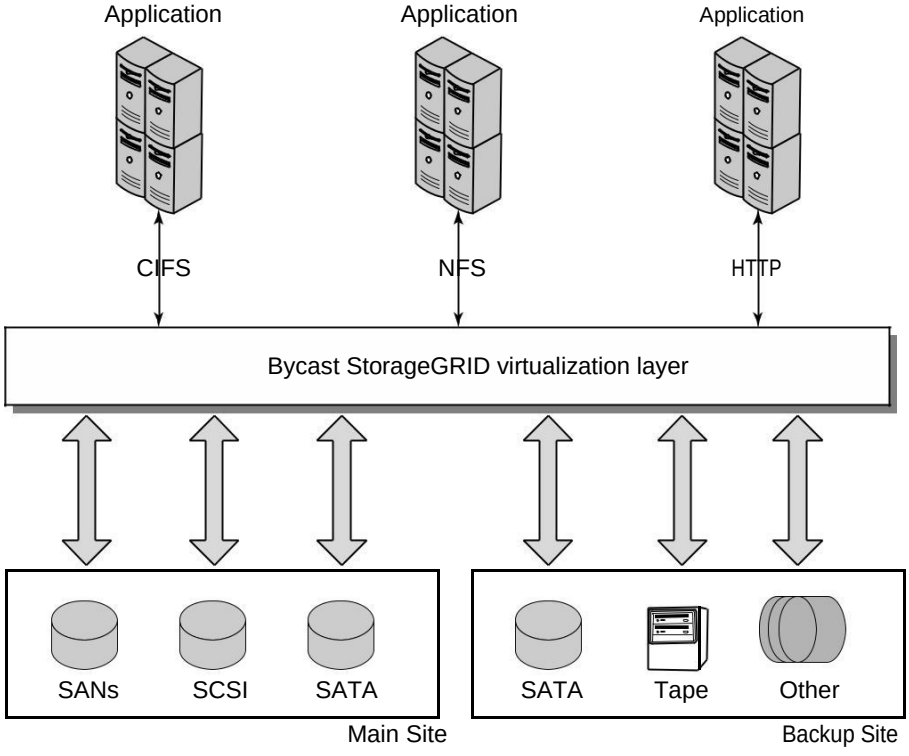
AWS created "Availability Zones" within regions, which are sets of systems that are isolated from one another. In theory, instances in different availability zones shouldn't fail at the same time. In practice, entire regions can be affected, and storage and system redundancy needs to be established on a multi-regional basis. AWS can perform load balancing on multiple instances and can perform failover from one geographical location to another, but this is an additional service that you must purchase. The important point about redundancy is that for it to be effective, it has to be imple-mented at the highest architectural level.

Companies wishing to aggregate storage assets into cloud storage systems can use an enterprise software product like StorageGRID. This storage virtualization software from Bycast (now a part of NetApp; http://bycast.com/) creates a virtualization layer that pools storage from different storage devices into a single management system. You can potentially pool petabytes of data stor-age, use different storage system types, and transport protocols even over geographically dispersed locations. Figure 15.5 shows how SystemGRID virtualizes storage into storage clouds.

StorageGRID can manage data from CIFS and NFS file systems over HTTP networks. Data can be replicated, migrated to different locations, and failed over upon demand. The degree of data repli-cation can be set by policy, and when storage in the pool fails, StorageGRID can failover to other copies of the data on redundant systems. StorageGRID can enforce policies and create a tiered stor-age system.

FIGURE 15.5

ByCast's StorageGRID allows you to create fault-tolerant cloud storage systems by creating a virtualization layer between storage assets and application servers.



Virtual storage containers

In traditional pooled storage deployments, storage partitions can be assigned and provide a device label called a Logical Unit Number (LUN). A LUN is a logical unit that serves as the target for stor-age operations, such as the SCSI protocol's READs and WRITEs (PUTs and GETs). The two main protocols used to build large disk pools, particularly in the form of Storage Area Networks (SANs), Fibre Channel and iSCSI both use LUNs to define a storage volume that appears to a connected computer as a device. Unused LUNs are the equivalent of a raw disk from which one or more vol-umes may be created.

Traditionally, pooled online storage assigns a LUN and then uses an authorization process called LUN masking to limit which connected computers (or hosts) can see which LUNs. LUN masking isn't as strong a security feature as the direct identification of a physical Host Bus Adapters (HBAs),

which are the storage networking equivalent of NICs (Network Interface Cards). LUN addresses can have their unique addresses spoofed more easily than a hardware address can. However, LUNs do protect against a server being able to write to a disk to which it shouldn't have access. Storage partitioning in large storage deployments also may be achieved using SAN zoning, as well as a par-titioning disk based on physical location.

When online storage is converted for use in a cloud storage system, none of these partitioning methods allows for easy, on-the-fly storage assignment and the high disk utilization rates that are required in a multi-tenancy storage system. Delivering effective cloud storage solutions requires the use of a virtual storage container, which allows a tenant to perform storage operations on the vir-tual storage container consistent with the capabilities of the underlying storage system. Different storage vendors call their virtual storage containers by different names, but all use this entity as a construct to create high-performance cloud storage systems. LUNs, files, and other objects are then created within the virtual storage container. Figure 15.6 shows a model for a virtual storage con-tainer, which defines a cloud storage domain. This model, based on the SNIA model but modified somewhat, includes the interface operations that are needed to use that domain.

When a tenant is granted access to a virtual storage container, he performs standard disk opera-tions such as partitioning, formatting, file system modifications, and CRUD (Create, Read, Update, and Delete) operations as desired. Data stored in a virtual storage container may be stored in chunks or buckets as the Amazon Simple Storage Service (Amazon S3) does, or it may be stored in containers that are in a hierarchical relationship typical of most file systems. The main requirement is that however cloud storage data is organized that data and its associated metadata may be discoverable.

Making cloud storage data discoverable on a TCP/IP network using HTTP or some other protocol requires that objects be assigned a unique identifier such as a URI (Uniform Resource Identifier) and that the relationship between objects and their metadata be specified. In the section "Developing Cloud System Interoperability," I describe the OCCI protocol for discovering and retrieving objects from a cloud.

Because virtual storage containers must be secured, these objects must carry a set of security attri-butes that protect a tenant's data from snooping, denial of service attacks, spoofing, inappropriate deletion, or unauthorized discovery. The main mechanism for securing one tenant's virtual storage container from another is to assign an IP address to the virtual storage container and then bind that container to a separate VLAN connecting storage to the tenant (host). Traffic flowing over the VLAN is encrypted, and the tenant is carefully authenticated by the system. Usually, data sent over the VLAN is compressed to improve data throughput over a WAN connection.

Different cloud storage vendors may implement their own proprietary management interfaces to connect distributed hosts or tenants to their provisioned storage in the cloud and to provide for security services. One open interface standard is the Storage Networking Industry Association's Cloud Data Management Interface (CDMI) described later in this chapter.

In evaluating cloud storage solutions, these factors are deemed to be important considerations:

- Client self-service
- Strong management capabilities
- Performance characteristics such as throughput
- Appropriate block-based storage protocol support such as iSCSI or FC SAN, or file-based storage protocol support such as NFS or CIFS to support your systems
- Seamless maintenance and upgrades

In addition to assigning security-provisioned cloud storage to a client or service requester, a cloud storage service provider must be able to deliver a measured level of service as captured by its stated Service Level Agreement (SLA). An SLA might specify that a particular Quality of Service (QoS) for a virtual storage container may be measured in terms of the I/O per second or IOPS that may be provided, as well as the reliability and availability of the service. QoS levels may be applied to dif-ferent services against a single virtual storage container or a single service applied to the client's multiple assigned storage containers. In situations where multiple virtual storage containers are assigned to a client, a mechanism would need to be provided to federate the different containers into a consolidated management console.

One unique characteristic of cloud-based storage solutions is that they permit rapid scaling, both in terms of performance and storage capacity. To provide for scaling, a virtual storage container must be easily migrated from one storage system to another. To increase the capacity of a storage provision, it is necessary to provide the capability to scale up or scale out across storage systems. To scale up, the service must allow for more disks and more spindles to be provisioned. To scale out, the service must allow for stored data to span additional storage systems. Cloud storage systems that successfully scale their provisioned storage for clients often allow for the multiple storage systems to be geographically dispersed and often provide load-balancing services across different storage instances.

Exploring Cloud Backup Solutions

Cloud storage is uniquely positioned to serve as a last line of defense in a strong backup routine, and backing up to the cloud is one of the most successful applications of cloud computing. This area is a cornucopia of solutions, many inexpensive and feature rich.

Backup types

Backups may be categorized as belonging to one of the following types:

- **Full system or image backups:** An image backup creates a complete copy of a volume, including all system files, the boot record, and any other data contained on the disk. To create an image backup of an active system, you may need to stop all applications (quiesce the system). An image backup allows a system to do what is referred to as a bare metal restore. Ghost is an example of software that supplies this type of backup.

- **Point-in-time (PIT) backups or snapshots:** The data is backed up, and then every so often changes are amended to the backup creating what is referred to as an incremental backup. This type of backup lets you restore your data to a point in time and saves multi-ple copies of any file that has been changed. At least 10 to 30 copies of previous versions of files should be saved.

  The first backup is quite slow over an Internet connection, but the incremental backup can be relatively fast. For example, software such as Carbonite may take several days to backup a system, but minutes to create the snapshot.

Note

The amount of time needed to backup a system is referred to as its backup window. ■

- **Differential and incremental backups:** A differential backup is related to an incremental backup, but with some subtle differences in the way the archive bit is handled. During an incremental backup, any changed files are copied to the backup media and their archive attribute is cleared by the incremental backup. In a differential backup, all of the changed files since the last full backup are copied by the backup software, which requires that the software leave the archive bit set to ON for any differential backup, as only a full backup can clear all files' archive bit.

  An archive bit is used by backup software to specify whether a file should be backed up or not. The bit is set on for backup and cleared when backup has copied the file. In a sense, an archive bit is a directive to the software. The archive bit comes into play in backup soft-ware in the sense that an incremental backup solution must examine the full backup data and then analyze all subsequent increments to find the latest file(s). In a differential backup, the backup software can obtain the up-to-date backup from the last full backup and the last incremental backup alone. Files in intervening incremental backups may be taken as temporary or scratch versions. While incremental backups are faster and more efficient from a storage perspective, they are also less fault tolerant.

- **Reverse Delta backup:** A reverse delta backup creates a full backup first and then period-ically synchronizes the full copy with the live version. The older versions of files that have been changed are archived so that a historical record of the backup exists. Among the soft-ware that uses this system is Apple's Time Machine and the RDIFF-BACKUP utility.

- **Continuous Data Protection (CDP) or mirroring:** The goal of this type of backup sys-tem is to create a cloned copy of your current data or drive. A cloud storage system con-tains a certain built-in latency, so unless the original data set is quiescent, the mirror lags behind the original in concurrency.

- **Open file backup:** Some applications such as database systems and messaging systems are mission critical and cannot be shut down before being backed up. An open file backup analyzes the transactions that are in progress, compares them to the file(s) at the start of the backup and the file(s) at the end of the backup, and creates a backup that represents a complete file as it would exist at the time the backup started after all the transactions have been processed. This is a difficult proposition, and open file backup systems are expensive and highly customized to a particular application such as SQL Server or Exchange.

3-2-1 Backup Rule

Peter Krogh's 3-2-1 Rule for data protection is a good one to follow. Krogh is a professional photographer, a member of the American Society of Media Photographers, and a consultant in the area of data storage and archiving. One of his clients is the Library of Congress, where data archival is a mission-critical task. As Krogh states on the site dpBestflow.org (http://www.dpbestflow.org/backup/ backup-overview#321), a simple but effective backup scenario includes the following elements:

3. Retain **three** copies of any file—an original and two backups.
2. Files must be on **two** different media types (such as hard drives and optical media) to protect against different types of hazards.
1. **One** copy must be stored offsite (or at least online).

If you have a local version of a file, then a version of that file stored in the cloud conforms to all three of the 3-2-1 backup rules.

- **Data archival:** The term *archiving* is used to specify the migration of data that is no longer in use to secondary or tertiary long-term data storage for retention. An archive is useful for legal compliance or to provide a long-term historical record.

Note

Data archives are often confused with backups, but the two operations are quite different. A backup creates a copy of the data, whereas an archive removes older information that is no longer operational and saves it for long-term storage. You can't restore your current data set from an archive. ■

Cloud backup features

Features of cloud storage backup solutions that are valuable listed roughly in order of importance include the following:

- Logon authentication.
- High encryption (at least 128-bit) of data transfers, preferably end-to-end, but at least for the data that is transferred over the Internet.
- Lossless data compression to improve throughput. A related feature called differential compression transfers only binary data that has changed since the last backup.
- Automated, scheduled backups.
- Fast backup (snapshots) after full online backup, with 10 to 30 historical versions of a file retained.
- Data versioning with the ability to retrieve historical versions of files from different backups.

- Multiplatform support. The most important clients to back up are Windows, Macintosh, and Linux/Unix.
- Bare file/folder restore.
- Adequate bandwidth and perhaps scalable bandwidth options to which to upgrade.
- Web-based management console with ease-of-use features such as drag and drop, e-mail updates, and file sharing.
- 24x7 technical support.
- Backed up data set validation; checking to determine if the backed up data matches the original data.
- Logging and reporting of operations.
- Open file backups of mission-critical transactional systems such as enterprise databases or e-mail/messaging applications.
- Multisite storage or replication, enabling data failover.

Table 15.2 lists some of the current backup services offered on unmanaged cloud storage.

Cloud attached backup

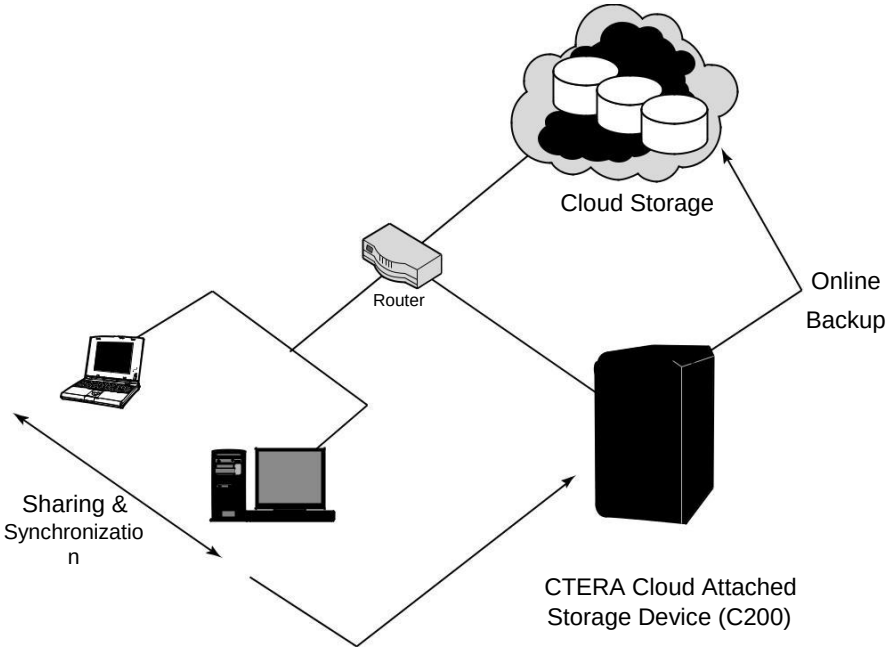
The backup solutions described have been client- or software-based solutions that are useful for an individual desktop or server. However, some interesting hardware-based solutions are available for backing up your systems to cloud-based storage.

CTERA (http://www.ctera.com/home/cloud-attached-storage.html) sells a server referred to as Cloud Attached Storage, which is meant for the Small and Medium Business (SMB) market, branch offices, and the Small Office Home Office (SOHO) market.

The CTERA Cloud Attached Storage backup server has the attributes of a NAS (Network Attached Storage), with the added feature that after you set up which systems you want to back up, create user accounts, and set the backup options through a browser interface, the system runs automated backup copying and synchronizing of your data with cloud storage. Backed up data may be shared between users. Figure 15.7 shows how the CTERA Cloud Attached Storage Device is deployed in practice.

CTERA cloud backup provides a solution that optimizes the backup based on bandwidth availabil-ity. It performs incremental backups from the server, compressing and encrypting the data that is transmitted to CTERA's cloud storage servers where de-duplication is performed. The CTERA server performs the backups of clients without requiring any client-based software. Clients have browser-based access to the backups or can locally access files using CTERA's "Virtual Cloud Drive" network drive. Snapshots are captured on the CTERA Next3 file system, which is based on the open source Ext3 file system.

CTERA's cloud-attached storage network backup scenario



The development of systems on a chip has enabled CTERA to create a scaled-down version of the CTERA server called the CTERA CloudPlug for the SOHO market. This palm-sized low-power device converts a USB/eSATA drive and your Ethernet network and turns the hard drive into a NAS server. CloudPlug performs backup and synchronization services and then performs auto-mated or on-demand backups and snapshots of your systems.

CloudPlug uses UPnP (Universal Plug and Play) and Bonjour to discover systems on the network and then installs a small agent on those systems. The software works with Microsoft Active Directory and allows for role-based user access. Among the protocols it supports are the Common Internet File Sharing (CIFS) used by Windows and Apple File Sharing (AFP) systems. It can back up NTFS, FAT32, EXT3, and the NEXT3 file systems. Laptops may be backed up from any loca-tion because they are assigned a roaming profile with dynamic IP support. The system also backs up locked files.

Cloud Storage Interoperability

Large network storage deployments tend to get populated by vendors who provide unique func-tionality for their systems by creating proprietary APIs for the storage hardware that they sell. This

problem exists for online network storage, Storage Area Networks (SANs), and to an even greater extent for cloud storage systems. Storage vendors have encouraged adoption of their proprietary APIs by making them "open," but cloud system vendors have not responded by making any single API an industry standard. The development of Open Source APIs from the Open Source commu-nity has only added more storage APIs to the mix.

Cloud Data Management Interface (CDMI)

An example of an open cloud storage management standard is the Storage Networking Industry Association's (SNIA; http://www.snia.org) Cloud Data Management Interface (CDMI). CDMI works with the storage domain model shown in Figure 15.8 to allow for interoperation between different cloud systems, whether on public, private, or hybrid cloud systems. CDMI includes commands that allow applications to access cloud storage and create, retrieve, update, and delete data objects; provides for data object discovery; enables storage data systems to commu-nicate with one another; and provides for security using standard storage protocols, monitoring and billing, and authentication methods. CDMI uses the same authorization and authentication mechanism as NFS (Network File System) does.

In the Cloud Data Management Interface (CDMI), the storage space is partitioned into units called containers. A container stores a set of data in it and serves as the named object upon which data service operations are performed. The CDMI data object can manage CDMI containers, as well as containers that are accessible in cloud storage through other supported protocols.

Figure 15.8 shows the SNIA cloud storage management model. In the figure, XAM stands for the eXtensible Access Method, a storage API developed by SNIA for accessing content on storage devices. VIM stands for Vendor Interface Modules, which is an interface that converts XAM requests into native commands that are supported by the storage hardware operating systems.

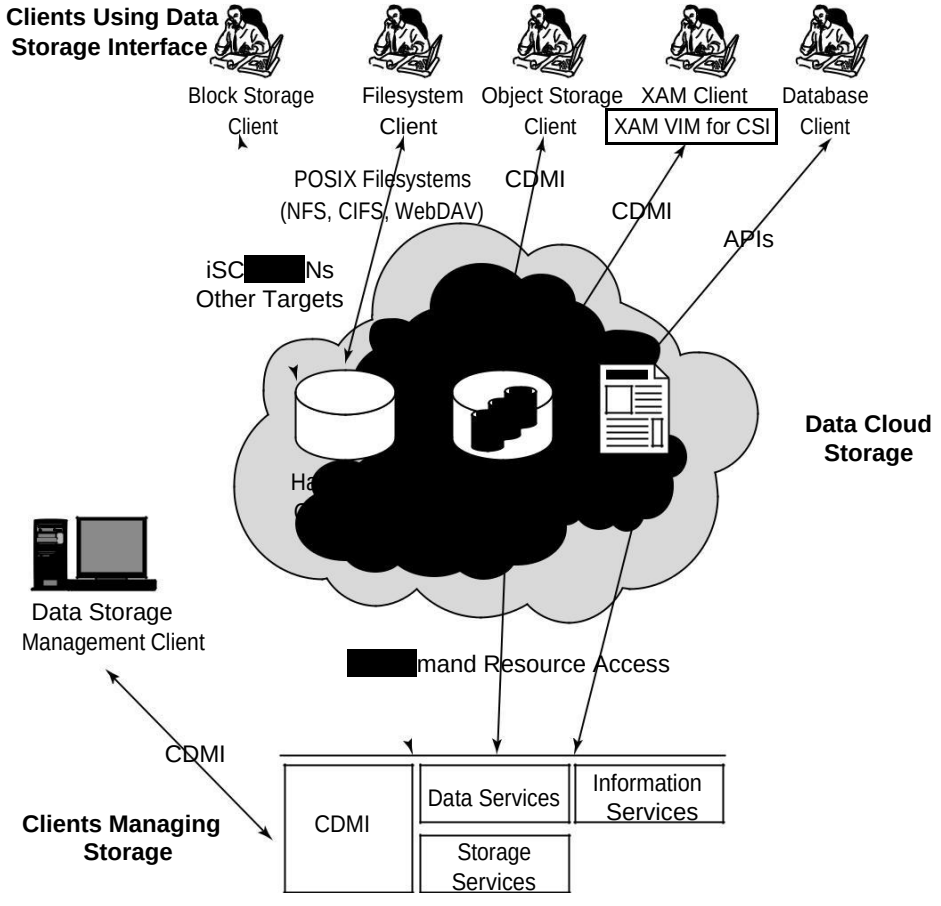
CDMI can access objects stored in the cloud by using standard HTTP command and the REST (Representational State Transfer) protocol to manipulate those objects. CDMI also can discover objects and can export and manage those exported objects as part of a storage space called a con-tainer. CDMI provides an interface through which applications can gain access to the storage objects in a container over the Web. Other features of CDMI are access controls, usage accounting, and the ability to advertise containers so that applications see these containers as if they are vol-umes (LUNs with a certain size).

CDMI uses metadata for HTTP, system, user, and storage media attributes accessing them through a standard interface using a schema that is known as the Resource Oriented Architecture (ROA). In this architecture, every resource is identified by a standardized URI (Uniform Resource Identifier) that may be translated into both hypertext (HTTP) and other forms. CDMI uses the SNIA eXtensi-ble Access Method (XAM) to discover and access metadata associated with each data object.

Metadata is stored not only for data objects, but for data containers so that any data placed into a container assumes the metadata associated with that container. Should there be conflicting meta-data at different levels of the hierarchy (container, object, and so on), the most granular level object's metadata attribute takes precedence.

FIGURE 15.8

CDMI allows data in cloud storage to be managed from a variety of resources.



Source: "Cloud Storage for Cloud Computing" SNIA/OGF, September 2009, http://ogf.org/Resources/documents/CloudStorageForCloudComputing.pdf.

In CDMI, resources are identified as nouns, which have attributes in the form of key-value pairs, upon which actions in the form of verbs may be performed. Standard actions include the standard CRUD operations: Create, Retrieve, Update, and Delete; which translates into the standard HTTP action verbs POST, GET, PUT, and DELETE. Additionally, the HEAD and OPTIONS verbs provide a wrapper for metadata and operational instructions.

A typical action might be a PUT or GET operation, as follows:

PUT http://www.cloudy.com/store/<*myfile*>
GET http://www.cloudy.com/compute/<*myfile*>

The domain cloudy.com would be the service provider, *myfile* is the instance, and compute is the folder containing the file. In a PUT operation, the container (*store*) is created if it didn't

exist previously. The metadata KEY/VALUE pair MIME is required in a PUT; other metadata KEY/ VALUE pairs are optional in a PUT. A variety of KEY/VALUE pairs describing object attributes in CDMI is defined by the standard.

Open Cloud Computing Interface (OCCI)

SNIA and the Open Grid Forum (OGF; http://www.ogf.org/) have created a joint working group to create the Open Cloud Computing Interface (OCCI), an open standard API for cloud computing infrastructure systems. OCCI is meant to span the different vendors' standards and allow for system interoperability.

Note

To view the Cloud Standards Wiki with information about all the different standards groups working in this area of technology, go to: http://cloud-standards.org/wiki/index.php?title=Main_Page**.**
This page contains links to the work of groups in cloud storage, virtual machines, protocols, and more. ■

The OCCI interface standard is based on the Resource Oriented Architecture (ROA) and uses the URI definition for OCCI that was previously defined by SNIA's Cloud Data Management Interface (CDMI) that OCCI interoperates with CDMI. Associations between resources appear in the HTTP header in the Atom Publishing Protocol (AtomPub or APP) that transfers the Atom Syndication Format (XML) used for XML Web news feeds. The OCCI API maps to other formats such as Atom/ Pub, JSON, and Plain Text.

OCCI specifies, but does not mandate, what is called a *service life cycle.* In a service life cycle, a cli-ent (service requestor) instantiates or invokes a new application and through OCCI commands provisions its storage resources, manages the application's use, and then manages the application's destruction and the release of its cloud storage.

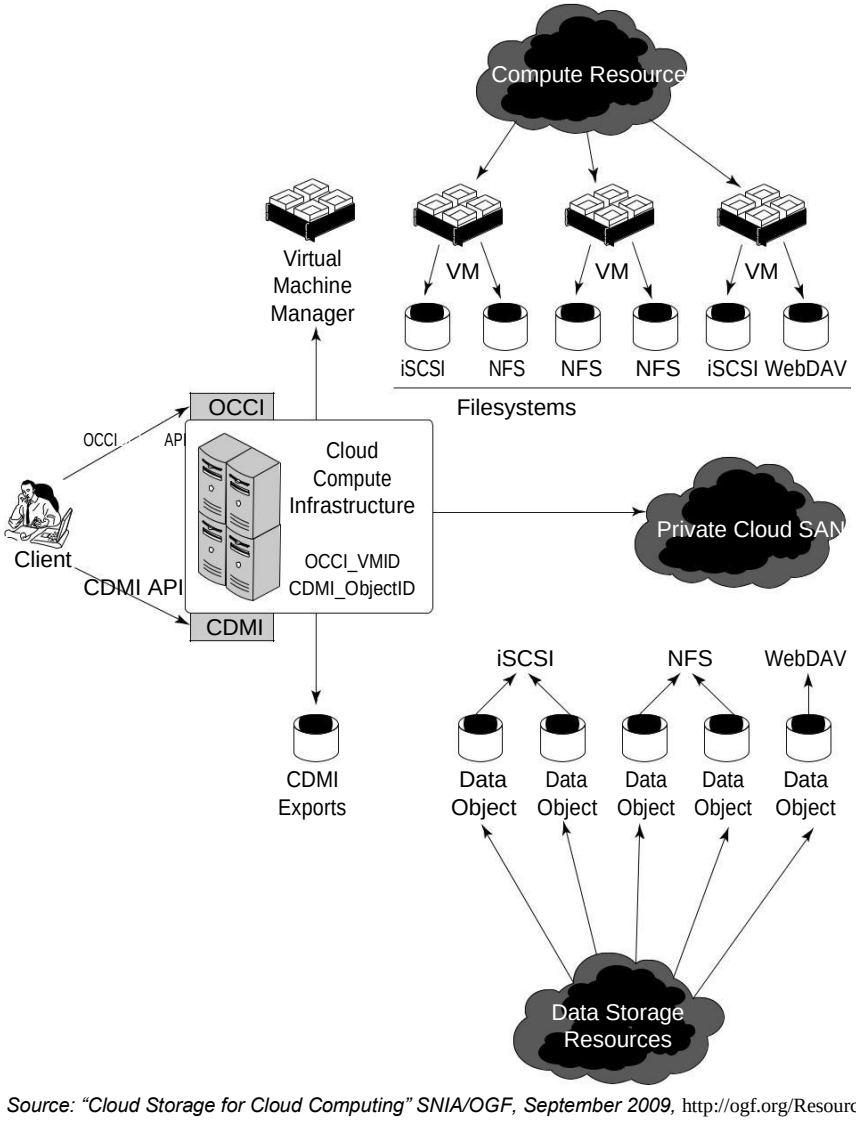
Cloud storage devices can be either a block or file system storage device, and in that regard they are no different than online network storage devices or even local storage. It is the ability to provide storage on a demand basis and pay as you go that is the key differentiator for cloud storage. The ability to provide storage on demand from a storage pool is referred to as *thin provisioning,* a term that also applies to compute resources such as virtual machines. Management of cloud storage is performed by *out-of-band* management systems through a data storage interface. Out-of-band refers to a management console that isn't on the storage network, but is most often on an Ethernet network inside a browser. From the management console, additional data services such as cloning, compression, de-duplication, and snapshots may be invoked.

As previously mentioned, CDMI and OCCI are meant to interoperate, and CDMI containers can be accessed through a data path and over other protocols. A CDMI container can be exported and then used as a virtual disk by Virtual Machines in the cloud. The cloud infrastructure management console can be used to attach exported CDMI containers to the Virtual Machine that is desired. CDMI exports containers so the information that is obtained from the OCCI interface is part of the exported container.

OCCI also can create containers that are interoperable with CDMI containers. These export operations can be initiated from either the OCCI or CDMI interfaces, with similar results. However, there are syntactical differences between using either interface as the export starting point. In Figure 15.9, CDMI and OCCI are shown interoperating with cloud resources of different types.

FIGURE 15.9

CDMI and OCCI interoperating in an integrated cloud system



Source: "Cloud Storage for Cloud Computing" SNIA/OGF, September 2009, http://ogf.org/Resources/documents/CloudStorageForCloudComputing.pdf.

Summary

In this chapter, you learned about the nature of stored digital data and the role that cloud storage will play in the future in storing and processing data.

Cloud storage is classified as either unmanaged or managed storage. Most user applications work with unmanaged storage. The two major classes of cloud-based storage applications described in this chapter were file sharing and backup utilities. Managed storage is cloud storage that you provi-sion for Web services or applications using cloud storage that you are developing. Managed storage requires you to prepare the disk and manage its use.

All cloud storage vendors partition storage on the basis of a virtual storage container. A model describing the virtual storage container is described. Efforts to make cloud storage systems interop-erate, particularly the CDMI and OCCI protocols, were described.

In Chapter 16, “Working with Productivity Software,” I consider desktop applications that replace office suite applications. These applications have the potential to displace a major portion of their commercial shrink-wrapped software counterparts over time, and while not as feature-filled as commercial software, they are surprisingly good. The chapter discusses the state of the art in this area and makes some predictions of what to expect over time.

Using Webmail Services

This chapter describes two of the most popular Web services that are deployed in the cloud: Webmail and syndicated content. Webmail sites are among the most popular Web sites in use today with the major services having hundreds of millions of user accounts.

Many Webmail services are free, and the rest are generally modestly priced. Webmail may be differentiated from hosted e-mail by its access through a Web browser. This makes them platform-independent. These services also sometimes use POP3 and IMAP, which allows them to be used to feed e-mail into traditional e-mail clients like Outlook and Thunderbird.

The current generation of Webmail services implements user interfaces based on Ajax and tends to follow a model that makes them look similar to the Microsoft Outlook e-mail client. These browser-based services provide filters, advanced search capabilities, sorting, tagging, and many other features. Most of these services use spam and virus detection to eliminate unwanted mail.

Syndication services are a method for publishing content from Web sites, blogs, wikis, and other services. It is a form of group e-mail, broadcast e-mail if you will.

There are both content providers and content consumers. Examples of content providers include not only the services just mentioned but also aggregation services and site. Many aggregation Web sites are well known for collecting content on the subject area in which they specialize.

RSS content can be read in most modern browsers. When you subscribe to an RSS feed, you create a bookmark or favorite that is updated automatically as new content appears. Another type of application called a newsreader allows you to collect subscriptions and view them all in one site. The iGoogle customization site is based on RSS and Atom news feeds.

Exploring the Cloud Mail Services

By any measure, browser-based hosted e-mail or "Webmail" is one of the great success stories of the Internet. It is the prototypical Software as a Service (SaaS) application. Webmail was also one of the first cloud computing applications to emerge and is today among the most heavily used services. Webmail is differentiated from hosted e-mail primarily by the use of browser-based client access. The underlying e-mail servers and the mail protocols are the same ones used for client/ server e-mail services, but the servers and services have been deployed on a massive scale.

Many of these services such as Gmail and Hotmail are free up to a certain level of service; it is certainly the price that has attracted such a large worldwide audience. When you layer on top of low price all the advantages that cloud computing offers—scale, ubiquitous access, platform independence, and others—it is not hard to understand Webmail's allure.

The first of the free hosted Webmail services to emerge was Hotmail. It was begun in 1996 by Sabeer Bhatia and Jack Smith with the name HoTMaiL. The capitalization indicated its origin as a Web- or HTML-based application. Microsoft acquired HotMail a year later and rebranded it as MSN Hotmail. The current version of the product is called Windows Live Hotmail, and it is part of the Windows Live suite of Web-based software products discussed in Chapter 10, "Using Microsoft Web Services."

It's anyone's guess who has the largest Webmail service. Based on the number of registered accounts, that honor would seem to be accorded to Microsoft, which has 360 million Hotmail accounts. Yahoo Mail! also claims to be the largest Webmail service with 260 million accounts. Google's Gmail by comparison has 176 million accounts as of the end of 2009, and AOL Mail (also called AIM Mail) is believed to be the fourth largest Webmail service.

Note

It is common practice among Webmail services to flag dormant accounts after a certain period of time and then to delete the accounts at some later time, should there be no activity. For Gmail and Hotmail, those actions occur after six and nine months, respectively. Yahoo! Mail deactivates accounts after only four months of inactivity. ■

Accounts are one thing; active use of accounts is another. A much more accurate picture of how popular these services are may be obtained by examining the number of visits (hits) that the different Web sites get. A *hit* can be measured relatively accurately by looking at the DNS server logs at key points in the Internet backbone.

The Internet data analytics company Experian's Hitwise.com service maintains a dashboard (http://www.hitwise.com/us/datacenter/main/dashboard-10133.html) shown in Figure 17.1 with the current percentage of hits made on individual sites. As of the week of 7/17/2010, these were the three top Web sites:

- 1. Facebook (9.16%)
- 2. Google (7.45%)
- 3. Yahoo! (3.76%)

Webmail services occupy these slots:

- 4. Yahoo! Mail (3.59%)
- 8. Windows Live Mail (1.60%)
- 11. Gmail (0.87%)
- 14. AOL Mail (0.59%)

These figures give a much more accurate picture of how important the different services are in real-world usage.

FIGURE 17.1

Experian's Hitwise.com site publishes a dashboard with the leading Web sites by different categories.

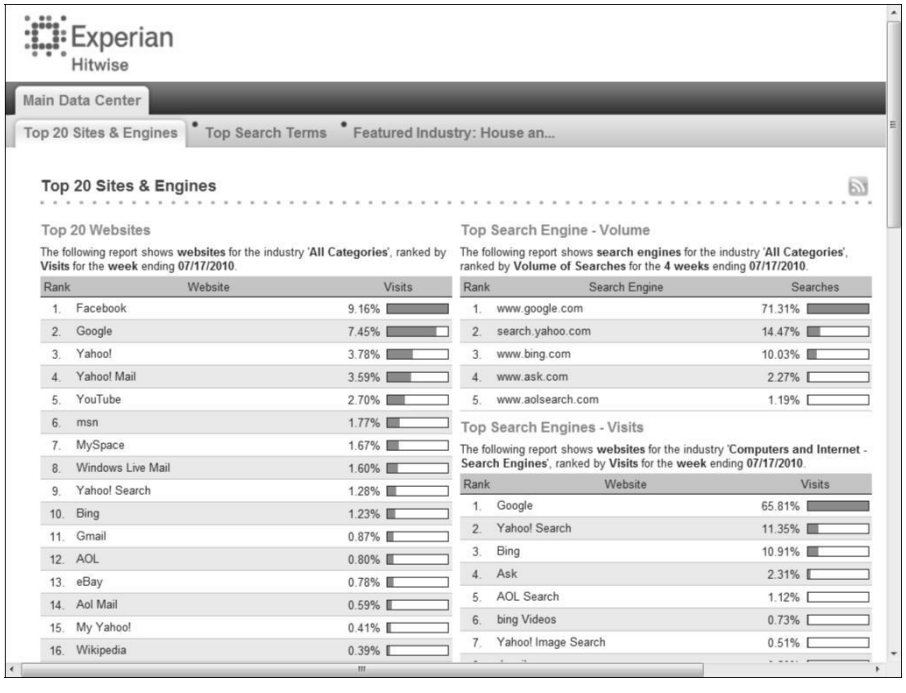


Table 17.1 summarizes the features of leading Webmail providers. The following sections describe the three largest Webmail services: Gmail, Hotmail, and Yahoo! Mail in detail.

TABLE 17.1

Webmail Features

Service/Owner	URL	Cost (U.S. $)	Storage/Max Attachment	IMAP or POP3 Support
AOL Mail AOL USA	https://mail.aol.com/	Free	Unlimited/25MB	Both
BlueTie BlueTie, Inc.	http://www.bluetie.com/	$4.99/mo	10GB/25MB	Both
ContactOffice Contract Office Group sa	http://www.contact office.com/	Free and paid	100MB to 10GB/25MB	Both
Excite IAC Search & Media	http://www.excite.com/	Free	1GB/25MB	No
FastMail.FM Opera Software Australia	http://www.fastmail.fm/	Free, $4.95/yr ad free, $34.95 enhanced	25MB to 15GB/10 to 50MB	IAMP for all, POP3 and SMTP paid
Gawab EgyptHome IT, Egypt	http://www.gawab.com/	Free	10GB/50MB	Both
Gmail Google USA	https://www.google.com/	Free	7.48GB/25MB	IMAP, POP3, POP+TLS, Microsoft Exchange
GMX Mail United Internet, Germany	http://www.gmx.net/	Free, paid for ProMail and TopMail	1 free or 5GB paid/20 to 50MB	Both (IMAP paid)
Hushmail Hush Communications Ltd.	http://www.hushmail.com/	Free, $34.99 premium	2MB free to 5GB premium/2MB free to 1GB premium	IMAP paid
Lavabit Lavabit LLC, USA	http://lavabit.com/	Two free options, and two paid options	128MB/32 to 128MB	Both (SSL optional)

Service/Owner	URL	Cost (U.S. $)	Storage/Max Attachment	IMAP or POP3 Support
LuxSci Lux Scientiae, Inc. USA	http://luxsci.com/	$9.99/mo	2GB+/100MB	IMAP, IMAP+SSL, IMAP+TLS, POP, POP+SSL, POP+TLS, Alternate Ports
Lycos Duam, Korea	http://www.lycos.com/	$19.95/yr	5GB/Unlimited	NA
Mail.com MMC, USA	http://www.mail.com/	Free, $3.99/mo or $19.99/ yr ad free	Unlimited/16MB	Free: None, Paid Both
Mail.ru Mail.ru, Russia	http://mail.ru/	Free, $5/ mo ad free	Unlimited/30MB	Both
Mail2World Mail2World, Inc. USA	http://www.mail2world.com/	Free, $19.95 premium	Unlimited/40MB	Both
MobileMe Apple, Inc. USA	http://www.mobileme.com	$99/yr	20GB/Unlimited	IMAP (POP3 optional
MyWay IAC Search & Media USA	http://www.myway.com/	Free	1GB/25MB	No
O2 Webmail Telefonica O2 UK	http://www.o2.co.uk/	Free	20MB/NA	POP3 only
Ovi Mail Nokia Finland	http://www.ovi.com/	Free	1GB/20MB	NA
Rackspace Email Rackspace USA	http://www.rackspace.com/	$1/mail-box month	10GB/50MB	Both
Runbox Runbox AS, Norway	http://www.runbox.com/	$49.95/yr	10GB/100MB	IMAP4 and POP3 with SSL
Seznam.cz Seznam.cz	http://www.seznam.cz/	Free	Unlimited/13MB	POP3
ThinkPost.net Thinkpost	http://www.thinkpost.net/	$5/mo	10GB/50MB	Both
Windows Live Hotmail Microsoft, USA	http://www.mail.live.com/	Free, $19.95 ad free	5GB/10 to 20MB	POP3

continued

TABLE 17.1 *(continued)*

Service/Owner	URL	Cost (U.S. $)	Storage/Max Attachment	IMAP or POP3 Support
WWW.COM Email WWW.COM	https://mail.www.com/web-email	$28.99	7GB/25MB	Both (w/SSL option)
Yahoo! Mail Yahoo!	http://mail.yahoo.com/	Free, $19.99/yr for Plus	Unlimited/25MB	POP3 in most countries, with Plus), or with YPOPs!

Source: http://en.wikipedia.org/wiki/Comparison_of_webmail_providers.

Google Gmail

Google Gmail (http://www.gmail.com or alternatively http://www.mail.google.com) is the third most popular of the large Webmail services and became available first in beta in 2004 from Google Labs and then for the use of the public in 2007 as "beta." In July 2009 Google announced that Gmail and the Google apps were released products. Gmail is available worldwide in 52 languages, works in nearly all modern browsers, and comes in both a desktop and mobile browser version. Google markets Gmail for domains to organizations and has a Google Apps Partner Edition that Google allows to be branded by ISPs, large organizations, port Web sites, and other organizations as paid services.

Gmail is written to look like an Internet chat utility, as shown in Figure 17.2, and it sorts e-mails by conversations or threads. Conversations containing multiple e-mails can be edited to delete individual messages, but Gmail tends to perform most of its operations such as archiving on the conversation as a whole. A conversation cannot be split into multiple conversations, nor can a conversation be added to another conversation. When conversations get large, as they do with group e-mails, Gmail can be cumbersome to work in. When a conversation gets to be 100 messages long, Gmail splits the conversation into a second section.

At the time of its offering, Gmail created something of a sensation by offering 1GB of free storage when competitors would allow their free customers to have only a few megabytes of storage for their accounts. Today Gmail's free accounts come with up to 7.48GB of free storage. Figure 17.2 shows a Gmail screen for a new user account.

Gmail was notable for its early use of Ajax (Asynchronous JavaScript and XML), something that has become the standard development platform for Webmail applications across the industry. When you compose a message in Gmail's Rich Text Format interface, Gmail performs an Autosave of its content at 1-minute intervals.

The product is extensible both through a number of add-ons from Gmail Labs and through its multi-tabbed settings page (https://mail.google.com/mail/?shva=1#settings), shown in Figure 17.3. As these add-ons (or experiments if you will) are tested and mature, some of them make their way into Gmail's default setup. Among the features to have graduated into Gmail are the integrated chat with SMS messaging that you see in the lower left of Figure 17.3 (Google Talk, which is discussed in the next chapter), offline access using Google Gears, and the Tasks feature.

FIGURE 17.2

The Gmail service presents an interface that is reminiscent of an Internet chat utility.

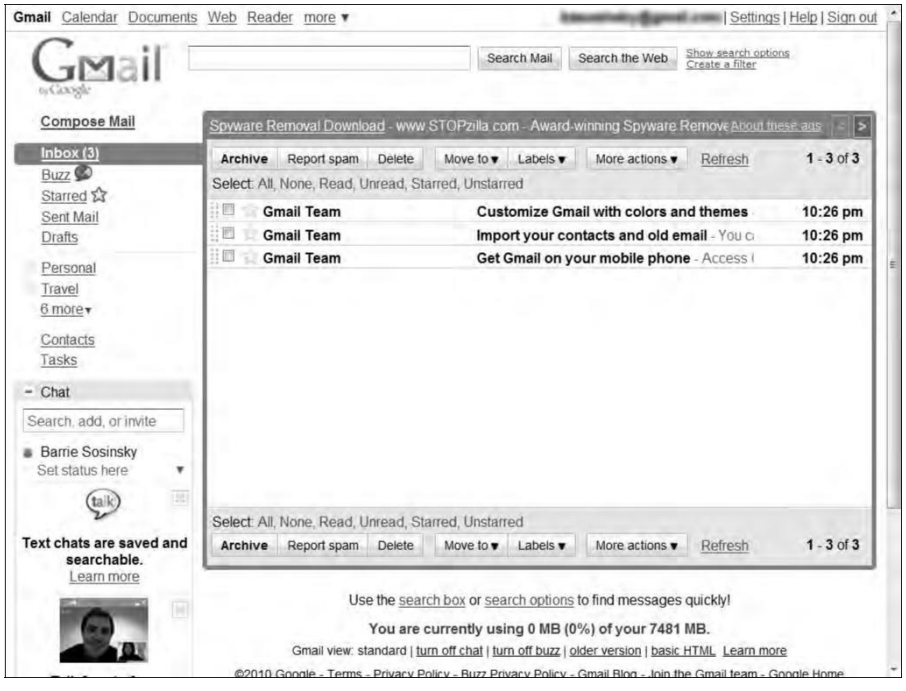
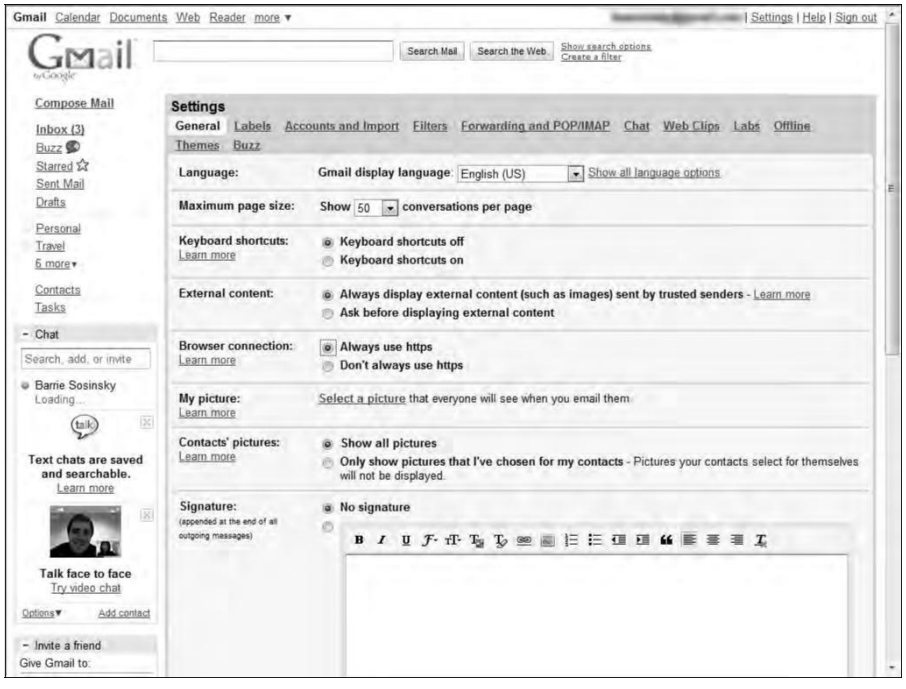


FIGURE 17.3

Gmail's General settings page



Although many e-mail services use spam filters based on Bayesian algorithms, the developers of Gmail opted for a system where the entire user base's assignment of spam is used to grade and mark e-mail as potential spam. You can set your own flag for whether any particular e-mail or sender is spam overriding Gmail's setting. Gmail also scans both incoming and outgoing e-mail attachments for viruses and blocks the receipt of any file that it recognizes as an executable file.

In Gmail, you can construct searches with multiple operators using the Advanced Search feature, which accepts keywords, sender, location, and date. A number of additional search criteria, such as "language:<*languagename*>," can be used to narrow a search. When you set flags on messages, those flags can be used in filters to narrow what you see in the window. Some flags are set by performing operations such as reading the message, archiving, and so forth.

Google uses an advertiser-driven model to supply its free service to users. In its search function, Google uses the keywords from your search and your search history to match sponsors to you. In

Gmail, Google scans the content of the e-mail and extracts keywords from your messages. This has raised the concern that mail sent through Gmail isn't as private as some advocates might like. Although you have agreed to Google's privacy policy, the people sending you e-mail have not.

In one of the General settings shown earlier in Figure 17.3, you can set an option to force Gmail to use the HTTPS transfer protocol instead of the non-secure HTTP protocol. This is the default setting in the current version of the program. For POP3 and IMAP access to your Gmail account through a mail client, the transport protocol is TLS (Transport Layer Security). Not all e-mail clients receive mail from Gmail's servers using TLS; some, such as Thunderbird, get the message transmitted to them over the wire as clear text. Using the Mail Fetcher feature, up to five POP3 accounts can be automatically retrieved and displayed within a Gmail account.

These are the Gmail POP3 settings:

- **POP server address:** pop.gmail.com.
- **POP user name:** Your full Gmail address (including @gmail.com); Google Apps users may have to enter <Username>@your_domain.com>.
- **POP password:** Your Gmail password.
- **POP port:** 995.
- **POP TLS/SSL required:** Yes.

The fact that Google scans and stores e-mail for up to 60 days makes it a target for hackers. The recent kerfuffle concerning hackers gaining access to human rights activists' Gmail accounts in China led to Google moving its servers from Beijing to Hong Kong and tends to validate these concerns. However, Google isn't unique in the way it handles Webmail, and these concerns apply to nearly every service described in this chapter.

Mail2Web

Mail2Web is the prototypical POP3 Webmail mail retrieval service, established in 1997. You log into your e-mail account from a browser using your account name and password, and then Mail2Web queries your mail server and downloads the messages that are unread for display. From the Mail2Web interface, you can read messages, reply, and create new messages, as shown in Figure 17.4. The basic service is free, but the company based in Toronto, Canada, has additional paid services for hosted Microsoft Exchange accounts.

Mail2Web also has a mobile e-mail service based on Exchange called Mail2Web.com Mobile E-mail. The mobile service works with RIM Blackberry cell phones. The company's instant messaging service for mobile devices allows users to connect to their AOL, ICQ, MSN, and Yahoo! IM accounts.

FIGURE 17.4

Mail2Web.com provides online access to any POP3 account.

Windows Live Hotmail

Windows Live Hotmail is Microsoft's Webmail offering and with localized versions in 36 languages. The original version of the product was found at http:www.hotmail.com and was rebranded as MSN Hotmail. Today, that URL redirects you to the login page for Windows Live Hotmail at http://mail.live.com. After you establish an account with the domain address of either <*Username*>@hotmail.com or <*Username*>@live.com, you are directed to your Windows Live Hotmail inbox, and the welcome message is displayed, as shown in Figure 17.5.

Windows Live Hotmail is one of the central applications in Microsoft Windows Live product portfolio, and it's integrated with other Windows Live applications. As you can see in Figure 17.5, Hotmail provides one-click access to Windows Live Calendar, Contacts, Messenger, and Spaces. The last version of Hotmail was released in June 2010 and added further integration to Windows Live Office and to your Windows Live SkyDrive online storage.

Although the interface looks similar to an Outlook client in that it has a folder-based navigation tree (the left panel), Windows Live Hotmail was created with Ajax technology. The current version of Hotmail is compatible with Internet Explorer, Firefox, and Chrome, but not Safari.

A welcome message shown in Windows Live Hotmail, a central offering of Windows Live Services

Hotmail has a strong feature set. This includes the ability to navigate the interface with a keyboard, automatic completion of input fields, and strong contact management and group e-mail support. Built into the product is a spam filter and virus scanning. You can support multiple e-mail accounts in Hotmail, allowing the product to serve as a central repository of your e-mail from different services.

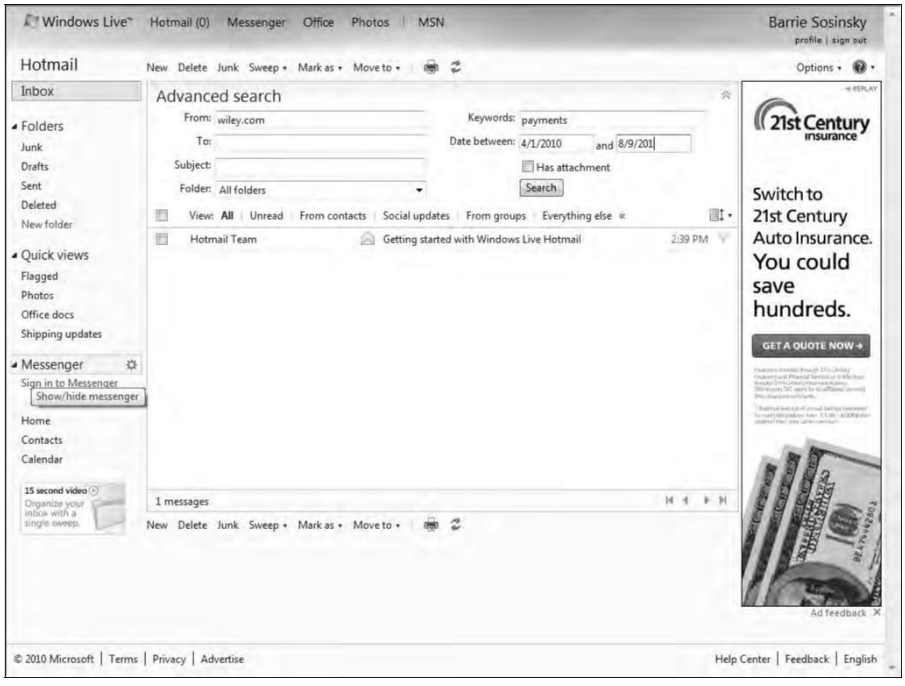
The newest version of Hotmail added the ability to set a spam filter directly with your mouse, called 1-Click Filters; to set a junk mail collection, called Inbox Sweeping; and to send attachments up to 10GB in size. Also new is a set of special content folders called Quick views, which can show messages you flag, display photos and Office documents, and show shipping update notices from shipping carriers.

One of the stronger features of Hotmail is its Advanced Search function, shown in Figure 17.6. You can search on addresses, domains, keywords, and dates, and you can perform a search scoped to different folders. The search terms you enter create a compound structured query.

You can get your Hotmail account using POP3 inside a traditional e-mail client or using a POP3 Web service such as Mail2Web.com.

FIGURE 17.6

Hotmail's Advanced Search creates a structured query across multiple fields.



The POP3 settings for Hotmail are as follows:

- **POP server:** pop3.live.com (Port 995).
- **POP SSL required:** Yes.
- **User name:** Enter your Hotmail account, <*Username*>@hotmail.com or <*Username*>@live.com.
- **Password:** Enter the password you use to log into Hotmail, or if you are using a Windows Live account, use that password.
- **SMTP server:** smtp.live.com (Port 25 or 587).
- **Authentication required:** Yes; this matches your POP username and password.
- **TLS/SSL required:** Yes.

Hotmail also can be viewed in Microsoft Office Outlook using the Outlook Connector or using Windows Live for Windows Mobile phones on that phone's operating system. The Microsoft Office Outlook Connector (for Outlook 2003, 2007, and 2010) lets you access messages, contacts, and your calendar from Outlook in your Hotmail account. If you have a premium subscription to

Hotmail, you also can access your notes and tasks using the connector. Windows Live Mail and Microsoft Outlook can synchronize messages with Hotmail using Microsoft's DeltaSync

protocol, but only on the Windows platform. Another synchronization feature called Exchange ActiveSync adds the ability to synchronize messages, contacts, and calendars on any mobile phone that has ActiveSync.

Yahoo! Mail

Yahoo! Mail, or as it is now rebranded "Y! Mail," is the largest Webmail service on the Internet. It is also one of the oldest, having appeared at the same time as Hotmail did in 1997. The original Y! Mail interface is still available for clients on older operating systems and other retro fan boys; it is shown in Figure 17.7.

In 2006, Yahoo! added a version of the user interface based on Ajax that looks like a form of Microsoft Outlook that has become something of an industry standard. This Ajax interface is based on the work of Oddpost, which the company acquired in 2004. Gmail also was heavily influenced by Oddpost's work. Figure 17.8 shows Y! Mail in the Ajax form.

FIGURE 17.7

Yahoo! Mail shown in the "Classic" interface format

FIGURE 17.8

The Ajax form of Y! Mail, referred to as the "All New Mail" option

Mail is meant to mimic a desktop client and has drag-and-drop capabilities, keyboard equivalents, tabbed widows, an advanced search, address auto-completion, and supports RSS feeds. You can send SMS messages to others in some countries using Y! Mail. Yahoo! Mail provides an unlimited amount of mail storage and a limit of 10MB per message and 25MB per attachment in the United States.

Y! Mail supports POP3 client access and mail forwarding. The service also runs a set of IMAP servers that you can access for free. The login to the service requires that a special command be sent, so you may need to either obtain a mail client that supports Y! Mail IMAP or make appropriate modifications to your own client. Versions of Mozilla Thunderbird and Mutt support the IMAP feature. In some countries, you can use Yahoo! SMTP server (smtp.mail.yahoo.com) to send messages.

These are the POP3 and SMTP settings for Yahoo! Mail:

- **Incoming Mail (POP3) Server:** pop3.mail.yahoo.com (use SSL, port: 995)
- **Outgoing Mail (SMTP) Server:** smtp.mail.yahoo.com (use SSL, port: 465, use authentication)

- **Account Name/Login Name:** Your Yahoo! Mail ID (your email address without the "@ yahoo.com")
- **Email Address:** Your Yahoo! Mail address (<*Username*>@yahoo.com)
- **Password:** Your Yahoo! Mail password

Yahoo! Mail has a filter feature that lets you create up to 100 filters per account. The paid versions of the service expand this limit to 200 filters. Those filters operate in addition to the built-in spam filter called SpamGuard that Yahoo! applies to its e-mail. SpamGuard works on messages before filters are applied. The system also flags some mail that it suspects of being spam to deferred delivery, a feature referred to as greylisting. While the details of greylisting aren't fully revealed, they probably work by seeing if additional mail of this type arrives in a certain period, and if not, the block is lifted.

Mail is authenticated by Y! Mail using DomainKeys, which is a service that verifies the DNS domain of the person sending the e-mail and establishes the messages integrity. DomainKeys is based on the IETF protocol call Identified Internet Mail, which was enhanced to create a new protocol called DomainKeys Identified Mail (DMIK; http://www.dkim.org/), so that DMIK now replaces the original version of DomainKeys. You can also filter messages and archive messages to your local drive. Y! Mail also allows a sender to send messages with other domains listed as the origin.

Y! Mail has been integrated with a number of Yahoo! Web services. Y! Mail is integrated with Yahoo! Messenger, so you can check your mail and connect with others using instant messaging. This integration has become a standard feature in Webmail offerings; both Gmail and Hotmail offer this feature. Yahoo! Messenger can exchange messages with Windows Live Messenger. You also can access Yahoo! Calendar from within the program. Some other applications that are accessible from Y! Mail are Flickr, Piknic, and Wordpress.

Yahoo! offers a premium subscription version of Y! Mail called Yahoo! Mail Plus. Another service called Yahoo! Business E-mail provides Webmail, POP3, IMAP, and SMTP services along with 10 accounts of the Plus version for a $25 setup charge with a $9.99 monthly subscription. You also can register accounts with two other Yahoo! e-mail service domains: ymail.com and rocketmail. com.

Working with Syndication Services

A syndication service is another way for people to send messages to a group of people; it's a form of published e-mail. To receive syndicated content, you must opt into the system and subscribe to the "feed" from one of the many content management system services. You can read RSS and Atom formatted content inside special applications called newsreaders, or as they are called more often "readers," as well as in many Webmail applications. After you subscribe to a feed, the reader uses the link provided to download content from a site that you haven't downloaded already.

The technology behind syndication is simple, but the value of the content can be impactful. It's a shame that more people don't make better use of this free Web service, although Web service providers have proved themselves to be creative consumers of syndication. Your personalized Google home page can be altered to include information from any of thousands of feeds, serving the role of a reader or aggregator application, and with Ajax you can rearrange pages of feeds into various channels. Web feeds also are used to follow blogger and wikis entries.

Note

RSS and Atom are prototypes of a class of XML specifications called syndication markup languages. ■

The RSS and Atom Protocols

Two technologies are behind most of the syndicated content being used on the Internet: RSS and Atom. The first of these technologies, RSS, stands for Really Simple Syndication. A typical RSS document or feed contains text and metadata that can be used to indicate publication dates, authors, keywords, and more. RSS uses an XML file format and the concept of an RSS world or module. Several modules exist that are XML namespaces, including Ecommerce RSS 2.0, Media RSS 2.0, and OpenSearch RSS 2.0 modules. The major browser providers use the feed icon shown in Figure 17.9 to indicate that you can subscribe to content on that Web page; the icon applies equally to RSS and Atom content.

FIGURE 17.9

The RSS syndication feed icon used in today's browsers.

The format has a long history of development starting as far back as 1995 as an effort to summarize content of Web pages. Dave Winer and UserLand Software were instrumental in developing the format specification, which exists in two main forms: RDF (RSS 0.91, 1.0, and 1.1), and RSS 2.* (RSS 0.91, 0.92-0.94, and 2.0.1). Winer added the ability for subscription to include audio files in RSS feeds at the end of 2000. Most consumers of RSS can work with either branch of the standard. Currently, the RSS Advisory Board (http://www.rssboard.org/rss-specification) manages this format.

RSS feeds are the basis of podcasts that are carried on the Apple iTunes store and in many other locations and helped spark a revolution in media distribution (and for which this author is eternally grateful).

Because of the number of people involved in the development of RSS and the absence of a standards body endorsing it, an alternative version of XML syndication called the Atom Publishing Protocol was developed by the IETF. That standard was released as part of the Proposed Standard RFC 4287 (http://tools.ietf.org/html/rfc4287), with the Atom Publishing Protocol published as RFC 5023 (http://bitworking.org/projects/atom/rfc5023.html). Atom has some structural differences with RSS, but is similar in approach and technology.

Most of the major browsers support RSS and Atom, but some ask you to choose between them. Blog and wiki content tends to use Atom as the format. When you view a syndication content management application, the aggregators tend to list feeds by content. This is possible because a feed contains keywords in its metadata.

Questions

1. Discuss the basic concepts of message-based transactions. Explain the protocol stack for an SOA architecture.
2. Short notes:
    i. Event-driven SOA
    ii. Enterprise Service Bus
    iii. Service catalogs
3. Discuss the applications in the cloud in terms of the following:
    i. Cloud transactions
    ii. Functionality mapping
    iii. Application attributes
    iv. Cloud service attributes
    v. System abstraction and Cloud Bursting
    vi. Applications and Cloud APIs
4. Define cloud storage. Explain managed and unmanaged cloud storage.
5. Explain the Cloud mail services services in terms of the following:
    i. Google Gmail
    ii. Mail2Web
    iii. Windows Live Hotmail
    iv. Yahoo mail
6. Give the concepts of Syndication services