21UCA501 -Cloud Technology Fundaments

UNIT II.

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.

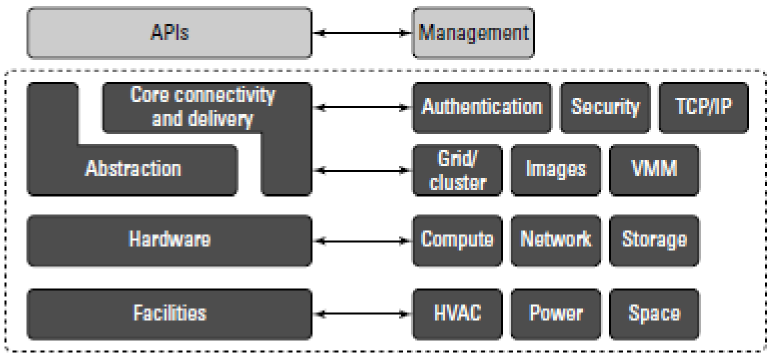
Figure 3.1 shows the portion of the cloud computing stack that is defined as the "server." In the diagram, the API is shown shaded in gray because it is an optional component that isn't always delivered with the server. The VMM component is the Virtual Machine Monitor, also called a hypervisor. This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



Platforms

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

Salesforce.com's Force.com Platform Windows Azure Platform Google Apps and the Google AppEngine These three services offer all the hosted hardware and software needed to build and deploy Web applications or services that are custom built by the developer within the context and range of capabilities that the platform allows. Platforms represent nearly the full cloud software stack, missing only the presentation layer that represents the user interface.

This is the same portion of the cloud computing stack that is a virtual appliance and is shown in Figure 3.2. What separates a platform from a virtual appliance is that the software that is installed is constructed from components and services and controlled through the API that the platform provider publishes.

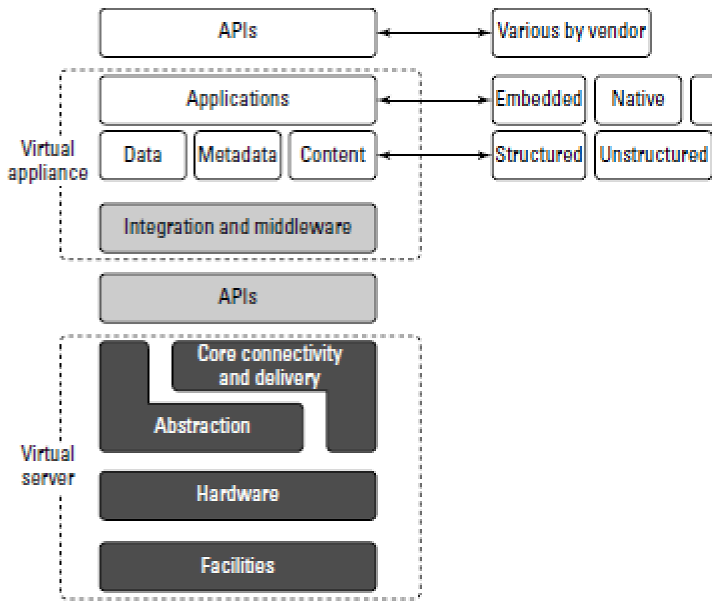
It makes sense for operating system vendors to move their development environments into the cloud with the same technologies that have been successfully used to create Web applications.

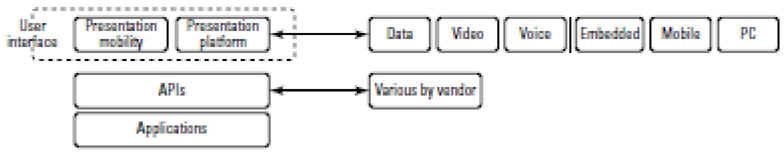
Thus, you might find a platform based on a Sun xVM hypervisor virtual machine that includes a NetBeans Integrated Development Environment (IDE) and that supports the Sun Glass Fish Web stack programmable using Perl or Ruby.

For Windows, Microsoft would be similarly interested in providing a platform that allowed Windows developers to run on a Hyper-V VM, use the ASP.NET application framework, support one of its enterprise applications such as SQL Server, and be programmable within Visual Studio—which is essentially what the Azure Platform does.

This approach allows someone to develop a program in the cloud that can be used by others.

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





Virtual Appliances

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

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

You run across virtual appliances in IaaS systems such as Amazon's Elastic Compute Cloud (EC2), which is discussed in detail in Chapter 9. Amazon Machine Images are virtual appliances that have been packaged to run on the gird of Xen nodes that comprise the Amazon Web Service's EC2 system.

Shown in Figure 3.4, the AMI library (http://developer.amazonwebservices.com/connect/kbcategory.jspa?categoryID=171) includes a variety of operating systems both proprietary and open source, a set of enterprise applications such as Oracle BPM, SQL Server, and even complete application stacks such as LAMP (Linux, Apache, MySQL, and PHP). Amazon has negotiated licenses from these vendors that are part of your per-use pricing when you run these applications on their servers.

Virtual appliances are far easier to install and run than an application that you must set up yourself. However, virtual appliances are also much larger than the application themselves would be because they are usually bundled with the operating system on which they are meant to run. An application that is 50 or 100MB might require a virtual appliance that is 500MB to 1GB in size.

Usually, when a virtual appliance is created, the operating system is stripped of all excess functionality that isn't required by the appliance, because the appliance is meant to be used as is.

Bagvapp (http://bagside.com/bagvapp/) offers virtual appliances, including ones based on Windows, all of which run on VMware Player.

HelpdeskLive (http://helpdesklive.info/download/VirtualBox%20VDI%20free%20images.html) offers various Linux distributions upon which you can build a virtual machine.

Jcinacio (http://www.jcinacio.com/) has Ubuntu appliances.

Jumpbox (http://www.jumpbox.com) offers open source virtual appliances installed by them as a managed service. Jumpbox offers virtual appliances for many applications including Bugzilla, DokuWiki, Drupal, Joomla!, Nagios, OpenVPN, PostgreSQL, Redmine, WordPress, and many others. Figure 3.6 shows the Jumpbox home page.

QEMU (http://www.qemu.org/) is a CPU emulator and virtual machine monitor.

Parallels (http://ptn.parallels.com/ptn) hosts a variety of appliances that includes Linux distros, server software, and other products.

ThoughtPolice (http://www.thoughtpolice.co.uk/vmware/) offers appliances based on a variety of Linux distributions.

VirtualBox (http://www.virtualbox.org/) is a virtual machine technology now owned by Oracle that can run various operating systems and serves as a host for a variety of virtual appliances.

Vmachines (http://www.vmachines.net/) is a site with desktop, server, and security related operating systems that run on VMware.

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 interposes communication (IPC) processes, many client/server protocols have been applied to distributed networking over the years. Various forms of RPC (Remote Procedure Call) implementations (including DCOM, Java RMI, and CORBA) attempt to solve the problem of engaging services and managing transactions over what is essentially a stateless network.

The first of the truly Web-centric RPC technologies was XML-RPC, which uses platform independent XML data to encode program calls that are transported over HTTP, the networking transport to which nearly everyone is connected.

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

WS-Addressing

WS-Discovery

WS-Eventing

WS-Federation

WS-MakeConnection

WS-Messaging

WS-MetadataExchange

WS-Notification

WS-Policy

WS-ResourceFramework

WS-Security

WS-Transfer

WS-Trust

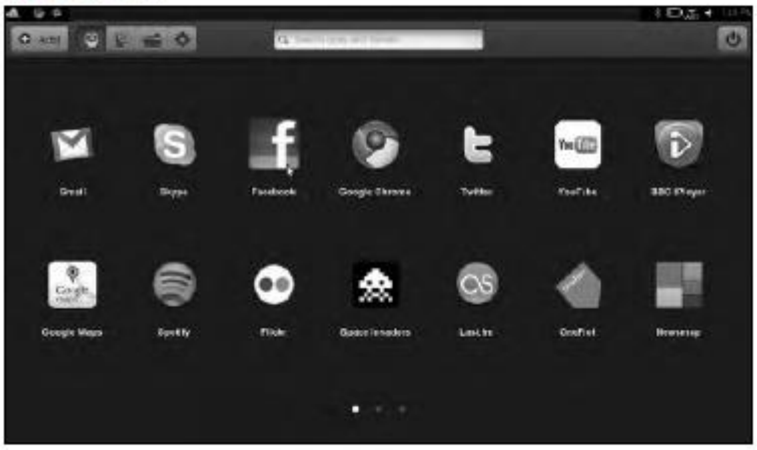
These different specifications provide a standard means of adding metadata to a SOAP message by modifying the message header while maintaining the message body structure. In this way, a standard method for metadata exchange is piggybacked onto the WSDL XML message. Each of these different WS-* specifications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

The Jolicloud Netbook OS

The popularity of ultralight netbooks and mobile phones has greatly expanded the potential audience of dedicated cloud computing devices, but until recently these devices ran standard operating systems such as Windows, Linux, and Macintosh on the PC and Android, IOS, and Windows Mobile (among others) on cell phones. The primary differentiation between these devices is whether or not they are capable of running video and animation (particularly Adobe Flash). None of these portable devices has been optimized to connect securely to the cloud by narrowing the operating system functions to harden these devices. The French firm Jolicloud (http://www. jolicloud.com/) has recently released a lightweight version of Linux designed specifically to run connected to the cloud as a dedicated cloud client. Jolicloud 1.0 ("The Anywhere OS") can be loaded onto a netbook as the only operating system, or it can be set up as a dual boot system that shares files with a Windows partition. Jolicloud concentrates on building a social platform with automatic software updates and installs.

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

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



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

Chromium OS: The Browser as an Operating System

The Google Chrome OS is a Linux open-source operating system designed to be a robust cloud client. Unlike many other Linux distributions, Google's Chrome is not a software installation, but is shipped installed on validated hardware from Google-approved OEMs (Original Equipment Manufacturers), just as the Android operating system is shipped on a variety of phones. The intent is to have a tightly coupled hardware offering that supports features in the Chrome OS and that would be highly efficient. Early designs have shown Chrome running on tablet designs that would position it as an Apple iPad competitor running on netbook-type devices in the $300-400 range. The expectation is that the first versions of Chrome systems will appear in late 2010, perhaps in both consumer and enterprise offerings. There is also an open-source version of this cloud client called Chromium (http://www.chromium.org/chromium-os), which shares the same code base. The Chromium architecture is built as a three-tier system with a hardware layer, the browser and window manager, and a set of system software and utilities.

The Chrome OS has been described as a hardened operating system because it incorporates a sandbox architecture for running applications and also performs automatic updates. Also included in the system is a version of remote desktop connection software that creates an encrypted connection like Microsoft's RDP, Citrix's ICA, or a VNC client. The Chrome OS hardware specification includes a Trusted Platform Module, which provides for a "trusted bootpath" along with a hardware switch that can be used to boot the system into a developer model. In that mode, some of the security features are turned off, allowing the user to reset the system. Demonstrations of early prototypes of Chrome and Chromium OS systems have shown that they are capable of nearly instantaneous startup. Chromium Linux kernel has adopted the Upstart (http://upstart.ubuntu.com/) event-based replacement for the init daemon, which is used to launch services concurrently, restore stalled jobs, and perform delayed system startup.The fast boot time is possible because the device is devoid of most of the devices in modern PCs.

Chromium has also adopted a set of security routines in firmware that run during startup and store the information necessary to perform verified system restoration. Essentially, the

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

page from *Alice in Wonderland,* displayed in Figure 3.10.

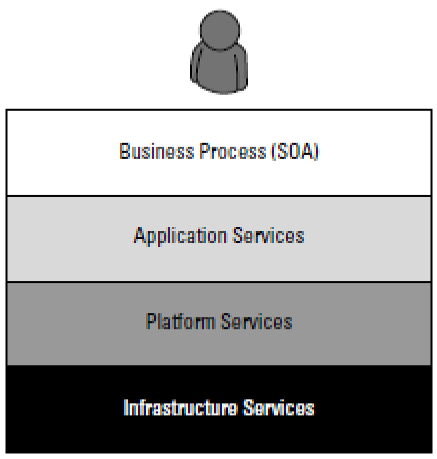
www.it-

Defining Infrastructure as a Service (IaaS)

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

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

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 virtualsystems, those requests are redirected to the real servers that do the actual work.

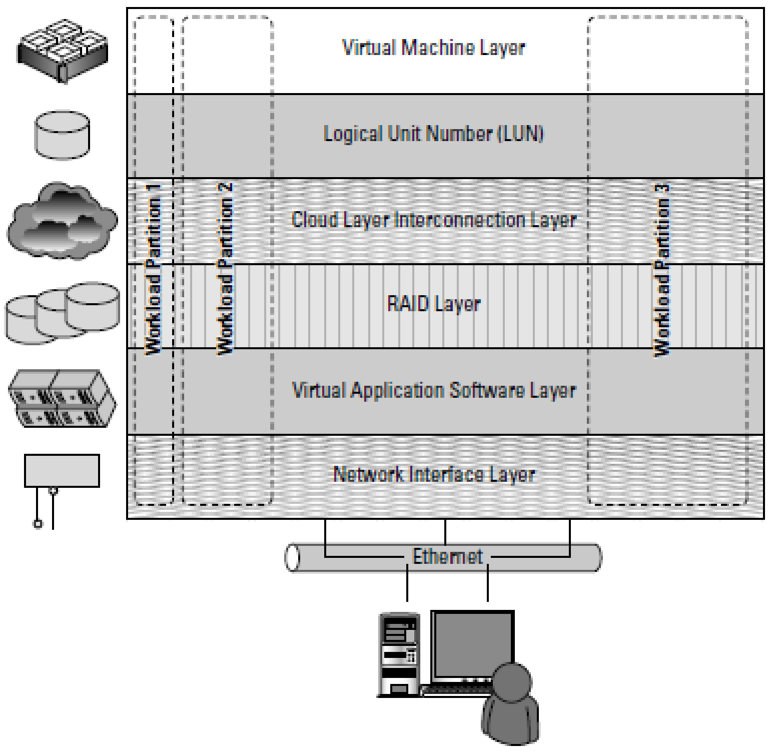
IaaS workloads

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

Figure 4.2 shows how three virtual private server instances are partitioned in an IaaS stack. The three workloads require three different sizes of computers: small, medium, and large. A client would reserve a machine equivalent required to run each of these workloads. The IaaS infrastructure runs these server instances in the data center that the service offers, drawing from a pool of virtualized machines, RAID storage, and network interface capacity. These three layers are expressions of physical systems that are partitioned as logical units. LUNs, the cloud interconnect layer, and the virtual application software layer are logical constructs. LUNs are logical storage containers, the cloud interconnect layer is a virtual network layer that is assigned IP addresses from the IaaS network pool, and the virtual application software layer contains software that runs on the physical VM instance(s) that have been partitioned from physical assets on the IaaS' private cloud. From an architectural standpoint, the client in an IaaS infrastructure is assigned its own private network. The Amazon Elastic Computer Cloud (EC2), described in detail in Chapter 8, behaves as if each server is its own separate network—unless you create your own Virtual Private Cloud (an EC2 add-on feature), which provides a workaround to this problem. When you scale your EC2 deployment, you are adding additional networks to your infrastructure, which makes it easy to logically scale an EC2 deployment, but imposes additional network overhead because traffic must be routed between logical networks. Amazon Web Service's routing limits broadcast and multicast traffic because Layer-2 (Data Link) networking is not supported. Rackspace Cloud (http://www. rackspacecloud.com/) follows the AWS IP assignment model. Other IaaS infrastructures such as the one Cloudscaling.com (http://www.cloudscaling. com) offers or a traditional VMWare cloud-assigned networks on a per-user basis, which allows for

Level 2 networking options. The most prominent Level 2 protocols that you might use are tunneling options, because they enable VLANs.

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.

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.