

Computing Clouds, Service-Oriented Architecture, and Programming

2

Three chapters in [Part 2](#) are devoted to cloud computing, including various cloud platforms for IaaS (infrastructure as a service), PaaS (platform as a service), and SaaS (software as a service) applications. We present service-oriented architectures developed in recent years. Parallel and distributed computing paradigms and their software support, language tools, and programming environments are covered for various cloud computing services.

CHAPTER 4: CLOUD PLATFORM ARCHITECTURE OVER VIRTUALIZED DATA CENTERS

This chapter covers the design principles and enabling technologies for cloud architecture and data center design. We start with data center design and management. Then we present the design principles of cloud platforms. We cover layered platform designs from infrastructure clouds including compute clouds, storage clouds, and application clouds. In particular, we study the center automation technology and interconnection networks. We apply virtualization support in resource provisioning and infrastructure management. Major public cloud platforms covered include Google AppEngine (GAE), Amazon Web Service (AWS), and Microsoft Windows Azure. The cloud security and trust management issues are studied.

This chapter is authored by Kai Hwang with partial contributions by Rajkumar Buyya and Kang Chen and the final manuscript was edited by Geoffrey Fox.

CHAPTER 5: SERVICE-ORIENTED ARCHITECTURES FOR DISTRIBUTED COMPUTING

We cover two major forms of distributed services—REST and web services—and the extension of the latter in Grid systems. Workflow is used to orchestrate or integrate multiple services and we describe the general approach used in workflow, the BPEL Web Service standard including Pegasus, Taverna, Kepler, Trident, and Swift. Example web interfaces or portals are described using Web Service (portlet) and Web 2.0 (gadget). We handle data and metadata in distributed systems using service registries and semantic web/grid.

This chapter is authored by Geoffrey Fox and Albert Zomaya with partial contributions by Rajkumar Buyya and technical assistance from Ali Javadzadeh Boloori, Chen Wang, Shrideep

Pallickara, Marlon Pierce, Suresh Marru, Michael McLennan, George Adams, III, Gerhard Klimeck, and Michael Wilde. The final manuscript was edited by Kai Hwang.

CHAPTER 6: CLOUD PROGRAMMING AND SOFTWARE ENVIRONMENTS

We introduce major cloud programming paradigms: MapReduce, BigTable, Twister, Dryad, DryadLINQ, Hadoop, Sawzall, and Pig Latin. We use concrete service examples to explain the implementation and application requirements in the cloud. We review core service models and access technologies. Cloud services provided by Google App Engine, Amazon Web Service, and Microsoft Windows Azure are illustrated by example applications. In particular, we illustrate how-to programming the GAE, AWS EC2, S3, EBS, and others. We review the open-source Eucalyptus, Nimbus, and OpenNebula and the startup Manjrasoft Aneka system for cloud computing.

This chapter is authored by Geoffrey Fox and Albert Zomaya with partial contributions by Rajkumar Buyya ([Section 6.5.3](#)) and Judy Qiu ([Section 6.2.6](#)). Special technical assistances are from Gregor von Laszewski, Javier Diaz, Archit Kulshrestha, Andrew Younge, Reza Moravaeji, Javid Teheri, and Renato Figueiredo. The final manuscript was edited by Kai Hwang.

Cloud Platform Architecture over Virtualized Data Centers

4

CHAPTER OUTLINE

| | |
|--|------------|
| Summary..... | 192 |
| 4.1 Cloud Computing and Service Models..... | 192 |
| 4.1.1 Public, Private, and Hybrid Clouds..... | 192 |
| 4.1.2 Cloud Ecosystem and Enabling Technologies..... | 196 |
| 4.1.3 Infrastructure-as-a-Service (IaaS)..... | 200 |
| 4.1.4 Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS)..... | 203 |
| 4.2 Data-Center Design and Interconnection Networks..... | 206 |
| 4.2.1 Warehouse-Scale Data-Center Design..... | 206 |
| 4.2.2 Data-Center Interconnection Networks..... | 208 |
| 4.2.3 Modular Data Center in Shipping Containers..... | 211 |
| 4.2.4 Interconnection of Modular Data Centers..... | 212 |
| 4.2.5 Data-Center Management Issues..... | 213 |
| 4.3 Architectural Design of Compute and Storage Clouds..... | 215 |
| 4.3.1 A Generic Cloud Architecture Design..... | 215 |
| 4.3.2 Layered Cloud Architectural Development..... | 218 |
| 4.3.3 Virtualization Support and Disaster Recovery..... | 221 |
| 4.3.4 Architectural Design Challenges..... | 225 |
| 4.4 Public Cloud Platforms: GAE, AWS, and Azure..... | 227 |
| 4.4.1 Public Clouds and Service Offerings..... | 227 |
| 4.4.2 Google App Engine (GAE)..... | 229 |
| 4.4.3 Amazon Web Services (AWS)..... | 231 |
| 4.4.4 Microsoft Windows Azure..... | 233 |
| 4.5 Inter-cloud Resource Management..... | 234 |
| 4.5.1 Extended Cloud Computing Services..... | 235 |
| 4.5.2 Resource Provisioning and Platform Deployment..... | 237 |
| 4.5.3 Virtual Machine Creation and Management..... | 243 |
| 4.5.4 Global Exchange of Cloud Resources..... | 246 |
| 4.6 Cloud Security and Trust Management..... | 249 |
| 4.6.1 Cloud Security Defense Strategies..... | 249 |
| 4.6.2 Distributed Intrusion/Anomaly Detection..... | 253 |
| 4.6.3 Data and Software Protection Techniques..... | 255 |
| 4.6.4 Reputation-Guided Protection of Data Centers..... | 257 |

| | |
|--|-----|
| 4.7 Bibliographic Notes and Homework Problems..... | 261 |
| Acknowledgements..... | 261 |
| References..... | 261 |
| Homework Problems..... | 265 |

SUMMARY

This chapter covers design principles, architectures, and enabling technologies of cloud platforms. We begin with a discussion of data-center design and management. Next, we present design choices for building compute and storage cloud platforms. We cover layered platform structure, virtualization support, resource provisioning, and infrastructure management. Several public cloud platforms are also studied, including Amazon Web Services, the Google App Engine, and Microsoft Azure. Subsequent chapters are devoted to service-oriented architectures, cloud computing paradigms, programming environments, and future cloud extensions.

4.1 CLOUD COMPUTING AND SERVICE MODELS

Over the past two decades, the world economy has rapidly moved from manufacturing to more service-oriented. In 2010, 80 percent of the U.S. economy was driven by the service industry, leaving only 15 percent in manufacturing and 5 percent in agriculture and other areas. Cloud computing benefits the service industry most and advances business computing with a new paradigm. In 2009, the global cloud service marketplace reached \$17.4 billion. IDC predicted in 2010 that the cloud-based economy may increase to \$44.2 billion by 2013. Developers of innovative cloud applications no longer acquire large capital equipment in advance. They just rent the resources from some large data centers that have been automated for this purpose.

In this and the next two chapters, we will study the cloud platform architecture, service models, and programming environments. Users can access and deploy cloud applications from anywhere in the world at very competitive costs. Virtualized cloud platforms are often built on top of large data centers. With that in mind, we examine first the server cluster in a data center and its interconnection issues. In other words, clouds aim to power the next generation of data centers by architecting them as virtual resources over automated hardware, databases, user interfaces, and application environments. In this sense, clouds grow out of the desire to build better data centers through automated resource provisioning.

4.1.1 Public, Private, and Hybrid Clouds

The concept of cloud computing has evolved from cluster, grid, and utility computing. Cluster and grid computing leverage the use of many computers in parallel to solve problems of any size. Utility and Software as a Service (SaaS) provide computing resources as a service with the notion of pay per use. Cloud computing leverages dynamic resources to deliver large numbers of services to end users. Cloud computing is a high-throughput computing (HTC) paradigm whereby the infrastructure provides the services through a large data center or server farms. The cloud computing model enables users to share access to resources from anywhere at any time through their connected devices.

Recall the introduction in [Chapter 1](#) in which we said that the cloud will free users to focus on user application development and create business value by outsourcing job execution to cloud providers. In this scenario, the computations (programs) are sent to where the data is located, rather than copying the data to millions of desktops as in the traditional approach. Cloud computing avoids large data movement, resulting in much better network bandwidth utilization. Furthermore, machine virtualization has enhanced resource utilization, increased application flexibility, and reduced the total cost of using virtualized data-center resources.

The cloud offers significant benefit to IT companies by freeing them from the low-level task of setting up the hardware (servers) and managing the system software. Cloud computing applies a virtual platform with elastic resources put together by on-demand provisioning of hardware, software, and data sets, dynamically. The main idea is to move desktop computing to a service-oriented platform using server clusters and huge databases at data centers. Cloud computing leverages its low cost and simplicity to both providers and users. According to Ian Foster [\[25\]](#), cloud computing intends to leverage multitasking to achieve higher throughput by serving many heterogeneous applications, large or small, simultaneously.

4.1.1.1 Centralized versus Distributed Computing

Some people argue that cloud computing is centralized computing at data centers. Others claim that cloud computing is the practice of distributed parallel computing over data-center resources. These represent two opposite views of cloud computing. All computations in cloud applications are distributed to servers in a data center. These are mainly *virtual machines* (VMs) in virtual clusters created out of data-center resources. In this sense, cloud platforms are systems distributed through virtualization.

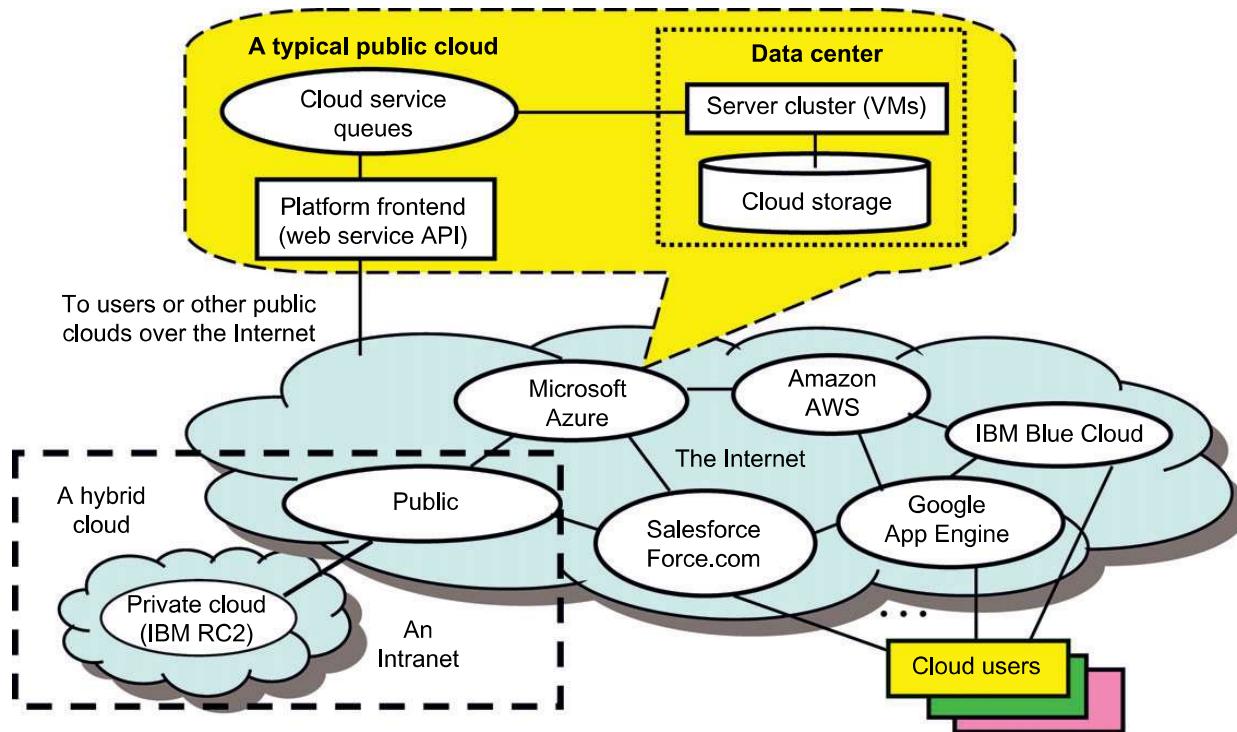
As [Figure 4.1](#) shows, both *public clouds* and *private clouds* are developed in the Internet. As many clouds are generated by commercial providers or by enterprises in a distributed manner, they will be interconnected over the Internet to achieve scalable and efficient computing services. Commercial cloud providers such as Amazon, Google, and Microsoft created their platforms to be distributed geographically. This distribution is partially attributed to fault tolerance, response latency reduction, and even legal reasons. Intranet-based private clouds are linked to public clouds to get additional resources. Nevertheless, users in Europe may not feel comfortable using clouds in the United States, and vice versa, until extensive *service-level agreements* (SLAs) are developed between the two user communities.

4.1.1.2 Public Clouds

A *public cloud* is built over the Internet and can be accessed by any user who has paid for the service. Public clouds are owned by service providers and are accessible through a subscription. The callout box in top of [Figure 4.1](#) shows the architecture of a typical public cloud. Many public clouds are available, including Google App Engine (GAE), Amazon Web Services (AWS), Microsoft Azure, IBM Blue Cloud, and Salesforce.com's [Force.com](#). The providers of the aforementioned clouds are commercial providers that offer a publicly accessible remote interface for creating and managing VM instances within their proprietary infrastructure. A public cloud delivers a selected set of business processes. The application and infrastructure services are offered on a flexible price-per-use basis.

4.1.1.3 Private Clouds

A *private cloud* is built within the domain of an intranet owned by a single organization. Therefore, it is client owned and managed, and its access is limited to the owning clients and their partners. Its deployment was not meant to sell capacity over the Internet through publicly accessible interfaces.

**FIGURE 4.1**

Public, private, and hybrid clouds illustrated by functional architecture and connectivity of representative clouds available by 2011.

Private clouds give local users a flexible and agile private infrastructure to run service workloads within their administrative domains. A private cloud is supposed to deliver more efficient and convenient cloud services. It may impact the cloud standardization, while retaining greater customization and organizational control.

4.1.1.4 Hybrid Clouds

A *hybrid cloud* is built with both public and private clouds, as shown at the lower-left corner of Figure 4.1. Private clouds can also support a hybrid cloud model by supplementing local infrastructure with computing capacity from an external public cloud. For example, the *Research Compute Cloud* (RC2) is a private cloud, built by IBM, that interconnects the computing and IT resources at eight IBM Research Centers scattered throughout the United States, Europe, and Asia. A hybrid cloud provides access to clients, the partner network, and third parties. In summary, public clouds promote standardization, preserve capital investment, and offer application flexibility. Private clouds attempt to achieve customization and offer higher efficiency, resiliency, security, and privacy. Hybrid clouds operate in the middle, with many compromises in terms of resource sharing.

4.1.1.5 Data-Center Networking Structure

The core of a cloud is the server cluster (or VM cluster). Cluster nodes are used as compute nodes. A few control nodes are used to manage and monitor cloud activities. The scheduling of user jobs

requires that you assign work to virtual clusters created for users. The gateway nodes provide the access points of the service from the outside world. These gateway nodes can be also used for security control of the entire cloud platform. In physical clusters and traditional grids, users expect static demand of resources. Clouds are designed to handle fluctuating workloads, and thus demand variable resources dynamically. Private clouds will satisfy this demand if properly designed and managed.

Data centers and supercomputers have some similarities as well as fundamental differences. We discussed supercomputers in [Chapter 2](#). In the case of data centers, scaling is a fundamental requirement. Data-center server clusters are typically built with large number of servers, ranging from thousands to millions of servers (nodes). For example, Microsoft has a data center in the Chicago area that has 100,000 eight-core servers, housed in 50 containers. In supercomputers, a separate data farm is used, while a data center uses disks on server nodes plus memory cache and databases.

Data centers and supercomputers also differ in networking requirements, as illustrated in [Figure 4.2](#). Supercomputers use custom-designed high-bandwidth networks such as fat trees or 3D torus networks (which we discussed in [Chapter 2](#)). Data-center networks are mostly IP-based commodity networks, such as the 10 Gbps Ethernet network, which is optimized for Internet access. [Figure 4.2](#) shows a multilayer structure for accessing the Internet. The server racks are at the bottom Layer 2, and they are connected through fast switches (S) as the hardware core. The data center is connected to the Internet at Layer 3 with many *access routers* (ARs) and *border routers* (BRs).

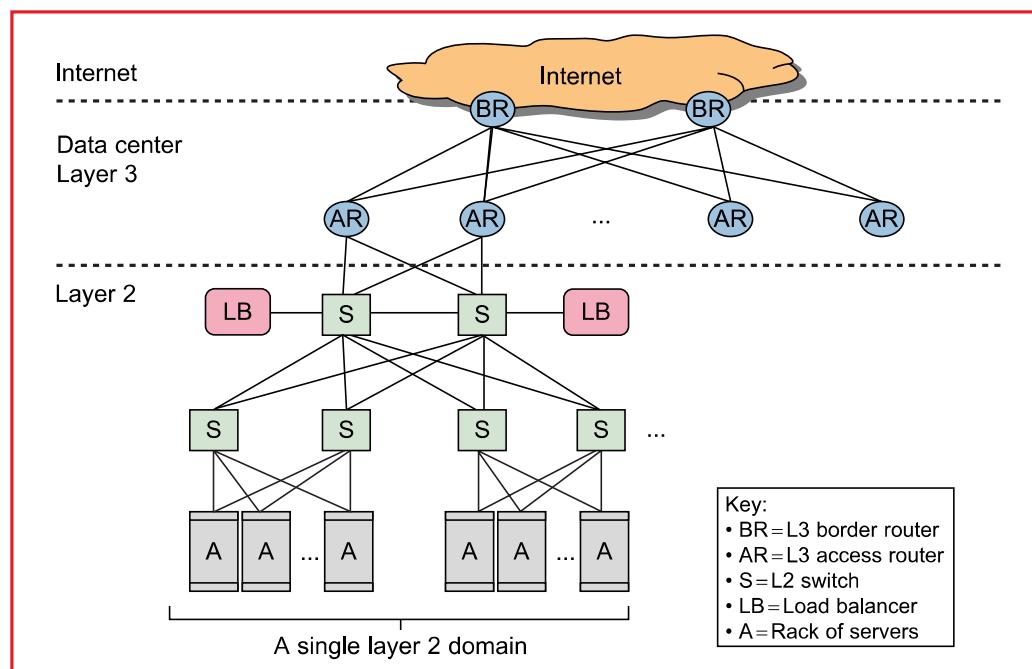


FIGURE 4.2

Standard data-center networking for the cloud to access the Internet.

(Courtesy of Dennis Gannon, 2010 [\[26\]](#))

An example of a private cloud is the one the U.S. National Aeronautics and Space Administration (NASA) is building to enable researchers to run climate models on remote systems it provides. This can save users the capital expense of HPC machines at local sites. Furthermore, NASA can build the complex weather models around its data centers, which is more cost-effective. Another good example is the cloud built by the European Council for Nuclear Research (CERN). This is a very big private cloud designed to distribute data, applications, and computing resources to thousands of scientists around the world.

These cloud models demand different levels of performance, data protection, and security enforcement. In this case, different SLAs may be applied to satisfy both providers and paid users. Cloud computing exploits many existing technologies. For example, grid computing is the backbone of cloud computing in that the grid has the same goals of resource sharing with better utilization of research facilities. Grids are more focused on delivering storage and computing resources while cloud computing aims to achieve economies of scale with abstracted services and resources.

4.1.1.6 Cloud Development Trends

Although most clouds built in 2010 are large public clouds, the authors believe private clouds will grow much faster than public clouds in the future. Private clouds are easier to secure and more trustworthy within a company or organization. Once private clouds become mature and better secured, they could be open or converted to public clouds. Therefore, the boundary between public and private clouds could be blurred in the future. Most likely, most future clouds will be hybrid in nature.

For example, an e-mail application can run in the service-access nodes and provide the user interface for outside users; the application can get the service from the internal cloud computing services (e.g., the e-mail storage service). There are also some service nodes designed to support the proper functioning of cloud computing clusters. These nodes are called runtime supporting service nodes. For example, there might be distributed locking services for supporting specific applications. Finally, it is possible that there will be some independent service nodes. Those nodes would provide independent services for other nodes in the cluster. For example, a news service need geographical information under service-access nodes.

With cost-effective performance as the key concept of clouds, we will consider the public cloud in this chapter, unless otherwise specified. Many executable application codes are much smaller than the web-scale data sets they process. Cloud computing avoids large data movement during execution. This will result in less traffic on the Internet and better network utilization. Clouds also alleviate the petascale I/O problem. Cloud performance and its *Quality of Service* (QoS) are yet to be proven in more real-life applications. We will model the performance of cloud computing in [Chapter 9](#), along with data protection, security measures, service availability, fault tolerance, and operating cost.

4.1.2 Cloud Ecosystem and Enabling Technologies

Cloud computing platforms differ from conventional computing platforms in many aspects. In this section, we will identify their differences in computing paradigms and cost models applied. The traditional computing model is specified below by the process on the left, which involves buying the hardware, acquiring the necessary system software, installing the system, testing the configuration,

and executing the application code and management of resources. What is even worse is that this cycle repeats itself in about every 18 months, meaning the machine we bought becomes obsolete every 18 months.

The cloud computing paradigm is shown on the right. This computing model follows a pay-as-you-go model. Therefore the cost is significantly reduced, because we simply rent computer resources without buying the computer in advance. All hardware and software resources are leased from the cloud provider without capital investment on the part of the users. Only the execution phase costs some money. The experts at IBM have estimated that an 80 percent to 95 percent saving results from cloud computing, compared with the conventional computing paradigm. This is very much desired, especially for small businesses, which requires limited computing power and thus avoid the purchase of expensive computers or servers repeatedly every few years.

| Classical Computing | Cloud Computing |
|--|--|
| (Repeat the following cycle every 18 months) | (Pay as you go per each service provided) |
| Buy and own | Subscribe |
| Hardware, system software, applications to meet peak needs | - - - |
| Install, configure, test, verify, evaluate, manage | Use (Save about 80-95% of the total cost) |
| - - - | - - - |
| Use | (Finally) |
| - - - | \$ - Pay for what you use |
| Pay \$\$\$\$\$ (High cost) | based on the QoS |

For example, IBM has estimated that the worldwide cloud service market may reach \$126 billion by 2012, including components, infrastructure services, and business services. Internet clouds work as service factories built around multiple data centers. To formalize the above cloud computing model, we characterize the cloud cost model, the cloud ecosystems, and enabling technologies. These topics help our readers understand the motivations behind cloud computing. The intention is to remove the barriers of cloud computing

4.1.2.1 Cloud Design Objectives

Despite the controversy surrounding the replacement of desktop or deskside computing by centralized computing and storage services at data centers or big IT companies, the cloud computing community has reached some consensus on what has to be done to make cloud computing universally acceptable. The following list highlights six design objectives for cloud computing:

- **Shifting computing from desktops to data centers** Computer processing, storage, and software delivery is shifted away from desktops and local servers and toward data centers over the Internet.
- **Service provisioning and cloud economics** Providers supply cloud services by signing SLAs with consumers and end users. The services must be efficient in terms of computing, storage, and power consumption. Pricing is based on a pay-as-you-go policy.
- **Scalability in performance** The cloud platforms and software and infrastructure services must be able to scale in performance as the number of users increases.

- **Data privacy protection** Can you trust data centers to handle your private data and records? This concern must be addressed to make clouds successful as trusted services.
- **High quality of cloud services** The QoS of cloud computing must be standardized to make clouds interoperable among multiple providers.
- **New standards and interfaces** This refers to solving the data lock-in problem associated with data centers or cloud providers. Universally accepted APIs and access protocols are needed to provide high portability and flexibility of virtualized applications.

4.1.2.2 Cost Model

In traditional IT computing, users must acquire their own computer and peripheral equipment as capital expenses. In addition, they have to face operational expenditures in operating and maintaining the computer systems, including personnel and service costs. Figure 4.3(a) shows the addition of variable operational costs on top of fixed capital investments in traditional IT. Note that the fixed cost is the main cost, and that it could be reduced slightly as the number of users increases. However, the operational costs may increase sharply with a larger number of users. Therefore, the total cost escalates quickly with massive numbers of users. On the other hand, cloud computing applies a pay-per-use business model, in which user jobs are outsourced to data centers. To use the cloud, one has no up-front cost in hardware acquisitions. Only variable costs are experienced by cloud users, as demonstrated in Figure 4.3(b).

Overall, cloud computing will reduce computing costs significantly for both small users and large enterprises. Computing economics does show a big gap between traditional IT users and cloud users. The savings in acquiring expensive computers up front releases a lot of burden for startup companies. The fact that cloud users only pay for operational expenses and do not have to invest in permanent equipment is especially attractive to massive numbers of small users. This is a major driving force for cloud computing to become appealing to most enterprises and heavy computer users. In fact, any IT users whose capital expenses are under more pressure than their operational expenses should consider sending their overflow work to utility computing or cloud service providers.

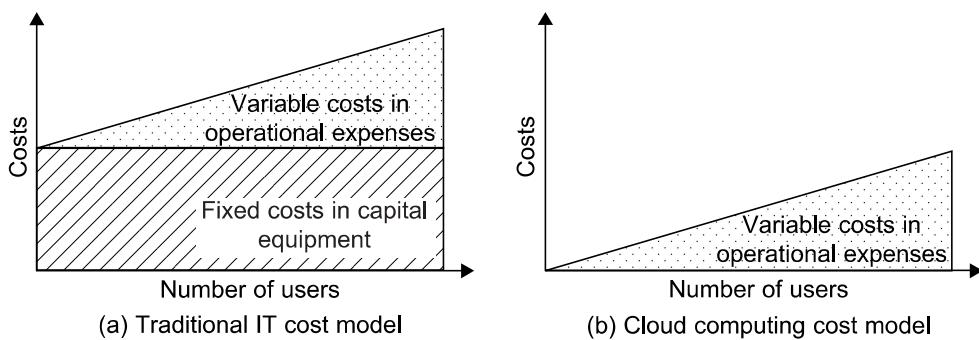


FIGURE 4.3

Computing economics between traditional IT users and cloud users.

4.1.2.3 Cloud Ecosystems

With the emergence of various Internet clouds, an ecosystem of providers, users, and technologies has appeared. This ecosystem has evolved around public clouds. Strong interest is growing in open source cloud computing tools that let organizations build their own IaaS clouds using their internal infrastructures. Private and hybrid clouds are not exclusive, since public clouds are involved in both cloud types. A private/hybrid cloud allows remote access to its resources over the Internet using remote web service interfaces such as that used in Amazon EC2.

An ecosystem was suggested by Sotomayor, et al. [39] (Figure 4.4) for building private clouds. They suggested four levels of ecosystem development in a private cloud. At the user end, consumers demand a flexible platform. At the cloud management level, the cloud manager provides virtualized resources over an IaaS platform. At the virtual infrastructure (VI) management level, the manager allocates VMs over multiple server clusters. Finally, at the VM management level, the VM managers handle VMs installed on individual host machines. An ecosystem of cloud tools attempts to span both cloud management and VI management. Integrating these two layers is complicated by the lack of open and standard interfaces between them.

An increasing number of startup companies are now basing their IT strategies on cloud resources, spending little or no capital to manage their own IT infrastructures. We desire a flexible and open architecture that enables organizations to build private/hybrid clouds. VI management is aimed at this goal. Example VI tools include oVirt (<https://fedorahosted.org/ovirt/>), vSphere/4

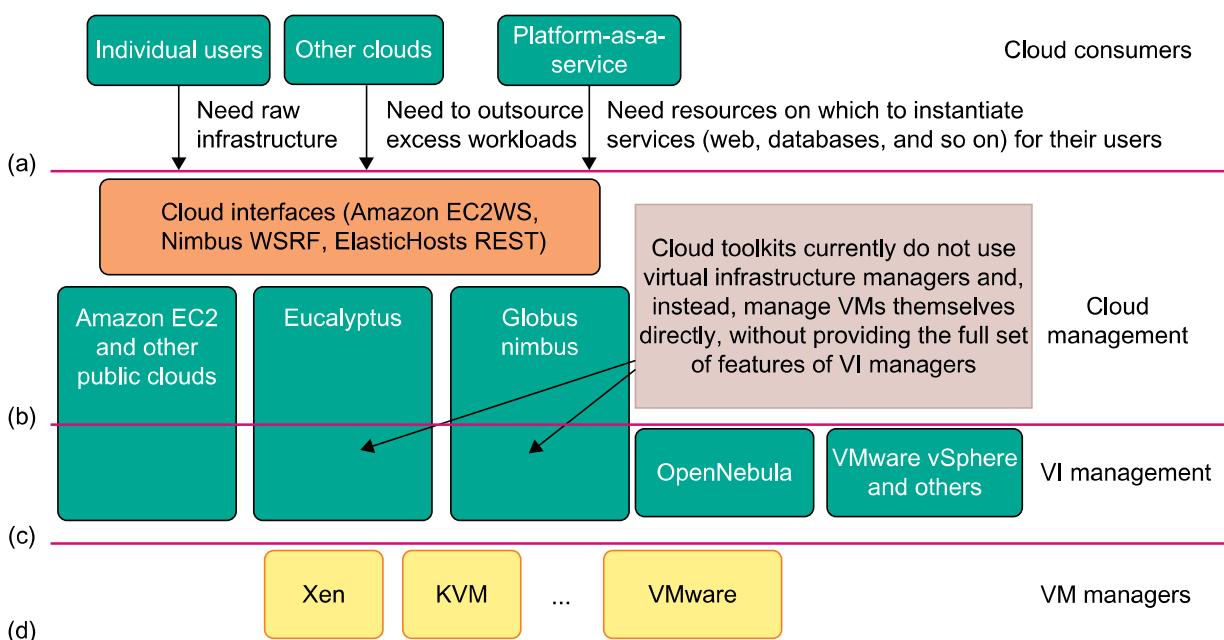


FIGURE 4.4

Cloud ecosystem for building private clouds: (a) Consumers demand a flexible platform; (b) Cloud manager provides virtualized resources over an IaaS platform; (c) VI manager allocates VMs; (d) VM managers handle VMs installed on servers.

(Courtesy of Sotomayor, et al. © IEEE [68])

(www.vmware.com/products/vsphere/) from VMWare, and VM Orchestrator (www.platform.com/Products/platform-vm-orchestrator) from Platfom Computing.

These tools support dynamic placement and VM management on a pool of physical resources, automatic load balancing, server consolidation, and dynamic infrastructure resizing and partitioning. In addition to public clouds such as Amazon EC2, Eucalyptus and Globus Nimbus are open source tools for virtualization of cloud infrastructure. To access these cloud management tools, one can use the Amazon EC2WS, Nimbus WSRF, and ElasticHost REST cloud interfaces. For VI management, OpenNebula and VMware vSphere can be used to manage all VM generation including Xen, KVM, and VMware tools.

4.1.2.4 Surge of Private Clouds

In general, private clouds leverage existing IT infrastructure and personnel within an enterprise or government organization. Both public and private clouds handle workloads dynamically. However, public clouds should be designed to handle workloads without communication dependency. Both types of clouds distribute data and VM resources. However, private clouds can balance workloads to exploit IT resources more efficiently within the same intranet. Private clouds can also provide pre-production testing and enforce data privacy and security policies more effectively. In a public cloud, the surge workload is often offloaded. The major advantage of public clouds lies in the avoidance of capital expenses by users in IT investments in hardware, software, and personnel.

Most companies start with virtualization of their computing machines to lower the operating costs. Companies such as Microsoft, Oracle, and SAP may want to establish policy-driven management of their computing resources, mainly to improve QoS to their employees and customers. By integrating virtualized data centers and company IT resources, they offer *IT as a service* to improve the agility of their company operations. This approach avoids replacement of a large number of servers every 18 months. As a result, these companies can upgrade their IT efficiency significantly.

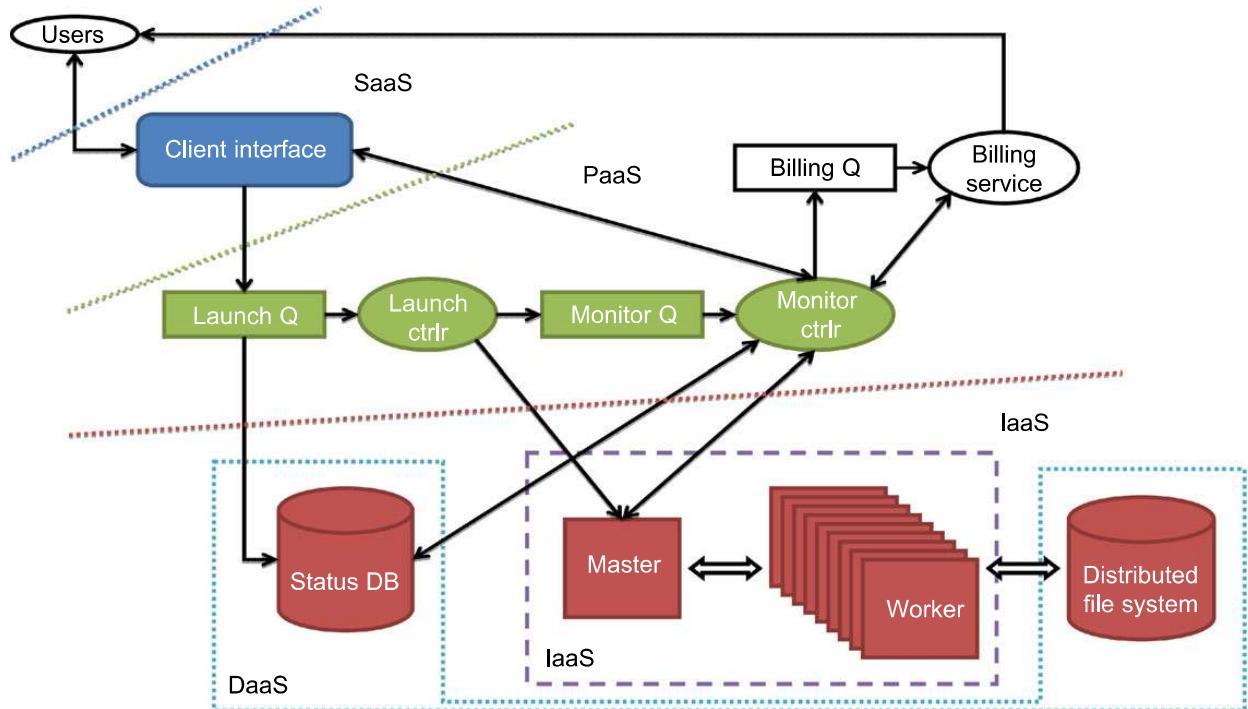
4.1.3 Infrastructure-as-a-Service (IaaS)

Cloud computing delivers infrastructure, platform, and software (application) as services, which are made available as subscription-based services in a pay-as-you-go model to consumers. The services provided over the cloud can be generally categorized into three different service models: namely IaaS, Platform as a Service (PaaS), and Software as a Service (SaaS). These form the three pillars on top of which cloud computing solutions are delivered to end users. All three models allow users to access services over the Internet, relying entirely on the infrastructures of cloud service providers.

These models are offered based on various SLAs between providers and users. In a broad sense, the SLA for cloud computing is addressed in terms of service availability, performance, and data protection and security. [Figure 4.5](#) illustrates three cloud models at different service levels of the cloud. SaaS is applied at the application end using special interfaces by users or clients. At the PaaS layer, the cloud platform must perform billing services and handle job queuing, launching, and monitoring services. At the bottom layer of the IaaS services, databases, compute instances, the file system, and storage must be provisioned to satisfy user demands.

4.1.3.1 Infrastructure as a Service

This model allows users to use virtualized IT resources for computing, storage, and networking. In short, the service is performed by rented cloud infrastructure. The user can deploy and run his

**FIGURE 4.5**

The IaaS, PaaS, and SaaS cloud service models at different service levels..

(Courtesy of J. Suh and S. Kang, USC)

applications over his chosen OS environment. The user does not manage or control the underlying cloud infrastructure, but has control over the OS, storage, deployed applications, and possibly select networking components. This IaaS model encompasses *storage as a service*, *compute instances as a service*, and *communication as a service*. The *Virtual Private Cloud (VPC)* in Example 4.1 shows how to provide Amazon EC2 clusters and S3 storage to multiple users. Many startup cloud providers have appeared in recent years. GoGrid, FlexiScale, and Aneka are good examples. Table 4.1 summarizes the IaaS offerings by five public cloud providers. Interested readers can visit the companies' web sites for updated information. More examples can be also found in two recent cloud books [10,18].

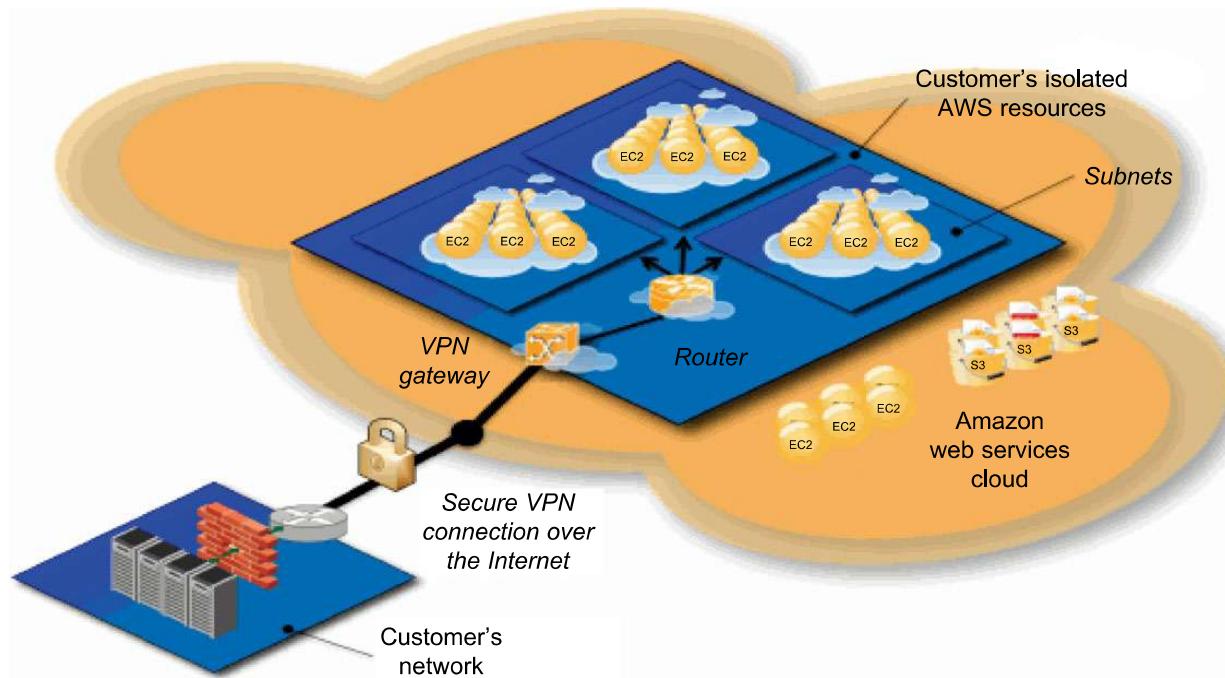
Example 4.1 Amazon VPC for Multiple Tenants

A user can use a private facility for basic computations. When he must meet a specific workload requirement, he can use the Amazon VPC to provide additional EC2 instances or more storage (S3) to handle urgent applications. Figure 4.6 shows VPC which is essentially a private cloud designed to address the privacy concerns of public clouds that hamper their application when sensitive data and software are involved.

Amazon EC2 provides the following services: resources from multiple data centers globally distributed, CL1, web services (SOAP and Query), web-based console user interfaces, access to VM instances via

Table 4.1 Public Cloud Offerings of IaaS [10,18]

| Cloud Name | VM Instance Capacity | API and Access Tools | Hypervisor, Guest OS |
|-----------------------------|---|-----------------------------------|--------------------------------------|
| Amazon EC2 | Each instance has 1–20 EC2 processors, 1.7–15 GB of memory, and 160–1.69 TB of storage. | CLI or web Service (WS) portal | Xen, Linux, Windows |
| GoGrid | Each instance has 1–6 CPUs, 0.5–8 GB of memory, and 30–480 GB of storage. | REST, Java, PHP, Python, Ruby | Xen, Linux, Windows |
| Rackspace Cloud | Each instance has a four-core CPU, 0.25–16 GB of memory, and 10–620 GB of storage. | REST, Python, PHP, Java, C#, .NET | Xen, Linux |
| FlexiScale in the UK | Each instance has 1–4 CPUs, 0.5–16 GB of memory, and 20–270 GB of storage. | web console | Xen, Linux, Windows |
| Joyent Cloud | Each instance has up to eight CPUs, 0.25–32 GB of memory, and 30–480 GB of storage. | No specific API, SSH, Virtual/MiN | OS-level virtualization, OpenSolaris |

**FIGURE 4.6**

Amazon VPC (virtual private cloud).

(Courtesy of VMWare, <http://aws.amazon.com/vpc/>)

SSH and Windows, 99.5 percent available agreements, per-hour pricing, Linux and Windows OSes, and automatic scaling and load balancing. We will illustrate the use of EC2 in more detail in [Chapter 6](#). VPC allows the user to isolate provisioned AWS processors, memory, and storage from interference by other users. Both *auto-scaling* and *elastic load balancing* services can support related demands. Auto-scaling enables users to automatically scale their VM instance capacity up or down. With auto-scaling, one can ensure that a sufficient number of Amazon EC2 instances are provisioned to meet desired performance. Or one can scale down the VM instance capacity to reduce costs, when the workload is reduced.

4.1.4 Platform-as-a-Service (PaaS) and Software-as-a-Service (SaaS)

In this section, we will introduce the PaaS and SaaS models for cloud computing. SaaS is often built on top of the PaaS, which is in turn built on top of the IaaS.

4.1.4.1 Platform as a Service (PaaS)

To be able to develop, deploy, and manage the execution of applications using provisioned resources demands a cloud platform with the proper software environment. Such a platform includes operating system and runtime library support. This has triggered the creation of the PaaS model to enable users to develop and deploy their user applications. [Table 4.2](#) highlights cloud platform services offered by five PaaS services. Further details of some of these PaaS offerings are provided in [Section 4.4](#) and in [Chapter 6](#). Some illustrative examples and case studies can be found in [\[10,18\]](#).

The platform cloud is an integrated computer system consisting of both hardware and software infrastructure. The user application can be developed on this virtualized cloud platform using some programming languages and software tools supported by the provider (e.g., Java, Python, .NET). The user does not manage the underlying cloud infrastructure. The cloud provider supports user application development and testing on a well-defined service platform. This PaaS model enables a collaborated

Table 4.2 Five Public Cloud Offerings of PaaS [\[10,18\]](#)

| Cloud Name | Languages and Developer Tools | Programming Models Supported by Provider | Target Applications and Storage Option |
|----------------------------|---|---|--|
| Google App Engine | Python, Java, and Eclipse-based IDE | MapReduce, web programming on demand | Web applications and BigTable storage |
| Salesforce.com's Force.com | Apex, Eclipse-based IDE, web-based Wizard | Workflow, Excel-like formula, Web programming on demand | Business applications such as CRM |
| Microsoft Azure | .NET, Azure tools for MS Visual Studio | Unrestricted model | Enterprise and web applications |
| Amazon Elastic MapReduce | Hive, Pig, Cascading, Java, Ruby, Perl, Python, PHP, R, C++ | MapReduce | Data processing and e-commerce |
| Aneka | .NET, stand-alone SDK | Threads, task, MapReduce | .NET enterprise applications, HPC |

software development platform for users from different parts of the world. This model also encourages third parties to provide software management, integration, and service monitoring solutions.

Example 4.2 Google App Engine for PaaS Applications

As web applications are running on Google's server clusters, they share the same capability with many other users. The applications have features such as automatic scaling and load balancing which are very convenient while building web applications. The distributed scheduler mechanism can also schedule tasks for triggering events at specified times and regular intervals. Figure 4.7 shows the operational model for GAE. To develop applications using GAE, a development environment must be provided.

Google provides a fully featured local development environment that simulates GAE on the developer's computer. All the functions and application logic can be implemented locally which is quite similar to traditional software development. The coding and debugging stages can be performed locally as

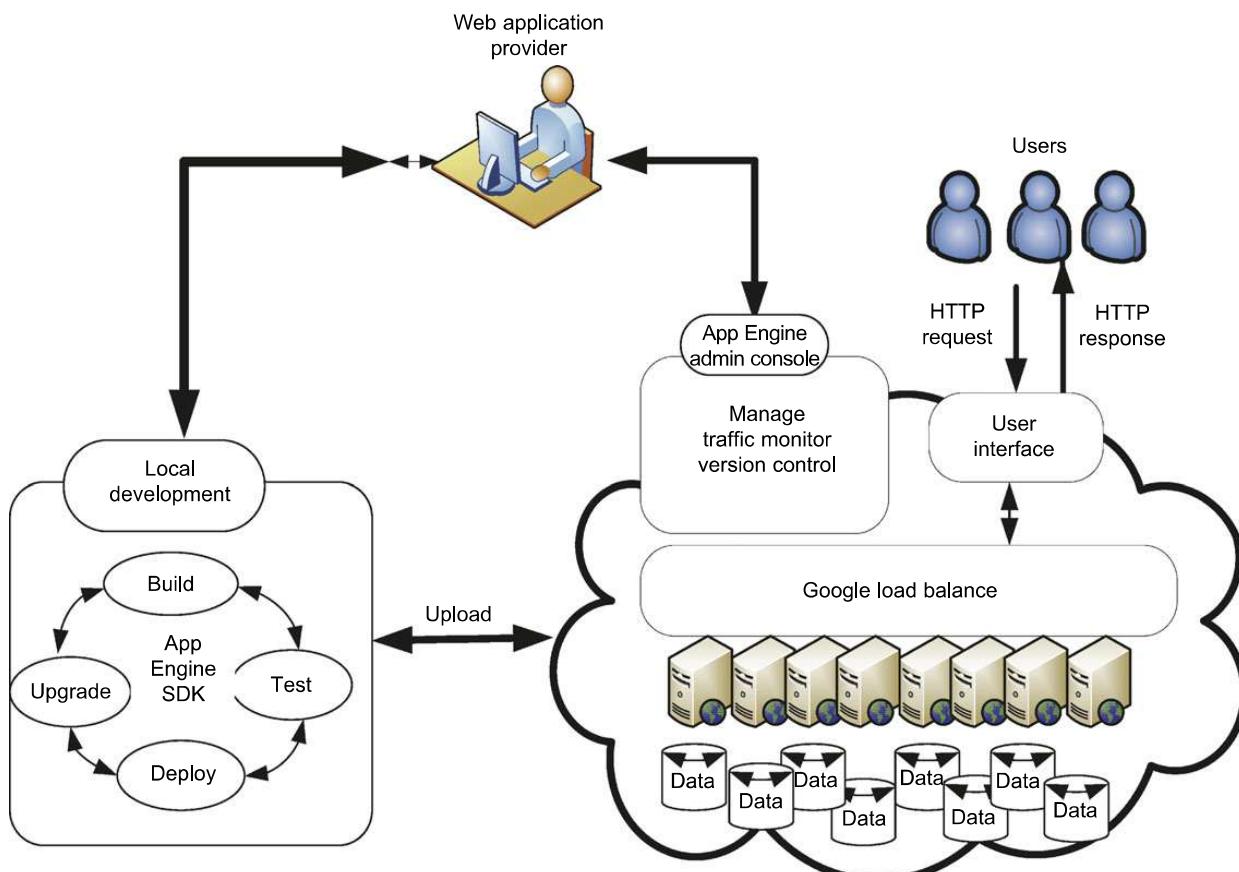


FIGURE 4.7

Google App Engine platform for PaaS operations.

(Courtesy of Yangting Wu, USC)

well. After these steps are finished, the SDK provided provides a tool for uploading the user's application to Google's infrastructure where the applications are actually deployed. Many additional third-party capabilities, including software management, integration, and service monitoring solutions, are also provided.

Here are some useful links when logging on to the GAE system:

- Google App Engine home page: <http://code.google.com/appengine/>
- Sign up for an account or use your Gmail account name: <https://appengine.google.com/>
- Download GAE SDK: <http://code.google.com/appengine/downloads.html>
- Python Getting Started Guide: <http://code.google.com/appengine/docs/python/gettingstarted/>
- Java Getting Started Guide: <http://code.google.com/appengine/docs/java/gettingstarted/>
- Quota page for free service: <http://code.google.com/appengine/docs/quotas.html#Resources>
- Billing page if you go over the quota: http://code.google.com/appengine/docs/billing.html#Billable_Quota_Unit_Cost

4.1.4.2 Software as a Service (SaaS)

This refers to browser-initiated application software over thousands of cloud customers. Services and tools offered by PaaS are utilized in construction of applications and management of their deployment on resources offered by IaaS providers. The SaaS model provides software applications as a service. As a result, on the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are kept rather low, compared with conventional hosting of user applications. Customer data is stored in the cloud that is either vendor proprietary or publicly hosted to support PaaS and IaaS.

The best examples of SaaS services include Google Gmail and docs, Microsoft SharePoint, and the CRM software from Salesforce.com. They are all very successful in promoting their own business or are used by thousands of small businesses in their day-to-day operations. Providers such as Google and Microsoft offer integrated IaaS and PaaS services, whereas others such as Amazon and GoGrid offer pure IaaS services and expect third-party PaaS providers such as Manjrasoft to offer application development and deployment services on top of their infrastructure services. To identify important cloud applications in enterprises, the success stories of three real-life cloud applications are presented in [Example 4.3](#) for HTC, news media, and business transactions. The benefits of using cloud services are evident in these SaaS applications.

Example 4.3 Three Success Stories on SaaS Applications

1. To discover new drugs through DNA sequence analysis, Eli Lily Company has used Amazon's AWS platform with provisioned server and storage clusters to conduct high-performance biological sequence analysis without using an expensive supercomputer. The benefit of this IaaS application is reduced drug deployment time with much lower costs.
2. The *New York Times* has applied Amazon's EC2 and S3 services to retrieve useful pictorial information quickly from millions of archival articles and newspapers. The *New York Times* has significantly reduced the time and cost in getting the job done.

3. Pitney Bowes, an e-commerce company, offers clients the opportunity to perform B2B transactions using the Microsoft Azure platform, along with .NET and SQL services. These offerings have significantly increased the company's client base.

4.1.4.3 Mashup of Cloud Services

At the time of this writing, public clouds are in use by a growing number of users. Due to the lack of trust in leaking sensitive data in the business world, more and more enterprises, organizations, and communities are developing private clouds that demand deep customization. An enterprise cloud is used by multiple users within an organization. Each user may build some strategic applications on the cloud, and demands customized partitioning of the data, logic, and database in the metadata representation. More private clouds may appear in the future.

Based on a 2010 Google search survey, interest in grid computing is declining rapidly. *Cloud mashups* have resulted from the need to use multiple clouds simultaneously or in sequence. For example, an industrial supply chain may involve the use of different cloud resources or services at different stages of the chain. Some public repository provides thousands of service APIs and mash-ups for web commerce services. Popular APIs are provided by Google Maps, Twitter, YouTube, Amazon eCommerce, Salesforce.com, etc.

4.2 DATA-CENTER DESIGN AND INTERCONNECTION NETWORKS

A data center is often built with a large number of servers through a huge interconnection network. In this section, we will study the design of large-scale data centers and small modular data centers that can be housed in a 40-ft truck container. Then we will take a look at interconnection of modular data centers and their management issues and solutions.

4.2.1 Warehouse-Scale Data-Center Design

Dennis Gannon claims: “The cloud is built on massive datacenters” [26]. Figure 4.8 shows a data center that is as large as a shopping mall (11 times the size of a football field) under one roof. Such a data center can house 400,000 to 1 million servers. The data centers are built economics of scale—meaning lower unit cost for larger data centers. A small data center could have 1,000 servers. The larger the data center, the lower the operational cost. The approximate monthly cost to operate a huge 400-server data center is estimated by network cost \$13/Mbps; storage cost \$0.4/GB; and administration costs. These unit costs are greater than those of a 1,000-server data center. The network cost to operate a small data center is about seven times greater and the storage cost is 5.7 times greater. Microsoft has about 100 data centers, large or small, which are distributed around the globe.

4.2.1.1 Data-Center Construction Requirements

Most data centers are built with commercially available components. An off-the-shelf server consists of a number of processor sockets, each with a multicore CPU and its internal cache hierarchy, local shared and coherent DRAM, and a number of directly attached disk drives. The DRAM and disk resources within the rack are accessible through first-level rack switches and all resources in all

**FIGURE 4.8**

A huge data center that is 11 times the size of a football field, housing 400,000 to 1 million servers.

(Courtesy of Dennis Gannon [26])

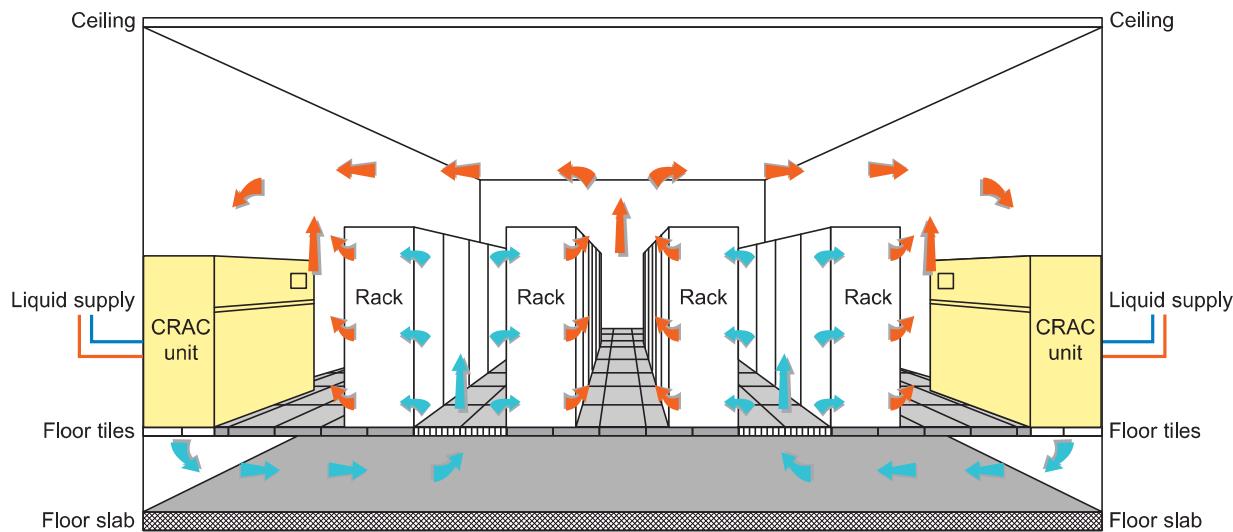
racks are accessible via a cluster-level switch. Consider a data center built with 2,000 servers, each with 8 GB of DRAM and four 1 TB disk drives. Each group of 40 servers is connected through a 1 Gbps link to a rack-level switch that has an additional eight 1 Gbps ports used for connecting the rack to the cluster-level switch.

It was estimated [9] that the bandwidth available from local disks is 200 MB/s, whereas the bandwidth from off-rack disks is 25 MB/s via shared rack uplinks. The total disk storage in the cluster is almost 10 million times larger than local DRAM. A large application must deal with large discrepancies in latency, bandwidth, and capacity. In a very large-scale data center, components are relatively cheaper. The components used in data centers are very different from those in building supercomputer systems.

With a scale of thousands of servers, concurrent failure, either hardware failure or software failure, of 1 percent of nodes is common. Many failures can happen in hardware; for example, CPU failure, disk I/O failure, and network failure. It is even quite possible that the whole data center does not work in the case of a power crash. Also, some failures are brought on by software. The service and data should not be lost in a failure situation. Reliability can be achieved by redundant hardware. The software must keep multiple copies of data in different locations and keep the data accessible while facing hardware or software errors.

4.2.1.2 Cooling System of a Data-Center Room

Figure 4.9 shows the layout and cooling facility of a warehouse in a data center. The data-center room has raised floors for hiding cables, power lines, and cooling supplies. The cooling system is somewhat simpler than the power system. The raised floor has a steel grid resting on stanchions

**FIGURE 4.9**

The cooling system in a raised-floor data center with hot-cold air circulation supporting water heat exchange facilities.

(Courtesy of DLB Associates, D. Dyer [22])

about 2–4 ft above the concrete floor. The under-floor area is often used to route power cables to racks, but its primary use is to distribute cool air to the server rack. The CRAC (*computer room air conditioning*) unit pressurizes the raised floor plenum by blowing cold air into the plenum.

The cold air escapes from the plenum through perforated tiles that are placed in front of server racks. Racks are arranged in long aisles that alternate between cold aisles and hot aisles to avoid mixing hot and cold air. The hot air produced by the servers circulates back to the intakes of the CRAC units that cool it and then exhaust the cool air into the raised floor plenum again. Typically, the incoming coolant is at 12–14°C and the warm coolant returns to a chiller. Newer data centers often insert a cooling tower to pre-cool the condenser water loop fluid. Water-based free cooling uses cooling towers to dissipate heat. The cooling towers use a separate cooling loop in which water absorbs the coolant's heat in a heat exchanger.

4.2.2 Data-Center Interconnection Networks

A critical core design of a data center is the interconnection network among all servers in the data-center cluster. This network design must meet five special requirements: low latency, high bandwidth, low cost, *message-passing interface* (MPI) communication support, and fault tolerance. The design of an inter-server network must satisfy both point-to-point and collective communication patterns among all server nodes. Specific design considerations are given in the following sections.

4.2.2.1 Application Traffic Support

The network topology should support all MPI communication patterns. Both point-to-point and collective MPI communications must be supported. The network should have high bisection bandwidth

to meet this requirement. For example, one-to-many communications are used for supporting distributed file access. One can use one or a few servers as metadata master servers which need to communicate with slave server nodes in the cluster. To support the MapReduce programming paradigm, the network must be designed to perform the map and reduce functions (to be treated in [Chapter 7](#)) at a high speed. In other words, the underlying network structure should support various network traffic patterns demanded by user applications.

4.2.2.2 Network Expandability

The interconnection network should be expandable. With thousands or even hundreds of thousands of server nodes, the cluster network interconnection should be allowed to expand once more servers are added to the data center. The network topology should be restructured while facing such expected growth in the future. Also, the network should be designed to support load balancing and data movement among the servers. None of the links should become a bottleneck that slows down application performance. The topology of the interconnection should avoid such bottlenecks.

The fat-tree and crossbar networks studied in [Chapter 2](#) could be implemented with low-cost Ethernet switches. However, the design could be very challenging when the number of servers increases sharply. The most critical issue regarding expandability is support of modular network growth for building data-center containers, as discussed in [Section 4.2.3](#). One single data-center container contains hundreds of servers and is considered to be the building block of large-scale data centers. The network interconnection among many containers will be explained in [Section 4.2.4](#). Cluster networks need to be designed for data-center containers. Cable connections are then needed among multiple data-center containers.

Data centers are not built by piling up servers in multiple racks today. Instead, data-center owners buy server containers while each container contains several hundred or even thousands of server nodes. The owners can just plug in the power supply, outside connection link, and cooling water, and the whole system will just work. This is quite efficient and reduces the cost of purchasing and maintaining servers. One approach is to establish the connection backbone first and then extend the backbone links to reach the end servers. One can also connect multiple containers through external switching and cabling.

4.2.2.3 Fault Tolerance and Graceful Degradation

The interconnection network should provide some mechanism to tolerate link or switch failures. In addition, multiple paths should be established between any two server nodes in a data center. Fault tolerance of servers is achieved by replicating data and computing among redundant servers. Similar redundancy technology should apply to the network structure. Both software and hardware network redundancy apply to cope with potential failures. On the software side, the software layer should be aware of network failures. Packet forwarding should avoid using broken links. The network support software drivers should handle this transparently without affecting cloud operations.

In case of failures, the network structure should degrade gracefully amid limited node failures. Hot-swappable components are desired. There should be no critical paths or critical points which may become a single point of failure that pulls down the entire system. Most design innovations are in the topology structure of the network. The network structure is often divided into two layers. The lower layer is close to the end servers, and the upper layer establishes the backbone connections among the server groups or sub-clusters. This hierarchical interconnection approach appeals to building data centers with modular containers.

4.2.2.4 Switch-centric Data-Center Design

At the time of this writing, there are two approaches to building data-center-scale networks: One is switch-centric and the other is server-centric. In a switch-centric network, the switches are used to connect the server nodes. The switch-centric design does not affect the server side. No modifications to the servers are needed. The server-centric design does modify the operating system running on the servers. Special drivers are designed for relaying the traffic. Switches still have to be organized to achieve the connections.

Example 4.4 A Fat-Tree Interconnection Network for Data Centers

Figure 4.10 shows a fat-tree switch network design for data-center construction. The fat-tree topology is applied to interconnect the server nodes. The topology is organized into two layers. Server nodes are in the bottom layer, and *edge switches* are used to connect the nodes in the bottom layer. The upper layer aggregates the lower-layer edge switches. A group of aggregation switches, edge switches, and their leaf nodes form a *pod*. *Core switches* provide paths among different pods. The fat-tree structure provides multiple paths between any two server nodes. This provides fault-tolerant capability with an alternate path in case of some isolated link failures.

The failure of an aggregation switch and core switch will not affect the connectivity of the whole network. The failure of any edge switch can only affect a small number of end server nodes. The extra switches in a pod provide higher bandwidth to support cloud applications in massive data movement. The building blocks used are the low-cost Ethernet switches. This reduces the cost quite a bit. The routing table provides extra routing paths in case of failure. The routing algorithms are built inside the switches. The end server nodes in the data center are not affected during a switch failure, as long as the alternate routing path does not fail at the same time.

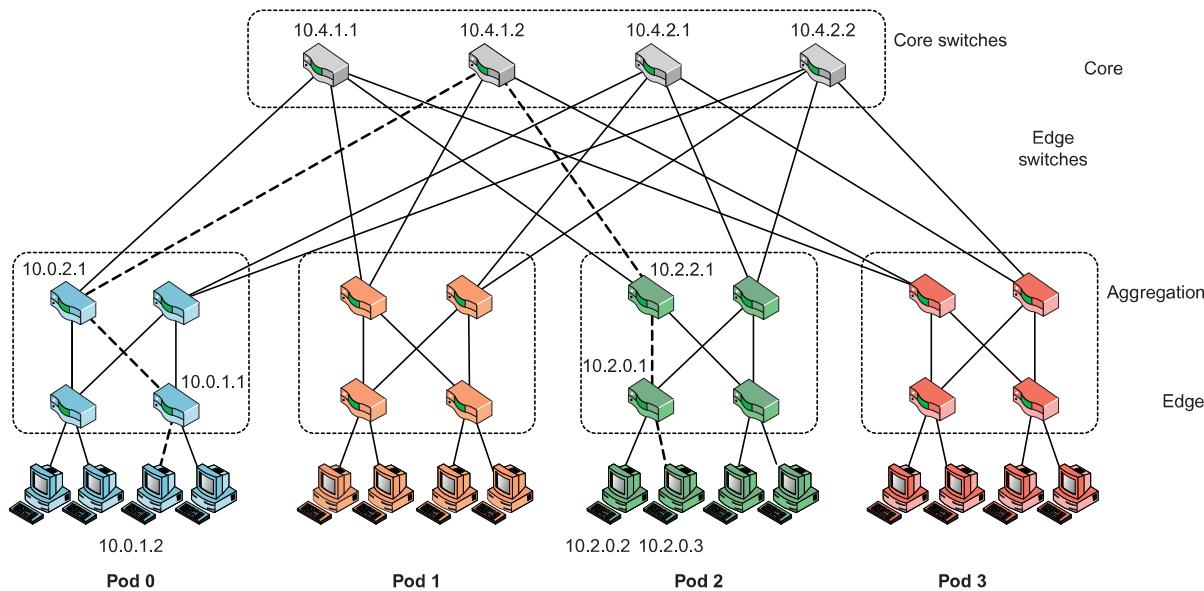


FIGURE 4.10

A fat-tree interconnection topology for scalable data-center construction.

(Courtesy of M. Al-Fares, et al. [2])

4.2.3 Modular Data Center in Shipping Containers

A modern data center is structured as a shipyard of server clusters housed in truck-towed containers. Figure 4.11 shows the housing of multiple sever racks in a truck-towed container in the SGI ICE Cube modular data center. Inside the container, hundreds of blade servers are housed in racks surrounding the container walls. An array of fans forces the heated air generated by the server racks to go through a heat exchanger, which cools the air for the next rack (detail in callout) on a continuous loop. The SGI ICE Cube container can house 46,080 processing cores or 30 PB of storage per container.

Large-scale data center built with modular containers appear as a big shipping yard of container trucks. This container-based data center was motivated by demand for lower power consumption, higher computer density, and mobility to relocate data centers to better locations with lower electricity costs, better cooling water supplies, and cheaper housing for maintenance engineers. Sophisticated cooling technology enables up to 80% reduction in cooling costs compared with traditional warehouse data centers. Both chilled air circulation and cold water are flowing through the heat exchange pipes to keep the server racks cool and easy to repair.

Data centers usually are built at a site where leases and utilities for electricity are cheaper, and cooling is more efficient. Both warehouse-scale and modular data centers in containers are needed. In fact, the modular truck containers can be used to put together a large-scale data center like a container shipping yard. In addition to location selection and power savings in data-center operations, one must consider data integrity, server monitoring, and security management in data centers. These problems are easier to handle if the data center is centralized in a single large building.

4.2.3.1 Container Data-Center Construction

The data-center module is housed in a truck-towable container. The modular container design includes the network, computer, storage, and cooling gear. One needs to increase cooling efficiency by varying the water and airflow with better airflow management. Another concern is to meet seasonal load requirements. The construction of a container-based data center may start with one system (server), then move to a rack system design, and finally to a container system. This staged development may take different amounts of time and demand increasing costs. Building a rack of 40 servers may



FIGURE 4.11

A modular data center built in a truck-towed ICE Cube container, that can be cooled by chilled air circulation with cold-water heat exchanges.

(Courtesy of SGI, Inc., <http://www.sgi.com/icecube>)

take half a day. Extending this to a whole container system with multiple racks for 1,000 servers requires the layout of the floor space with power, networking, cooling, and complete testing.

The container must be designed to be weatherproof and easy to transport. Modular data-center construction and testing may take a few days to complete if all components are available and power and water supplies are handy. The modular data-center approach supports many cloud service applications. For example, the health care industry will benefit by installing a data center at all clinic sites. However, how to exchange information with the central database and maintain periodic consistency becomes a rather challenging design issue in a hierarchically structured data center. The security of collocation cloud services may involve multiple data centers.

4.2.4 Interconnection of Modular Data Centers

Container-based data-center modules are meant for construction of even larger data centers using a farm of container modules. Some proposed designs of container modules are presented in this section. Their interconnections are shown for building scalable data centers. The following example is a server-centric design of the data-center module.

Example 4.5 A Server-Centric Network for a Modular Data Center

Guo, et al. [30] have developed a server-centric BCube network (Figure 4.12) for interconnecting modular data centers. The servers are represented by circles, and switches by rectangles. The BCube provides a layered structure. The bottom layer contains all the server nodes and they form Level 0. Level 1 switches form the top layer of BCube₀. BCube is a recursively constructed structure. The BCube₀ consists of n servers connecting to an n -port switch. The BCube_k ($k \geq 1$) is structured from n BCube_{k-1} with n^k n -port switches. The example of BCube₁ is illustrated in Figure 4.12, where the connection rule is that the i -th server in the j -th BCube₀ connects to the j -th port of the i -th Level 1 switch. The servers in the BCube have multiple ports attached. This allows extra devices to be used in the server.

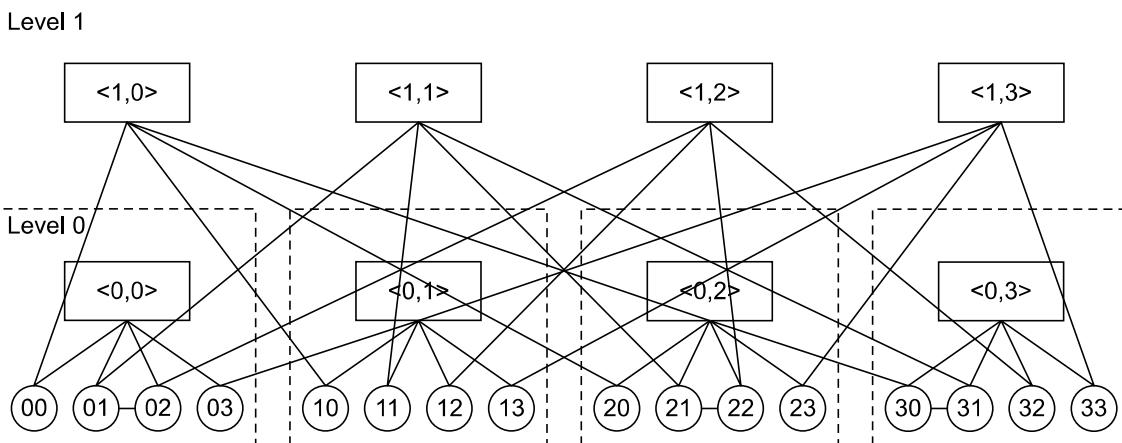


FIGURE 4.12

BCube, a high-performance, server-centric network for building modular data centers.

(Courtesy of C. Guo, et al. [30])

The BCube provides multiple paths between any two nodes. Multiple paths provide extra bandwidth to support communication patterns in different cloud applications. The BCube provides a kernel module in the server OS to perform routing operations. The kernel module supports packet forwarding while the incoming packets are not destined to the current node. Such modification of the kernel will not influence the upper layer applications. Thus, the cloud application can still run on top of the BCube network structure without any modification.

4.2.4.1 Inter-Module Connection Networks

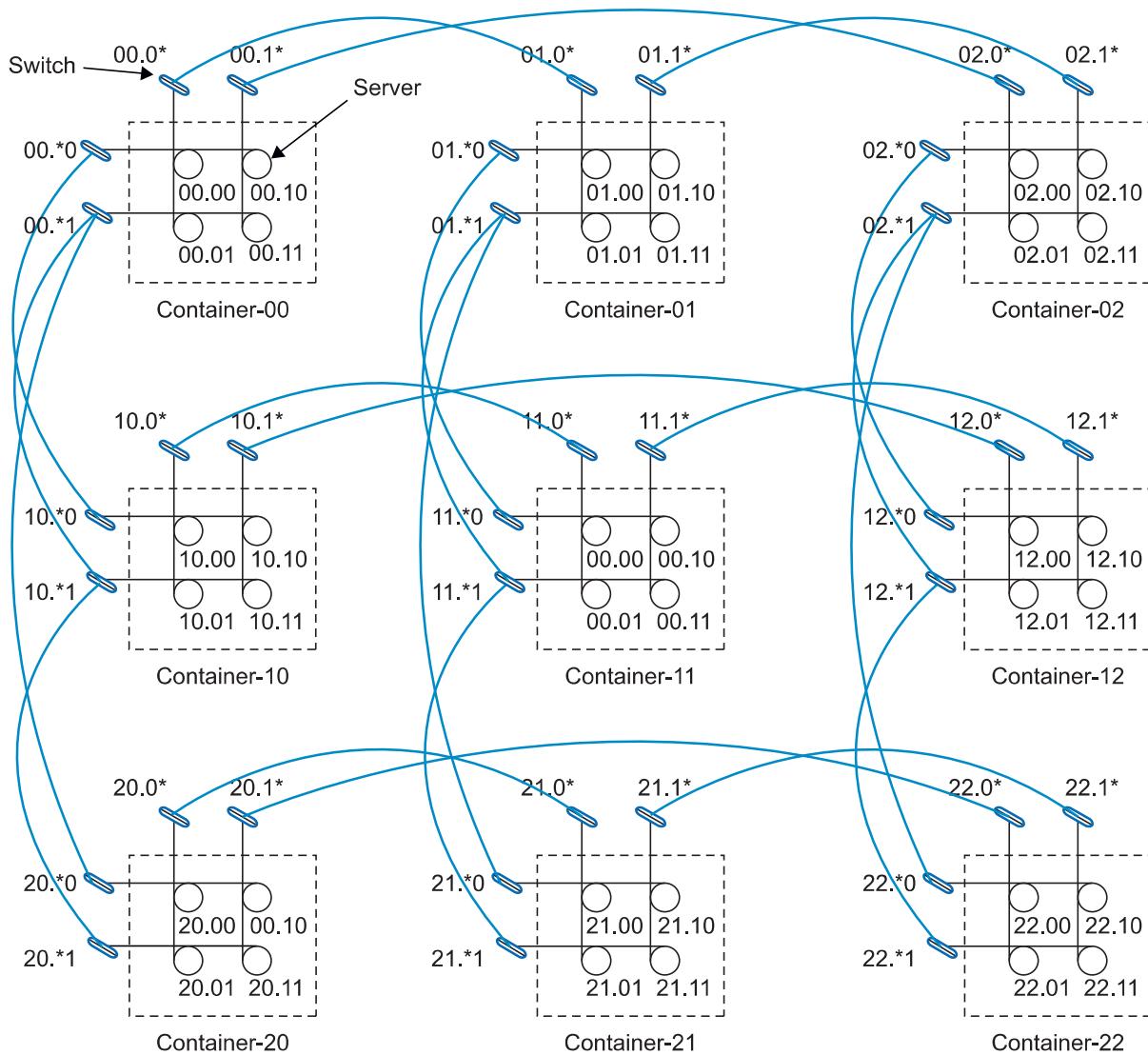
The BCube is commonly used inside a server container. The containers are considered the building blocks for data centers. Thus, despite the design of the inner container network, one needs another level of networking among multiple containers. In [Figure 4.13](#), Wu, et al. [82] have proposed a network topology for intercontainer connection using the aforementioned BCube network as building blocks. The proposed network was named MDCube (for *Modularized Datacenter Cube*). This network connects multiple BCube containers by using high-speed switches in the BCube. Similarly, the MDCube is constructed by shuffling networks with multiple containers. [Figure 4.13](#) shows how a 2D MDCube is constructed from nine BCube₁ containers.

The architecture builds a virtual hypercube at the container level, in addition to the cube structure inside the container (BCube). With the server container built with the BCube network, the MDCube is used to build a large-scale data center for supporting cloud application communication patterns. Readers are referred to the article at [45] for detailed implementation and simulation results of this interconnection network over multiple modular data centers built in containers. In fact, there are many other ways to use MDCube to build the network. Essentially, this network architecture builds a virtual hypercube at the container level, in addition to the cube structure inside the container (BCube). With the server container built with the BCube network, the MDCube is used to build a large-scale data center for supporting cloud application communication patterns [82].

4.2.5 Data-Center Management Issues

Here are basic requirements for managing the resources of a data center. These suggestions have resulted from the design and operational experiences of many data centers in the IT and service industries.

- **Making common users happy** The data center should be designed to provide quality service to the majority of users for at least 30 years.
- **Controlled information flow** Information flow should be streamlined. Sustained services and high availability (HA) are the primary goals.
- **Multiuser manageability** The system must be managed to support all functions of a data center, including traffic flow, database updating, and server maintenance.
- **Scalability to prepare for database growth** The system should allow growth as workload increases. The storage, processing, I/O, power, and cooling subsystems should be scalable.
- **Reliability in virtualized infrastructure** Failover, fault tolerance, and VM live migration should be integrated to enable recovery of critical applications from failures or disasters.

**FIGURE 4.13**

A 2D MDcube constructed from nine BCube containers.

(Courtesy of Wu, et al. [82])

- **Low cost to both users and providers** The cost to users and providers of the cloud system built over the data centers should be reduced, including all operational costs.
- **Security enforcement and data protection** Data privacy and security defense mechanisms must be deployed to protect the data center against network attacks and system interrupts and to maintain data integrity from user abuses or network attacks.
- **Green information technology** Saving power consumption and upgrading energy efficiency are in high demand when designing and operating current and future data centers.

4.2.5.1 Marketplaces in Cloud Computing Services

Container-based data-center implementation can be done more efficiently with factory racking, stacking, and packing. One should avoid layers of packaging at the customer site. However, the data centers are still custom-crafted rather than prefab units. The modular approach is more space-efficient with power densities in excess of 1250 W/sq ft. Rooftop or parking lot installation is acceptable. One should leave sufficient redundancy to allow upgrades over time.

4.3 ARCHITECTURAL DESIGN OF COMPUTE AND STORAGE CLOUDS

This section presents basic cloud design principles. We start with basic cloud architecture to process massive amounts of data with a high degree of parallelism. Then we study virtualization support, resource provisioning, infrastructure management, and performance modeling.

4.3.1 A Generic Cloud Architecture Design

An Internet cloud is envisioned as a public cluster of servers provisioned on demand to perform collective web services or distributed applications using data-center resources. In this section, we will discuss cloud design objectives and then present a basic cloud architecture design.

4.3.1.1 Cloud Platform Design Goals

Scalability, virtualization, efficiency, and reliability are four major design goals of a cloud computing platform. Clouds support Web 2.0 applications. Cloud management receives the user request, finds the correct resources, and then calls the provisioning services which invoke the resources in the cloud. The cloud management software needs to support both physical and virtual machines. Security in shared resources and shared access of data centers also pose another design challenge.

The platform needs to establish a very large-scale HPC infrastructure. The hardware and software systems are combined to make it easy and efficient to operate. System scalability can benefit from cluster architecture. If one service takes a lot of processing power, storage capacity, or network traffic, it is simple to add more servers and bandwidth. System reliability can benefit from this architecture. Data can be put into multiple locations. For example, user e-mail can be put in three disks which expand to different geographically separate data centers. In such a situation, even if one of the data centers crashes, the user data is still accessible. The scale of the cloud architecture can be easily expanded by adding more servers and enlarging the network connectivity accordingly.

4.3.1.2 Enabling Technologies for Clouds

The key driving forces behind cloud computing are the ubiquity of broadband and wireless networking, falling storage costs, and progressive improvements in Internet computing software. Cloud users are able to demand more capacity at peak demand, reduce costs, experiment with new services, and remove unneeded capacity, whereas service providers can increase system utilization via multiplexing, virtualization, and dynamic resource provisioning. Clouds are enabled by the progress in hardware, software, and networking technologies summarized in [Table 4.3](#).

Table 4.3 Cloud-Enabling Technologies in Hardware, Software, and Networking

| Technology | Requirements and Benefits |
|--------------------------------|---|
| Fast platform deployment | Fast, efficient, and flexible deployment of cloud resources to provide dynamic computing environment to users |
| Virtual clusters on demand | Virtualized cluster of VMs provisioned to satisfy user demand and virtual cluster reconfigured as workload changes |
| Multitenant techniques | SaaS for distributing software to a large number of users for their simultaneous use and resource sharing if so desired |
| Massive data processing | Internet search and web services which often require massive data processing, especially to support personalized services |
| Web-scale communication | Support for e-commerce, distance education, telemedicine, social networking, digital government, and digital entertainment applications |
| Distributed storage | Large-scale storage of personal records and public archive information which demands distributed storage over the clouds |
| Licensing and billing services | License management and billing services which greatly benefit all types of cloud services in utility computing |

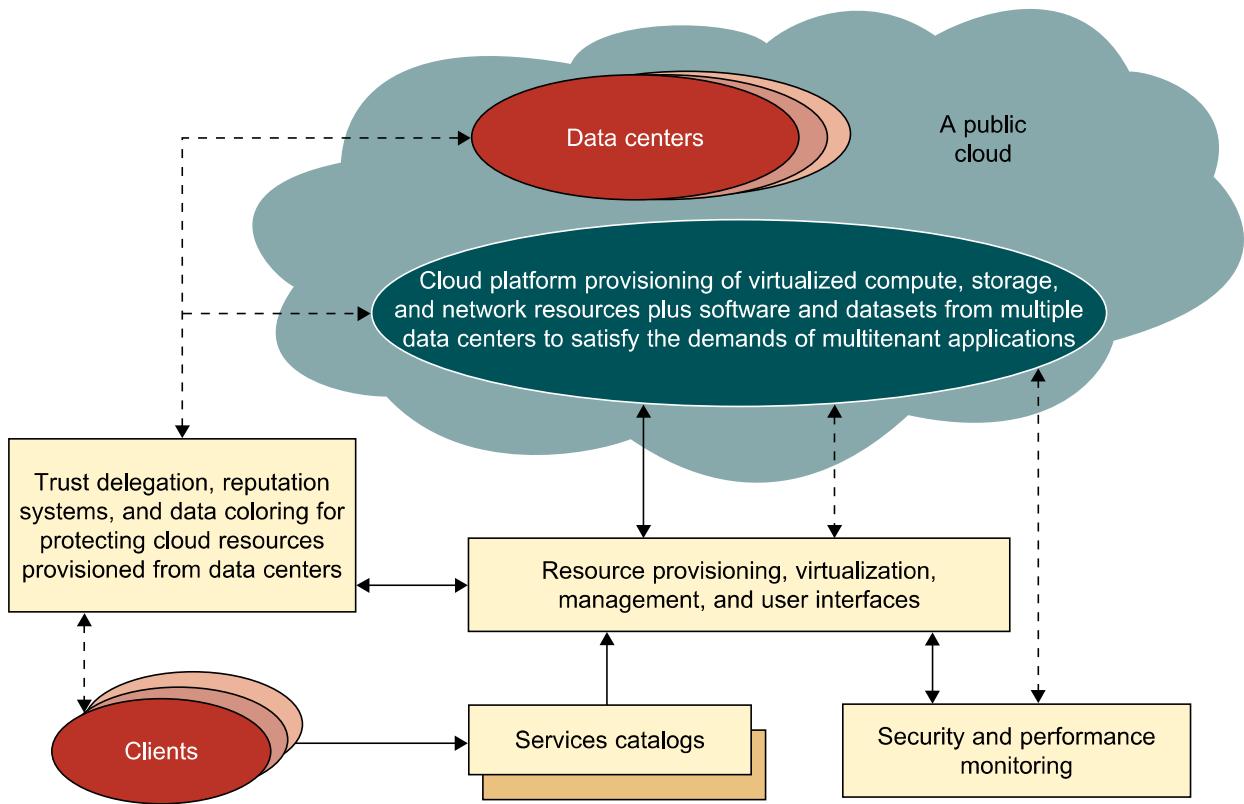
These technologies play instrumental roles in making cloud computing a reality. Most of these technologies are mature today to meet increasing demand. In the hardware area, the rapid progress in multicore CPUs, memory chips, and disk arrays has made it possible to build faster data centers with huge amounts of storage space. Resource virtualization enables rapid cloud deployment and disaster recovery. *Service-oriented architecture* (SOA) also plays a vital role.

Progress in providing SaaS, Web 2.0 standards, and Internet performance have all contributed to the emergence of cloud services. Today's clouds are designed to serve a large number of tenants over massive volumes of data. The availability of large-scale, distributed storage systems is the foundation of today's data centers. Of course, cloud computing is greatly benefitted by the progress made in license management and automatic billing techniques in recent years.

4.3.1.3 A Generic Cloud Architecture

Figure 4.14 shows a security-aware cloud architecture. The Internet cloud is envisioned as a massive cluster of servers. These servers are provisioned on demand to perform collective web services or distributed applications using data-center resources. The cloud platform is formed dynamically by provisioning or deprovisioning servers, software, and database resources. Servers in the cloud can be physical machines or VMs. User interfaces are applied to request services. The provisioning tool carves out the cloud system to deliver the requested service.

In addition to building the server cluster, the cloud platform demands distributed storage and accompanying services. The cloud computing resources are built into the data centers, which are typically owned and operated by a third-party provider. Consumers do not need to know the underlying technologies. In a cloud, software becomes a service. The cloud demands a high degree of trust of massive amounts of data retrieved from large data centers. We need to build a framework to process large-scale data stored in the storage system. This demands a distributed file system over the database system. Other cloud resources are added into a cloud platform, including storage area networks (SANs), database systems, firewalls, and security devices. Web service providers offer

**FIGURE 4.14**

A security-aware cloud platform built with a virtual cluster of VMs, storage, and networking resources over the data-center servers operated by providers.

(Courtesy of K. Hwang and D. Li, 2010 [36])

special APIs that enable developers to exploit Internet clouds. Monitoring and metering units are used to track the usage and performance of provisioned resources.

The software infrastructure of a cloud platform must handle all resource management and do most of the maintenance automatically. Software must detect the status of each node server joining and leaving, and perform relevant tasks accordingly. Cloud computing providers, such as Google and Microsoft, have built a large number of data centers all over the world. Each data center may have thousands of servers. The location of the data center is chosen to reduce power and cooling costs. Thus, the data centers are often built around hydroelectric power. The cloud physical platform builder is more concerned about the performance/price ratio and reliability issues than sheer speed performance.

In general, private clouds are easier to manage, and public clouds are easier to access. The trends in cloud development are that more and more clouds will be hybrid. This is because many cloud applications must go beyond the boundary of an intranet. One must learn how to create a private cloud and how to interact with public clouds in the open Internet. Security becomes a critical issue in safeguarding the operation of all cloud types. We will study cloud security and privacy issues at the end of this chapter.

4.3.2 Layered Cloud Architectural Development

The architecture of a cloud is developed at three layers: infrastructure, platform, and application, as demonstrated in [Figure 4.15](#). These three development layers are implemented with virtualization and standardization of hardware and software resources provisioned in the cloud. The services to public, private, and hybrid clouds are conveyed to users through networking support over the Internet and intranets involved. It is clear that the infrastructure layer is deployed first to support IaaS services. This infrastructure layer serves as the foundation for building the platform layer of the cloud for supporting PaaS services. In turn, the platform layer is a foundation for implementing the application layer for SaaS applications. Different types of cloud services demand application of these resources separately.

The infrastructure layer is built with virtualized compute, storage, and network resources. The abstraction of these hardware resources is meant to provide the flexibility demanded by users. Internally, virtualization realizes automated provisioning of resources and optimizes the infrastructure management process. The platform layer is for general-purpose and repeated usage of the collection of software resources. This layer provides users with an environment to develop their applications, to test operation flows, and to monitor execution results and performance. The platform should be able to assure users that they have scalability, dependability, and security protection. In a way, the virtualized cloud platform serves as a “system middleware” between the infrastructure and application layers of the cloud.

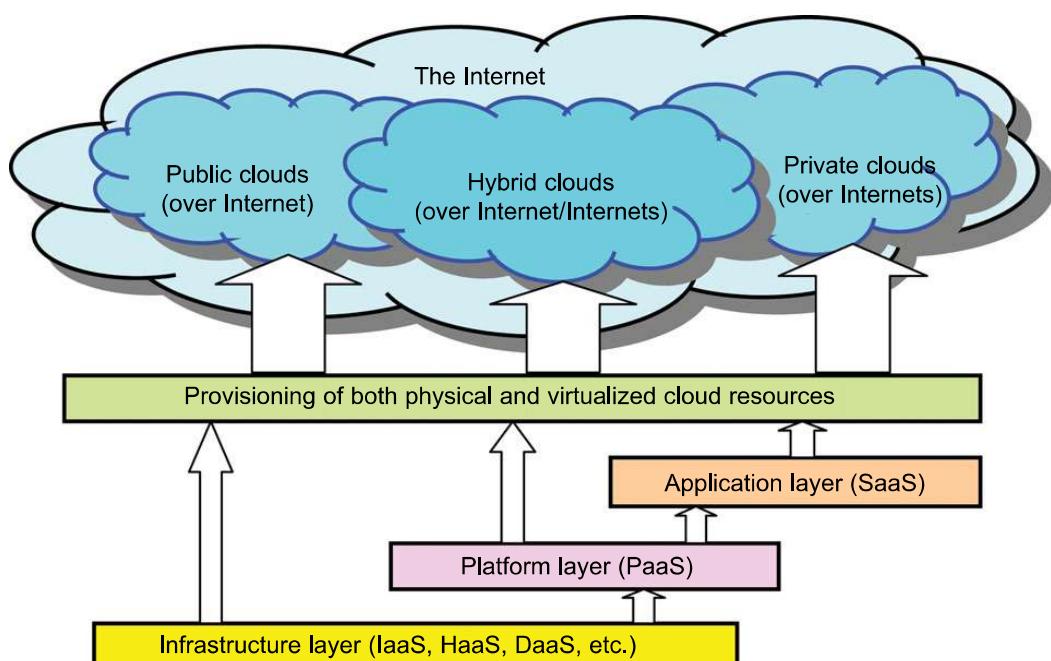


FIGURE 4.15

Layered architectural development of the cloud platform for IaaS, PaaS, and SaaS applications over the Internet.

The application layer is formed with a collection of all needed software modules for SaaS applications. Service applications in this layer include daily office management work, such as information retrieval, document processing, and calendar and authentication services. The application layer is also heavily used by enterprises in business marketing and sales, consumer relationship management (CRM), financial transactions, and supply chain management. It should be noted that not all cloud services are restricted to a single layer. Many applications may apply resources at mixed layers. After all, the three layers are built from the bottom up with a dependence relationship.

From the provider's perspective, the services at various layers demand different amounts of functionality support and resource management by providers. In general, SaaS demands the most work from the provider, PaaS is in the middle, and IaaS demands the least. For example, Amazon EC2 provides not only virtualized CPU resources to users, but also management of these provisioned resources. Services at the application layer demand more work from providers. The best example of this is the Salesforce.com CRM service, in which the provider supplies not only the hardware at the bottom layer and the software at the top layer, but also the platform and software tools for user application development and monitoring.

4.3.2.1 Market-Oriented Cloud Architecture

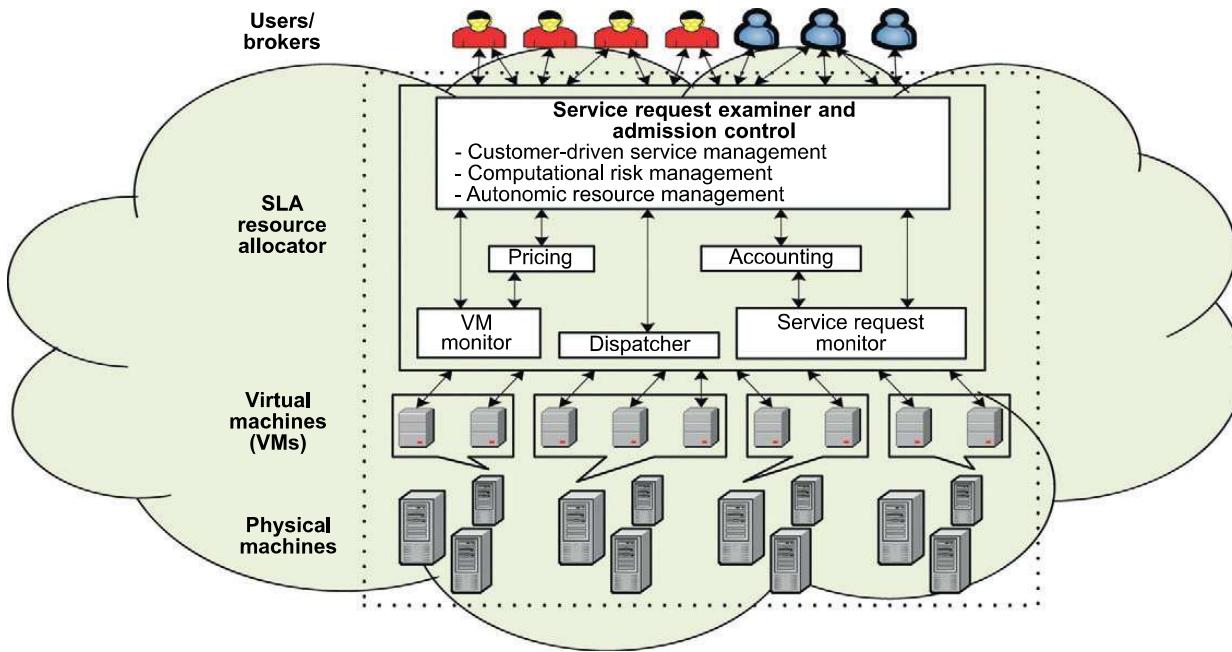
As consumers rely on cloud providers to meet more of their computing needs, they will require a specific level of QoS to be maintained by their providers, in order to meet their objectives and sustain their operations. Cloud providers consider and meet the different QoS parameters of each individual consumer as negotiated in specific SLAs. To achieve this, the providers cannot deploy traditional system-centric resource management architecture. Instead, market-oriented resource management is necessary to regulate the supply and demand of cloud resources to achieve market equilibrium between supply and demand.

The designer needs to provide feedback on economic incentives for both consumers and providers. The purpose is to promote QoS-based resource allocation mechanisms. In addition, clients can benefit from the potential cost reduction of providers, which could lead to a more competitive market, and thus lower prices. [Figure 4.16](#) shows the high-level architecture for supporting market-oriented resource allocation in a cloud computing environment. This cloud is basically built with the following entities:

Users or brokers acting on user's behalf submit service requests from anywhere in the world to the data center and cloud to be processed. The SLA resource allocator acts as the interface between the data center/cloud service provider and external users/brokers. It requires the interaction of the following mechanisms to support SLA-oriented resource management. When a service request is first submitted the service request examiner interprets the submitted request for QoS requirements before determining whether to accept or reject the request.

The request examiner ensures that there is no overloading of resources whereby many service requests cannot be fulfilled successfully due to limited resources. It also needs the latest status information regarding resource availability (from the VM Monitor mechanism) and workload processing (from the Service Request Monitor mechanism) in order to make resource allocation decisions effectively. Then it assigns requests to VMs and determines resource entitlements for allocated VMs.

The Pricing mechanism decides how service requests are charged. For instance, requests can be charged based on submission time (peak/off-peak), pricing rates (fixed/changing), or availability of

**FIGURE 4.16**

Market-oriented cloud architecture to expand/shrink leasing of resources with variation in QoS/demand from users.

(Courtesy of Raj Buyya, et al. [11])

resources (supply/demand). Pricing serves as a basis for managing the supply and demand of computing resources within the data center and facilitates in prioritizing resource allocations effectively. The Accounting mechanism maintains the actual usage of resources by requests so that the final cost can be computed and charged to users. In addition, the maintained historical usage information can be utilized by the Service Request Examiner and Admission Control mechanism to improve resource allocation decisions.

The VM Monitor mechanism keeps track of the availability of VMs and their resource entitlements. The Dispatcher mechanism starts the execution of accepted service requests on allocated VMs. The Service Request Monitor mechanism keeps track of the execution progress of service requests. Multiple VMs can be started and stopped on demand on a single physical machine to meet accepted service requests, hence providing maximum flexibility to configure various partitions of resources on the same physical machine to different specific requirements of service requests. In addition, multiple VMs can concurrently run applications based on different operating system environments on a single physical machine since the VMs are isolated from one another on the same physical machine.

4.3.2.2 Quality of Service Factors

The data center comprises multiple computing servers that provide resources to meet service demands. In the case of a cloud as a commercial offering to enable crucial business operations of companies, there are critical QoS parameters to consider in a service request, such as time, cost, reliability, and trust/security. In particular, QoS requirements cannot be static and may change over time due to continuing changes in business operations and operating environments. In short, there should

be greater importance on customers since they pay to access services in clouds. In addition, the state of the art in cloud computing has no or limited support for dynamic negotiation of SLAs between participants and mechanisms for automatic allocation of resources to multiple competing requests. Negotiation mechanisms are needed to respond to alternate offers protocol for establishing SLAs [72].

Commercial cloud offerings must be able to support customer-driven service management based on customer profiles and requested service requirements. Commercial clouds define computational risk management tactics to identify, assess, and manage risks involved in the execution of applications with regard to service requirements and customer needs. The cloud also derives appropriate market-based resource management strategies that encompass both customer-driven service management and computational risk management to sustain SLA-oriented resource allocation. The system incorporates autonomic resource management models that effectively self-manage changes in service requirements to satisfy both new service demands and existing service obligations, and leverage VM technology to dynamically assign resource shares according to service requirements.

4.3.3 Virtualization Support and Disaster Recovery

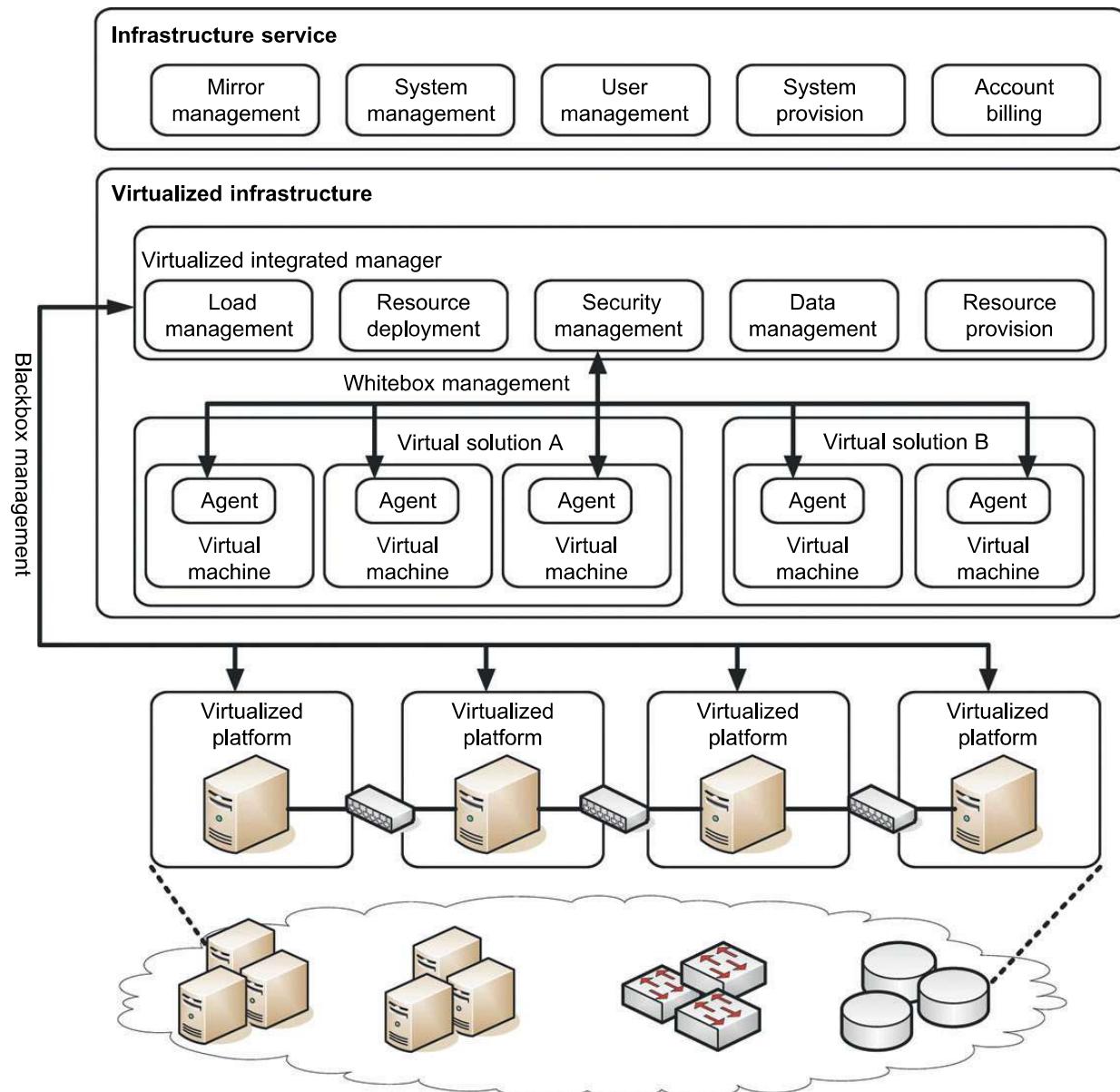
One very distinguishing feature of cloud computing infrastructure is the use of system virtualization and the modification to provisioning tools. Virtualization of servers on a shared cluster can consolidate web services. As the VMs are the containers of cloud services, the provisioning tools will first find the corresponding physical machines and deploy the VMs to those nodes before scheduling the service to run on the virtual nodes.

In addition, in cloud computing, virtualization also means the resources and fundamental infrastructure are virtualized. The user will not care about the computing resources that are used for providing the services. Cloud users do not need to know and have no way to discover physical resources that are involved while processing a service request. Also, application developers do not care about some infrastructure issues such as scalability and fault tolerance (i.e., they are virtualized). Application developers focus on service logic. [Figure 4.17](#) shows the infrastructure needed to virtualize the servers in a data center for implementing specific cloud applications.

4.3.3.1 Hardware Virtualization

In many cloud computing systems, virtualization software is used to virtualize the hardware. System virtualization software is a special kind of software which simulates the execution of hardware and runs even unmodified operating systems. Cloud computing systems use virtualization software as the running environment for legacy software such as old operating systems and unusual applications. Virtualization software is also used as the platform for developing new cloud applications that enable developers to use any operating systems and programming environments they like. The development environment and deployment environment can now be the same, which eliminates some runtime problems.

Some cloud computing providers have used virtualization technology to provide this service for developers. As mentioned before, system virtualization software is considered the hardware analog mechanism to run an unmodified operating system, usually on bare hardware directly, on top of software. [Table 4.4](#) lists some of the system virtualization software in wide use at the time of this writing. Currently, the VMs installed on a cloud computing platform are mainly used for hosting third-party programs. VMs provide flexible runtime services to free users from worrying about the system environment.

**FIGURE 4.17**

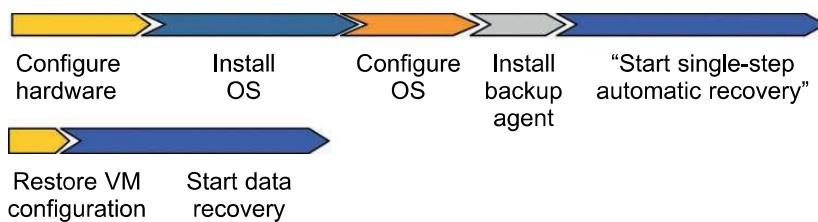
Virtualized servers, storage, and network for cloud platform construction.

(Courtesy of Zhong-Yuan Qin, SouthEast University, China)

Using VMs in a cloud computing platform ensures extreme flexibility for users. As the computing resources are shared by many users, a method is required to maximize the users' privileges and still keep them separated safely. Traditional sharing of cluster resources depends on the user and group mechanism on a system. Such sharing is not flexible. Users cannot customize the system for their special purposes. Operating systems cannot be changed. The separation is not complete.

Table 4.4 Virtualized Resources in Compute, Storage, and Network Clouds [4]

| Provider | AWS | Microsoft Azure | GAE |
|--|--|--|--|
| Compute cloud with virtual cluster of servers | x86 instruction set, Xen VMs, resource elasticity allows scalability through virtual cluster, or a third party such as RightScale must provide the cluster | Common language runtime VMs provisioned by declarative descriptions | Predefined application framework handlers written in Python, automatic scaling up and down, server failover inconsistent with the web applications |
| Storage cloud with virtual storage | Models for block store (EBS) and augmented key/blob store (SimpleDB), automatic scaling varies from EBS to fully automatic (SimpleDB, S3) | SQL Data Services (restricted view of SQL Server), Azure storage service | MegaStore/BigTable |
| Network cloud services | Declarative IP-level topology; placement details hidden, security groups restricting communication, availability zones isolate network failure, elastic IP applied | Automatic with user's declarative descriptions or roles of app. components | Fixed topology to accommodate three-tier web app. structure, scaling up and down is automatic and programmer-invisible |

**FIGURE 4.18**

Recovery overhead of a conventional disaster recovery scheme, compared with that required to recover from live migration of VMs.

An environment that meets one user's requirements often cannot satisfy another user. Virtualization allows users to have full privileges while keeping them separate.

Users have full access to their own VMs, which are completely separate from other users' VMs. Multiple VMs can be mounted on the same physical server. Different VMs may run with different OSes. We also need to establish the virtual disk storage and virtual networks needed by the VMs. The virtualized resources form a resource pool. The virtualization is carried out by special servers dedicated to generating the virtualized resource pool. The virtualized infrastructure (black box in the middle) is built with many *virtualizing integration managers*. These managers handle loads, resources, security, data, and provisioning functions. Figure 4.18 shows two VM platforms. Each platform carries out a virtual solution to a user job. All cloud services are managed in the boxes at the top.

4.3.3.2 Virtualization Support in Public Clouds

Armbrust, et al. [4] have assessed in [Table 4.4](#) three public clouds in the context of virtualization support: AWS, Microsoft Azure, and GAE. AWS provides extreme flexibility (VMs) for users to execute their own applications. GAE provides limited application-level virtualization for users to build applications only based on the services that are created by Google. Microsoft provides programming-level virtualization (.NET virtualization) for users to build their applications.

The VMware tools apply to workstations, servers, and virtual infrastructure. The Microsoft tools are used on PCs and some special servers. The XenEnterprise tool applies only to Xen-based servers. Everyone is interested in the cloud; the entire IT industry is moving toward the vision of the cloud. Virtualization leads to HA, disaster recovery, dynamic load leveling, and rich provisioning support. Both cloud computing and utility computing leverage the benefits of virtualization to provide a scalable and autonomous computing environment.

4.3.3.3 Storage Virtualization for Green Data Centers

IT power consumption in the United States has more than doubled to 3 percent of the total energy consumed in the country. The large number of data centers in the country has contributed to this energy crisis to a great extent. More than half of the companies in the Fortune 500 are actively implementing new corporate energy policies. Recent surveys from both IDC and Gartner confirm the fact that virtualization had a great impact on cost reduction from reduced power consumption in physical computing systems. This alarming situation has made the IT industry become more energy-aware. With little evolution of alternate energy resources, there is an imminent need to conserve power in all computers. Virtualization and server consolidation have already proven handy in this aspect. Green data centers and benefits of storage virtualization are considered to further strengthen the synergy of green computing.

4.3.3.4 Virtualization for IaaS

VM technology has increased in ubiquity. This has enabled users to create customized environments atop physical infrastructure for cloud computing. Use of VMs in clouds has the following distinct benefits: (1) System administrators consolidate workloads of underutilized servers in fewer servers; (2) VMs have the ability to run legacy code without interfering with other APIs; (3) VMs can be used to improve security through creation of sandboxes for running applications with questionable reliability; And (4) virtualized cloud platforms can apply performance isolation, letting providers offer some guarantees and better QoS to customer applications.

4.3.3.5 VM Cloning for Disaster Recovery

VM technology requires an advanced disaster recovery scheme. One scheme is to recover one physical machine by another physical machine. The second scheme is to recover one VM by another VM. As shown in the top timeline of [Figure 4.18](#), traditional disaster recovery from one physical machine to another is rather slow, complex, and expensive. Total recovery time is attributed to the hardware configuration, installing and configuring the OS, installing the backup agents, and the long time to restart the physical machine. To recover a VM platform, the installation and configuration times for the OS and backup agents are eliminated. Therefore, we end up with a much shorter disaster recovery time, about 40 percent of that to recover the physical machines. Virtualization aids in fast disaster recovery by VM encapsulation.

We discussed disaster recovery in [Chapters 2 and 3](#). The cloning of VMs offers an effective solution. The idea is to make a clone VM on a remote server for every running VM on a local server. Among all the clone VMs, only one needs to be active. The remote VM should be in a suspended mode. A cloud control center should be able to activate this clone VM in case of failure of the original VM, taking a snapshot of the VM to enable live migration in a minimal amount of time. The migrated VM can run on a shared Internet connection. Only updated data and modified states are sent to the suspended VM to update its state. The *Recovery Property Objective (RPO)* and *Recovery Time Objective (RTO)* are affected by the number of snapshots taken. Security of the VMs should be enforced during live migration of VMs.

4.3.4 Architectural Design Challenges

In this section, we will identify six open challenges in cloud architecture development. Armbrust, et al. [4] have observed some of these topics as both obstacles and opportunities. Plausible solutions to meet these challenges are discussed shortly.

4.3.4.1 **Challenge 1—Service Availability and Data Lock-in Problem**

The management of a cloud service by a single company is often the source of single points of failure. To achieve HA, one can consider using multiple cloud providers. Even if a company has multiple data centers located in different geographic regions, it may have common software infrastructure and accounting systems. Therefore, using multiple cloud providers may provide more protection from failures. Another availability obstacle is distributed *denial of service* (DDoS) attacks. Criminals threaten to cut off the incomes of SaaS providers by making their services unavailable. Some utility computing services offer SaaS providers the opportunity to defend against DDoS attacks by using quick scale-ups.

Software stacks have improved interoperability among different cloud platforms, but the APIs themselves are still proprietary. Thus, customers cannot easily extract their data and programs from one site to run on another. The obvious solution is to standardize the APIs so that a SaaS developer can deploy services and data across multiple cloud providers. This will rescue the loss of all data due to the failure of a single company. In addition to mitigating data lock-in concerns, standardization of APIs enables a new usage model in which the same software infrastructure can be used in both public and private clouds. Such an option could enable “surge computing,” in which the public cloud is used to capture the extra tasks that cannot be easily run in the data center of a private cloud.

4.3.4.2 **Challenge 2—Data Privacy and Security Concerns**

Current cloud offerings are essentially public (rather than private) networks, exposing the system to more attacks. Many obstacles can be overcome immediately with well-understood technologies such as encrypted storage, virtual LANs, and network middleboxes (e.g., firewalls, packet filters). For example, you could encrypt your data before placing it in a cloud. Many nations have laws requiring SaaS providers to keep customer data and copyrighted material within national boundaries.

Traditional network attacks include buffer overflows, DoS attacks, spyware, malware, rootkits, Trojan horses, and worms. In a cloud environment, newer attacks may result from hypervisor malware, guest hopping and hijacking, or VM rootkits. Another type of attack is the man-in-the-middle

attack for VM migrations. In general, passive attacks steal sensitive data or passwords. Active attacks may manipulate kernel data structures which will cause major damage to cloud servers. We will study all of these security and privacy problems on clouds in [Section 4.5](#).

4.3.4.3 Challenge 3—Unpredictable Performance and Bottlenecks

Multiple VMs can share CPUs and main memory in cloud computing, but I/O sharing is problematic. For example, to run 75 EC2 instances with the STREAM benchmark requires a mean bandwidth of 1,355 MB/second. However, for each of the 75 EC2 instances to write 1 GB files to the local disk requires a mean disk write bandwidth of only 55 MB/second. This demonstrates the problem of I/O interference between VMs. One solution is to improve I/O architectures and operating systems to efficiently virtualize interrupts and I/O channels.

Internet applications continue to become more data-intensive. If we assume applications to be “pulled apart” across the boundaries of clouds, this may complicate data placement and transport. Cloud users and providers have to think about the implications of placement and traffic at every level of the system, if they want to minimize costs. This kind of reasoning can be seen in Amazon’s development of its new CloudFront service. Therefore, data transfer bottlenecks must be removed, bottleneck links must be widened, and weak servers should be removed. We will study performance issues in [Chapter 8](#).

4.3.4.4 Challenge 4—Distributed Storage and Widespread Software Bugs

The database is always growing in cloud applications. The opportunity is to create a storage system that will not only meet this growth, but also combine it with the cloud advantage of scaling arbitrarily up and down on demand. This demands the design of efficient distributed SANs. Data centers must meet programmers’ expectations in terms of scalability, data durability, and HA. Data consistency checking in SAN-connected data centers is a major challenge in cloud computing.

Large-scale distributed bugs cannot be reproduced, so the debugging must occur at a scale in the production data centers. No data center will provide such a convenience. One solution may be a reliance on using VMs in cloud computing. The level of virtualization may make it possible to capture valuable information in ways that are impossible without using VMs. Debugging over simulators is another approach to attacking the problem, if the simulator is well designed.

4.3.4.5 Challenge 5—Cloud Scalability, Interoperability, and Standardization

The pay-as-you-go model applies to storage and network bandwidth; both are counted in terms of the number of bytes used. Computation is different depending on virtualization level. GAE automatically scales in response to load increases and decreases; users are charged by the cycles used. AWS charges by the hour for the number of VM instances used, even if the machine is idle. The opportunity here is to scale quickly up and down in response to load variation, in order to save money, but without violating SLAs.

Open Virtualization Format (OVF) describes an open, secure, portable, efficient, and extensible format for the packaging and distribution of VMs. It also defines a format for distributing software to be deployed in VMs. This VM format does not rely on the use of a specific host platform, virtualization platform, or guest operating system. The approach is to address virtual platform-agnostic packaging with certification and integrity of packaged software. The package supports virtual appliances to span more than one VM.

OVF also defines a transport mechanism for VM templates, and can apply to different virtualization platforms with different levels of virtualization. In terms of cloud standardization, we suggest the ability for virtual appliances to run on any virtual platform. We also need to enable VMs to run on heterogeneous hardware platform hypervisors. This requires hypervisor-agnostic VMs. We also need to realize cross-platform live migration between x86 Intel and AMD technologies and support legacy hardware for load balancing. All these issue are wide open for further research.

4.3.4.6 Challenge 6—Software Licensing and Reputation Sharing

Many cloud computing providers originally relied on open source software because the licensing model for commercial software is not ideal for utility computing. The primary opportunity is either for open source to remain popular or simply for commercial software companies to change their licensing structure to better fit cloud computing. One can consider using both pay-for-use and bulk-use licensing schemes to widen the business coverage.

One customer's bad behavior can affect the reputation of the entire cloud. For instance, blacklisting of EC2 IP addresses by spam-prevention services may limit smooth VM installation. An opportunity would be to create reputation-guarding services similar to the "trusted e-mail" services currently offered (for a fee) to services hosted on smaller ISPs. Another legal issue concerns the transfer of legal liability. Cloud providers want legal liability to remain with the customer, and vice versa. This problem must be solved at the SLA level. We will study reputation systems for protecting data centers in the next section.

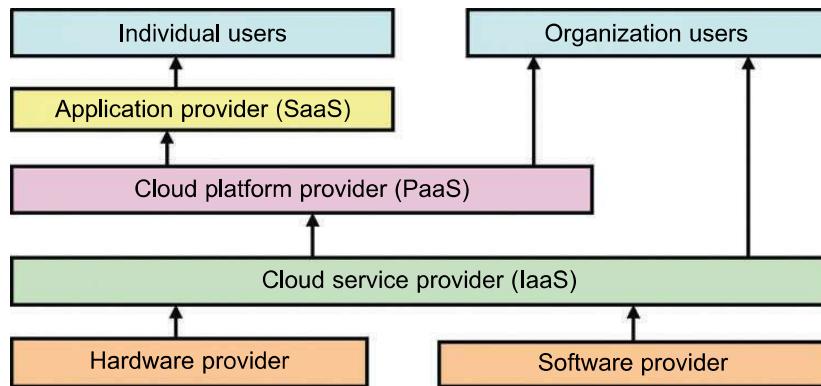
4.4 PUBLIC CLOUD PLATFORMS: GAE, AWS, AND AZURE

In this section, we will review the system architectures of four commercially available cloud platforms. These case studies will prepare readers for subsequent sections and chapters.

4.4.1 Public Clouds and Service Offerings

Cloud services are demanded by computing and IT administrators, software vendors, and end users. Figure 4.19 introduces five levels of cloud players. At the top level, individual users and organizational users demand very different services. The application providers at the SaaS level serve mainly individual users. Most business organizations are serviced by IaaS and PaaS providers. The infrastructure services (IaaS) provide compute, storage, and communication resources to both applications and organizational users. The cloud environment is defined by the PaaS or platform providers. Note that the platform providers support both infrastructure services and organizational users directly.

Cloud services rely on new advances in machine virtualization, SOA, grid infrastructure management, and power efficiency. Consumers purchase such services in the form of IaaS, PaaS, or SaaS as described earlier. Also, many cloud entrepreneurs are selling value-added utility services to massive numbers of users. The cloud industry leverages the growing demand by many enterprises and business users to outsource their computing and storage jobs to professional providers. The provider service charges are often much lower than the cost for users to replace their obsolete servers frequently. Table 4.5 summarizes the profiles of five major cloud providers by 2010 standards.

**FIGURE 4.19**

Roles of individual and organizational users and their interaction with cloud providers under various cloud service models.

Table 4.5 Five Major Cloud Platforms and Their Service Offerings [36]

| Model | IBM | Amazon | Google | Microsoft | Salesforce |
|----------------------------|--|--|---|--|---|
| PaaS | BlueCloud, WCA, RC2 | | App Engine (GAE) | Windows Azure | Force.com |
| IaaS | Ensembles | AWS | | Windows Azure | |
| SaaS | Lotus Live | | Gmail, Docs | .NET service, Dynamic CRM | Online CRM, Gifttag |
| Virtualization | | OS and Xen | Application Container | OS level/ Hypel-V | |
| Service Offerings | SOA, B2, TSAM, RAD, Web 2.0 | EC2, S3, SQS, SimpleDB | GFS, Chubby, BigTable, MapReduce | Live, SQL Hotmail | Apex, visual force, record security |
| Security Features | WebSphere2 and PowerVM tuned for protection | PKI, VPN, EBS to recover from failure | Chubby locks for security enforcement | Replicated data, rule- based access control | Admin./record security, uses metadata API |
| User Interfaces | | EC2 command-line tools | Web-based admin. console | Windows Azure portal | |
| Web API | Yes | Yes | Yes | Yes | Yes |
| Programming Support | AMI | | Python | .NET Framework | |

Note: WCA: WebSphere CloudBurst Appliance; RC2: Research Compute Cloud; RAD: Rational Application Developer; SOA: Service-Oriented Architecture; TSAM: Tivoli Service Automation Manager; EC2: Elastic Compute Cloud; S3: Simple Storage Service; SQS: Simple Queue Service; GAE: Google App Engine; AWS: Amazon Web Services; SQL: Structured Query Language; EBS: Elastic Block Store; CRM: Consumer Relationship Management.

Amazon pioneered the IaaS business in supporting e-commerce and cloud applications by millions of customers simultaneously. The elasticity in the Amazon cloud comes from the flexibility provided by the hardware and software services. EC2 provides an environment for running virtual servers on demand. S3 provides unlimited online storage space. Both EC2 and S3 are supported in the AWS platform. Microsoft offers the Azure platform for cloud applications. It has also supported the .NET service, dynamic CRM, Hotmail, and SQL applications. [Salsforce.com](#) offers extensive SaaS applications for online CRM applications using its [Force.com](#) platforms.

As [Table 4.5](#) shows, all IaaS, PaaS, and SaaS models allow users to access services over the Internet, relying entirely on the infrastructures of the cloud service providers. These models are offered based on various SLAs between the providers and the users. SLAs are more common in network services as they account for the QoS characteristics of network services. For cloud computing services, it is difficult to find a reasonable precedent for negotiating an SLA. In a broader sense, the SLAs for cloud computing address service availability, data integrity, privacy, and security protection. Blank spaces in the table refer to unknown or underdeveloped features.

4.4.2 Google App Engine (GAE)

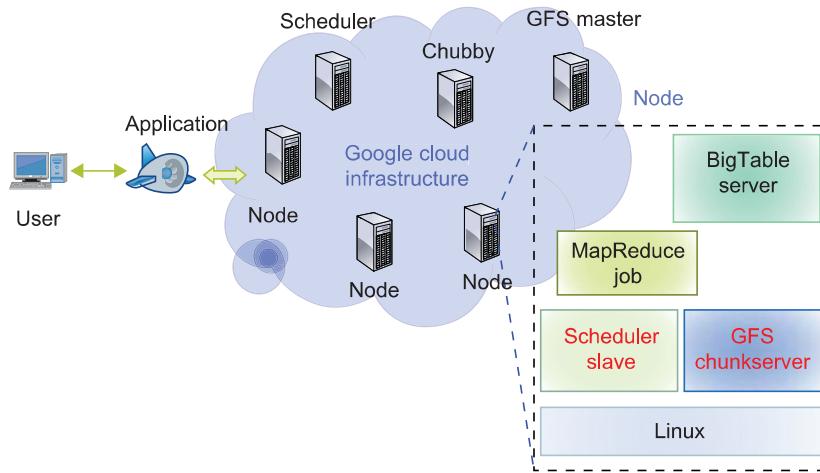
Google has the world's largest search engine facilities. The company has extensive experience in massive data processing that has led to new insights into data-center design (see [Chapter 3](#)) and novel programming models that scale to incredible sizes. The Google platform is based on its search engine expertise, but as discussed earlier with MapReduce, this infrastructure is applicable to many other areas. Google has hundreds of data centers and has installed more than 460,000 servers worldwide. For example, 200 Google data centers are used at one time for a number of cloud applications. Data items are stored in text, images, and video and are replicated to tolerate faults or failures. Here we discuss Google's App Engine (GAE) which offers a PaaS platform supporting various cloud and web applications.

4.4.2.1 Google Cloud Infrastructure

Google has pioneered cloud development by leveraging the large number of data centers it operates. For example, Google pioneered cloud services in Gmail, Google Docs, and Google Earth, among other applications. These applications can support a large number of users simultaneously with HA. Notable technology achievements include the Google File System (GFS), MapReduce, BigTable, and Chubby. In 2008, Google announced the GAE web application platform which is becoming a common platform for many small cloud service providers. This platform specializes in supporting scalable (elastic) web applications. GAE enables users to run their applications on a large number of data centers associated with Google's search engine operations.

4.4.2.2 GAE Architecture

[Figure 4.20](#) shows the major building blocks of the Google cloud platform which has been used to deliver the cloud services highlighted earlier. GFS is used for storing large amounts of data. MapReduce is for use in application program development. Chubby is used for distributed application lock services. BigTable offers a storage service for accessing structured data. These technologies are described in more detail in [Chapter 8](#). Users can interact with Google applications via the

**FIGURE 4.20**

Google cloud platform and major building blocks, the blocks shown are large clusters of low-cost servers.

(Courtesy of Kang Chen, Tsinghua University, China)

web interface provided by each application. Third-party application providers can use GAE to build cloud applications for providing services. The applications all run in data centers under tight management by Google engineers. Inside each data center, there are thousands of servers forming different clusters.

Google is one of the larger cloud application providers, although its fundamental service program is private and outside people cannot use the Google infrastructure to build their own service. The building blocks of Google's cloud computing application include the Google File System for storing large amounts of data, the MapReduce programming framework for application developers, Chubby for distributed application lock services, and BigTable as a storage service for accessing structural or semistructural data. With these building blocks, Google has built many cloud applications. Figure 4.20 shows the overall architecture of the Google cloud infrastructure. A typical cluster configuration can run the Google File System, MapReduce jobs, and BigTable servers for structure data. Extra services such as Chubby for distributed locks can also run in the clusters.

GAE runs the user program on Google's infrastructure. As it is a platform running third-party programs, application developers now do not need to worry about the maintenance of servers. GAE can be thought of as the combination of several software components. The frontend is an application framework which is similar to other web application frameworks such as ASP, J2EE, and JSP. At the time of this writing, GAE supports Python and Java programming environments. The applications can run similar to web application containers. The frontend can be used as the dynamic web serving infrastructure which can provide the full support of common technologies.

4.4.2.3 Functional Modules of GAE

The GAE platform comprises the following five major components. The GAE is not an infrastructure platform, but rather an application development platform for users. We describe the component functionalities separately.

- a. The *datastore* offers object-oriented, distributed, structured data storage services based on BigTable techniques. The datastore secures data management operations.
- b. The *application runtime environment* offers a platform for scalable web programming and execution. It supports two development languages: Python and Java.
- c. The *software development kit* (SDK) is used for local application development. The SDK allows users to execute test runs of local applications and upload application code.
- d. The *administration console* is used for easy management of user application development cycles, instead of for physical resource management.
- e. The *GAE web service infrastructure* provides special interfaces to guarantee flexible use and management of storage and network resources by GAE.

Google offers essentially free GAE services to all Gmail account owners. You can register for a GAE account or use your Gmail account name to sign up for the service. The service is free within a quota. If you exceed the quota, the page instructs you on how to pay for the service. Then you download the SDK and read the Python or Java guide to get started. Note that GAE only accepts Python, Ruby, and Java programming languages. The platform does not provide any IaaS services, unlike Amazon, which offers IaaS and PaaS. This model allows the user to deploy user-built applications on top of the cloud infrastructure that are built using the programming languages and software tools supported by the provider (e.g., Java, Python). Azure does this similarly for .NET. The user does not manage the underlying cloud infrastructure. The cloud provider facilitates support of application development, testing, and operation support on a well-defined service platform.

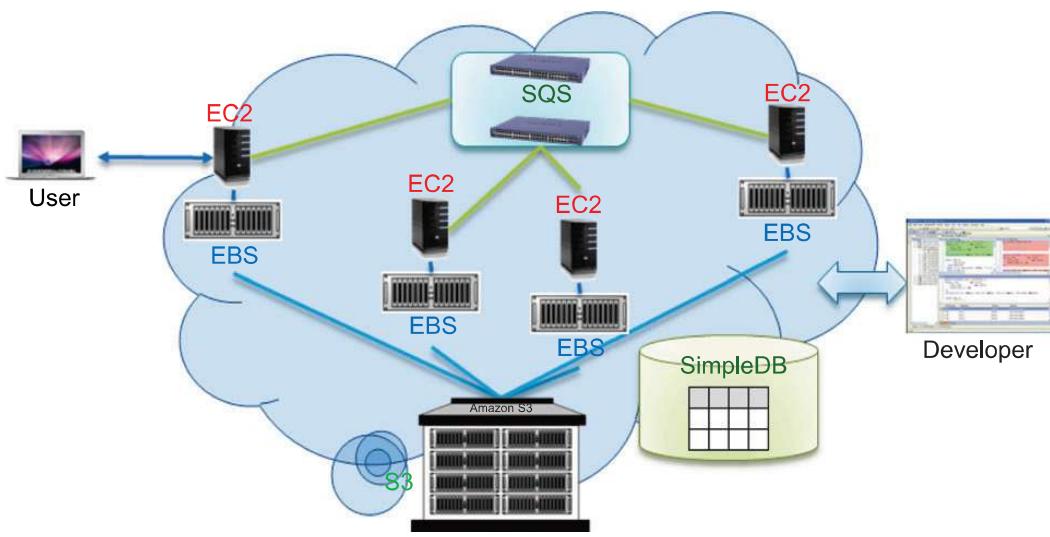
4.4.2.4 GAE Applications

Well-known GAE applications include the Google Search Engine, Google Docs, Google Earth, and Gmail. These applications can support large numbers of users simultaneously. Users can interact with Google applications via the web interface provided by each application. Third-party application providers can use GAE to build cloud applications for providing services. The applications are all run in the Google data centers. Inside each data center, there might be thousands of server nodes to form different clusters. (See the previous section.) Each cluster can run multipurpose servers.

GAE supports many web applications. One is a storage service to store application-specific data in the Google infrastructure. The data can be persistently stored in the backend storage server while still providing the facility for queries, sorting, and even transactions similar to traditional database systems. GAE also provides Google-specific services, such as the Gmail account service (which is the login service, that is, applications can use the Gmail account directly). This can eliminate the tedious work of building customized user management components in web applications. Thus, web applications built on top of GAE can use the APIs authenticating users and sending e-mail using Google accounts.

4.4.3 Amazon Web Services (AWS)

VMs can be used to share computing resources both flexibly and safely. Amazon has been a leader in providing public cloud services (<http://aws.amazon.com/>). Amazon applies the IaaS model in providing its services. Figure 4.21 shows the AWS architecture. EC2 provides the virtualized platforms to the host VMs where the cloud application can run. S3 (Simple Storage Service) provides the object-oriented storage service for users. EBS (Elastic Block Service) provides the block storage

**FIGURE 4.21**

Amazon cloud computing infrastructure (Key services are identified here; many more are listed in [Table 4.6](#)).

(Courtesy of Kang Chen, Tsinghua University, China)

interface which can be used to support traditional applications. SQS stands for Simple Queue Service, and its job is to ensure a reliable message service between two processes. The message can be kept reliably even when the receiver processes are not running. Users can access their objects through SOAP with either browsers or other client programs which support the SOAP standard.

[Table 4.6](#) summarizes the service offerings by AWS in 12 application tracks. Details of EC2, S3, and EBS are available in [Chapter 6](#) where we discuss programming examples. Amazon offers queuing and notification services (SQS and SNS), which are implemented in the AWS cloud. Note brokering systems run very efficiently in clouds and offer a striking model for controlling sensors and providing office support of smartphones and tablets. Different from Google, Amazon provides a more flexible cloud computing platform for developers to build cloud applications. Small and medium-size companies can put their business on the Amazon cloud platform. Using the AWS platform, they can service large numbers of Internet users and make profits through those paid services.

ELB automatically distributes incoming application traffic across multiple Amazon EC2 instances and allows user to avoid nonoperating nodes and to equalize load on functioning images. Both auto-scaling and ELB are enabled by CloudWatch which monitors running instances. CloudWatch is a web service that provides monitoring for AWS cloud resources, starting with Amazon EC2. It provides customers with visibility into resource utilization, operational performance, and overall demand patterns, including metrics such as CPU utilization, disk reads and writes, and network traffic.

Amazon (like Azure) offers a *Relational Database Service (RDS)* with a messaging interface to be covered in [Section 4.1](#). The Elastic MapReduce capability is equivalent to Hadoop running on the basic EC2 offering. AWS Import/Export allows one to ship large volumes of data to and from EC2 by shipping physical disks; it is well known that this is often the highest bandwidth connection between geographically distant systems. Amazon CloudFront implements a content distribution

Table 4.6 AWS Offerings in 2011

| Service Area | Service Modules and Abbreviated Names |
|-----------------------------|--|
| Compute | Elastic Compute Cloud (EC2), Elastic MapReduce, Auto Scaling |
| Messaging | Simple Queue Service (SQS), Simple Notification Service (SNS) |
| Storage | Simple Storage Service (S3), Elastic Block Storage (EBS), AWS Import/Export |
| Content Delivery | Amazon CloudFront |
| Monitoring | Amazon CloudWatch |
| Support | AWS Premium Support |
| Database | Amazon SimpleDB, Relational Database Service (RDS) |
| Networking | Virtual Private Cloud (VPC) (Example 4.1 , Figure 4.6), Elastic Load Balancing |
| Web Traffic | Alexa Web Information Service, Alexa Web Sites |
| E-Commerce | Fulfillment Web Service (FWS) |
| Payments and Billing | Flexible Payments Service (FPS), Amazon DevPay |
| Workforce | Amazon Mechanical Turk |

(Courtesy of Amazon, <http://aws.amazon.com> [3])

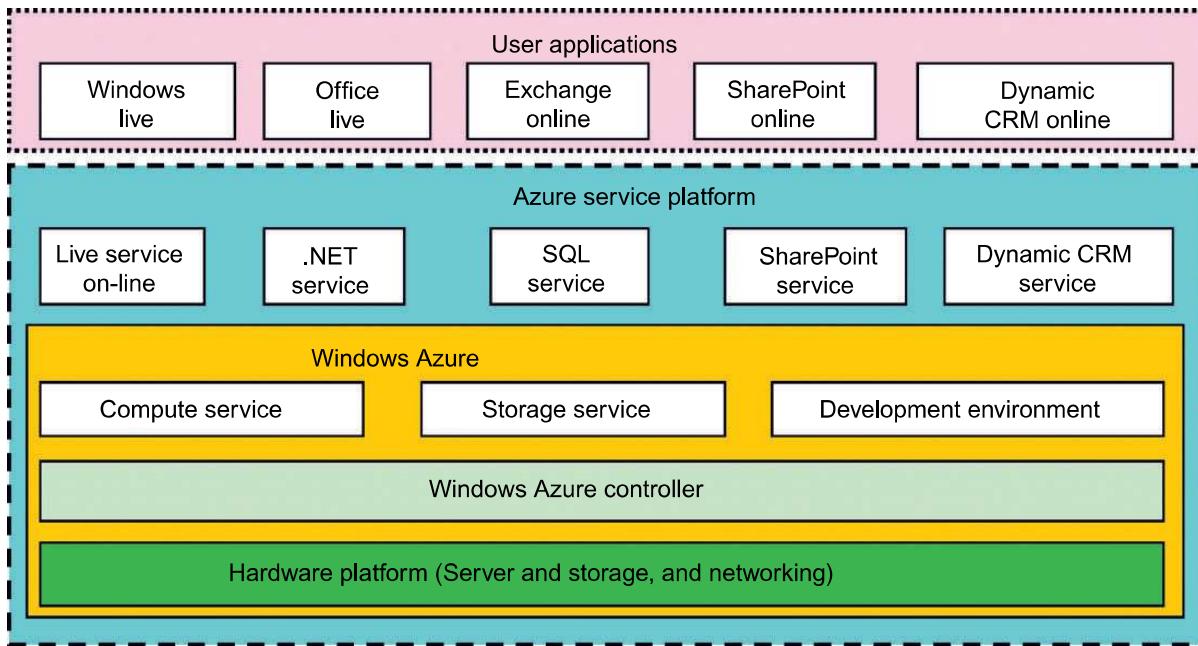
network. Amazon DevPay is a simple-to-use online billing and account management service that makes it easy for businesses to sell applications that are built into or run on top of AWS.

FPS provides developers of commercial systems on AWS with a convenient way to charge Amazon's customers that use such services built on AWS. Customers can pay using the same login credentials, shipping address, and payment information they already have on file with Amazon. The FWS allows merchants to access Amazon's fulfillment capabilities through a simple web service interface. Merchants can send order information to Amazon to fulfill customer orders on their behalf. In July 2010, Amazon offered MPI clusters and cluster compute instances. The AWS cluster compute instances use hardware-assisted virtualization instead of the para-virtualization used by other instance types and requires booting from the EBS. Users are freed to create a new AMI as needed.

4.4.4 Microsoft Windows Azure

In 2008, Microsoft launched a Windows Azure platform to meet the challenges in cloud computing. This platform is built over Microsoft data centers. [Figure 4.22](#) shows the overall architecture of Microsoft's cloud platform. The platform is divided into three major component platforms. Windows Azure offers a cloud platform built on Windows OS and based on Microsoft virtualization technology. Applications are installed on VMs deployed on the data-center servers. Azure manages all servers, storage, and network resources of the data center. On top of the infrastructure are the various services for building different cloud applications. Cloud-level services provided by the Azure platform are introduced below. More details on Azure services are given in [Chapter 6](#).

- **Live service** Users can visit Microsoft Live applications and apply the data involved across multiple machines concurrently.
- **.NET service** This package supports application development on local hosts and execution on cloud machines.

**FIGURE 4.22**

Microsoft Windows Azure platform for cloud computing.

(Courtesy of Microsoft, 2010, <http://www.microsoft.com/windowsazure>)

- **SQL Azure** This function makes it easier for users to visit and use the relational database associated with the SQL server in the cloud.
- **SharePoint service** This provides a scalable and manageable platform for users to develop their special business applications in upgraded web services.
- **Dynamic CRM service** This provides software developers a business platform in managing CRM applications in financing, marketing, and sales and promotions.

All these cloud services in Azure can interact with traditional Microsoft software applications, such as Windows Live, Office Live, Exchange online, SharePoint online, and dynamic CRM online. The Azure platform applies the standard web communication protocols SOAP and REST. The Azure service applications allow users to integrate the cloud application with other platforms or third-party clouds. You can download the Azure development kit to run a local version of Azure. The powerful SDK allows Azure applications to be developed and debugged on the Windows hosts.

4.5 INTER-CLOUD RESOURCE MANAGEMENT

This section characterizes the various cloud service models and their extensions. The cloud service trends are outlined. Cloud resource management and intercloud resource exchange schemes are reviewed. We will discuss the defense of cloud resources against network threats in Section 4.6.

4.5.1 Extended Cloud Computing Services

Figure 4.23 shows six layers of cloud services, ranging from hardware, network, and collocation to infrastructure, platform, and software applications. We already introduced the top three service layers as SaaS, PaaS, and IaaS, respectively. The cloud platform provides PaaS, which sits on top of the IaaS infrastructure. The top layer offers SaaS. These must be implemented on the cloud platforms provided. Although the three basic models are dissimilar in usage, as shown in Table 4.7, they are built one on top of another. The implication is that one cannot launch SaaS applications with a cloud platform. The cloud platform cannot be built if compute and storage infrastructures are not there.

The bottom three layers are more related to physical requirements. The bottommost layer provides *Hardware as a Service (HaaS)*. The next layer is for interconnecting all the hardware components, and is simply called *Network as a Service (NaaS)*. *Virtual LANs* fall within the scope of NaaS. The next layer up offers *Location as a Service (LaaS)*, which provides a collocation service to house, power, and secure all the physical hardware and network resources. Some authors say this layer provides *Security as a Service ("SaaS")*. The cloud infrastructure layer can be further subdivided as *Data as a Service (DaaS)* and *Communication as a Service (CaaS)* in addition to compute and storage in IaaS.

We will examine commercial trends in cloud services in subsequent sections. Here we will mainly cover the top three layers with some success stories of cloud computing. As shown in Table 4.7, cloud players are divided into three classes: (1) cloud service providers and IT administrators, (2) software developers or vendors, and (3) end users or business users. These cloud players vary in their roles under the IaaS, PaaS, and SaaS models. The table entries distinguish the three cloud models as viewed by different players. From the software vendors' perspective, application performance on a given cloud platform is most important. From the providers' perspective, cloud infrastructure

| | | | |
|---|---|--|--|
| Cloud application (SaaS) | Concur, RightNOW, Teleo, Kenexa, Webex, Blackbaud, salesforce.com, Netsuite, Kenexa, etc. | | |
| Cloud software environment (PaaS) | Force.com, App Engine, Facebook, MS Azure, NetSuite, IBM BlueCloud, SGI Cyclone, eBay | | |
| Cloud software infrastructure | Amazon AWS, OpSource Cloud, IBM Ensembles, Rackspace cloud, Windows Azure, HP, Banknorth | | |
| Collocation cloud services (LaaS) | Savvis, Internap, NTTCommunications, Digital Realty Trust, 365 Main | | |
| Network cloud services (NaaS) | Owest, AT&T, AboveNet | | |
| Hardware/Virtualization cloud services (HaaS) | VMware, Intel, IBM, XenEnterprise | | |

FIGURE 4.23

A stack of six layers of cloud services and their providers.

(Courtesy of T. Chou, Active Book Express, 2010 [16])

Table 4.7 Cloud Differences in Perspectives of Providers, Vendors, and Users

| Cloud Players | IaaS | PaaS | SaaS |
|-----------------------------------|--------------------------|---|----------------------------------|
| IT administrators/cloud providers | Monitor SLAs | Monitor SLAs and enable service platforms | Monitor SLAs and deploy software |
| Software developers (vendors) | To deploy and store data | Enabling platforms via configurators and APIs | Develop and deploy software |
| End users or business users | To deploy and store data | To develop and test web software | Use business software |

performance is the primary concern. From the end users' perspective, the quality of services, including security, is the most important.

4.5.1.1 Cloud Service Tasks and Trends

Cloud services are introduced in five layers. The top layer is for SaaS applications, as further subdivided into the five application areas in Figure 4.23, mostly for business applications. For example, CRM is heavily practiced in business promotion, direct sales, and marketing services. CRM offered the first SaaS on the cloud successfully. The approach is to widen market coverage by investigating customer behaviors and revealing opportunities by statistical analysis. SaaS tools also apply to distributed collaboration, and financial and human resources management. These cloud services have been growing rapidly in recent years.

PaaS is provided by Google, [Salesforce.com](#), and Facebook, among others. IaaS is provided by Amazon, Windows Azure, and RackRack, among others. Collocation services require multiple cloud providers to work together to support supply chains in manufacturing. Network cloud services provide communications such as those by AT&T, Qwest, and AboveNet. Details can be found in Clou's introductory book on business clouds [18]. The vertical cloud services in Figure 4.25 refer to a sequence of cloud services that are mutually supportive. Often, cloud mashup is practiced in vertical cloud applications.

4.5.1.2 Software Stack for Cloud Computing

Despite the various types of nodes in the cloud computing cluster, the overall software stacks are built from scratch to meet rigorous goals (see Table 4.7). Developers have to consider how to design the system to meet critical requirements such as high throughput, HA, and fault tolerance. Even the operating system might be modified to meet the special requirement of cloud data processing. Based on the observations of some typical cloud computing instances, such as Google, Microsoft, and Yahoo!, the overall software stack structure of cloud computing software can be viewed as layers. Each layer has its own purpose and provides the interface for the upper layers just as the traditional software stack does. However, the lower layers are not completely transparent to the upper layers.

The platform for running cloud computing services can be either physical servers or virtual servers. By using VMs, the platform can be flexible, that is, the running services are not bound to specific hardware platforms. This brings flexibility to cloud computing platforms. The software layer on top of the platform is the layer for storing massive amounts of data. This layer acts like

the file system in a traditional single machine. Other layers running on top of the file system are the layers for executing cloud computing applications. They include the database storage system, programming for large-scale clusters, and data query language support. The next layers are the components in the software stack.

4.5.1.3 Runtime Support Services

As in a cluster environment, there are also some runtime supporting services in the cloud computing environment. Cluster monitoring is used to collect the runtime status of the entire cluster. One of the most important facilities is the cluster job management system introduced in [Chapter 2](#). The scheduler queues the tasks submitted to the whole cluster and assigns the tasks to the processing nodes according to node availability. The distributed scheduler for the cloud application has special characteristics that can support cloud applications, such as scheduling the programs written in MapReduce style. The runtime support system keeps the cloud cluster working properly with high efficiency.

Runtime support is software needed in browser-initiated applications applied by thousands of cloud customers. The SaaS model provides the software applications as a service, rather than letting users purchase the software. As a result, on the customer side, there is no upfront investment in servers or software licensing. On the provider side, costs are rather low, compared with conventional hosting of user applications. The customer data is stored in the cloud that is either vendor proprietary or a publicly hosted cloud supporting PaaS and IaaS.

4.5.2 Resource Provisioning and Platform Deployment

The emergence of computing clouds suggests fundamental changes in software and hardware architecture. Cloud architecture puts more emphasis on the number of processor cores or VM instances. Parallelism is exploited at the cluster node level. In this section, we will discuss techniques to provision computer resources or VMs. Then we will talk about storage allocation schemes to interconnect distributed computing infrastructures by harnessing the VMs dynamically.

4.5.2.1 Provisioning of Compute Resources (VMs)

Providers supply cloud services by signing SLAs with end users. The SLAs must commit sufficient resources such as CPU, memory, and bandwidth that the user can use for a preset period. Underprovisioning of resources will lead to broken SLAs and penalties. Overprovisioning of resources will lead to resource underutilization, and consequently, a decrease in revenue for the provider. Deploying an autonomous system to efficiently provision resources to users is a challenging problem. The difficulty comes from the unpredictability of consumer demand, software and hardware failures, heterogeneity of services, power management, and conflicts in signed SLAs between consumers and service providers.

Efficient VM provisioning depends on the cloud architecture and management of cloud infrastructures. Resource provisioning schemes also demand fast discovery of services and data in cloud computing infrastructures. In a virtualized cluster of servers, this demands efficient installation of VMs, live VM migration, and fast recovery from failures. To deploy VMs, users treat them as physical hosts with customized operating systems for specific applications. For example, Amazon's EC2 uses Xen as the virtual machine monitor (VMM). The same VMM is used in IBM's Blue Cloud.

In the EC2 platform, some predefined VM templates are also provided. Users can choose different kinds of VMs from the templates. IBM's Blue Cloud does not provide any VM templates. In general, any type of VM can run on top of Xen. Microsoft also applies virtualization in its Azure cloud platform. The provider should offer resource-economic services. Power-efficient schemes for caching, query processing, and thermal management are mandatory due to increasing energy waste by heat dissipation from data centers. Public or private clouds promise to streamline the on-demand provisioning of software, hardware, and data as a service, achieving economies of scale in IT deployment and operation.

4.5.2.2 Resource Provisioning Methods

Figure 4.24 shows three cases of static cloud resource provisioning policies. In case (a), overprovisioning with the peak load causes heavy resource waste (shaded area). In case (b), underprovisioning (along the capacity line) of resources results in losses by both user and provider in that paid demand by the users (the shaded area above the capacity) is not served and wasted resources still exist for those demanded areas below the provisioned capacity. In case (c), the constant provisioning of resources with fixed capacity to a declining user demand could result in even worse resource waste. The user may give up the service by canceling the demand, resulting in reduced revenue for the provider. Both the user and provider may be losers in resource provisioning without elasticity.

Three resource-provisioning methods are presented in the following sections. The *demand-driven method* provides static resources and has been used in grid computing for many years. The *event-driven method* is based on predicted workload by time. The *popularity-driven method* is based

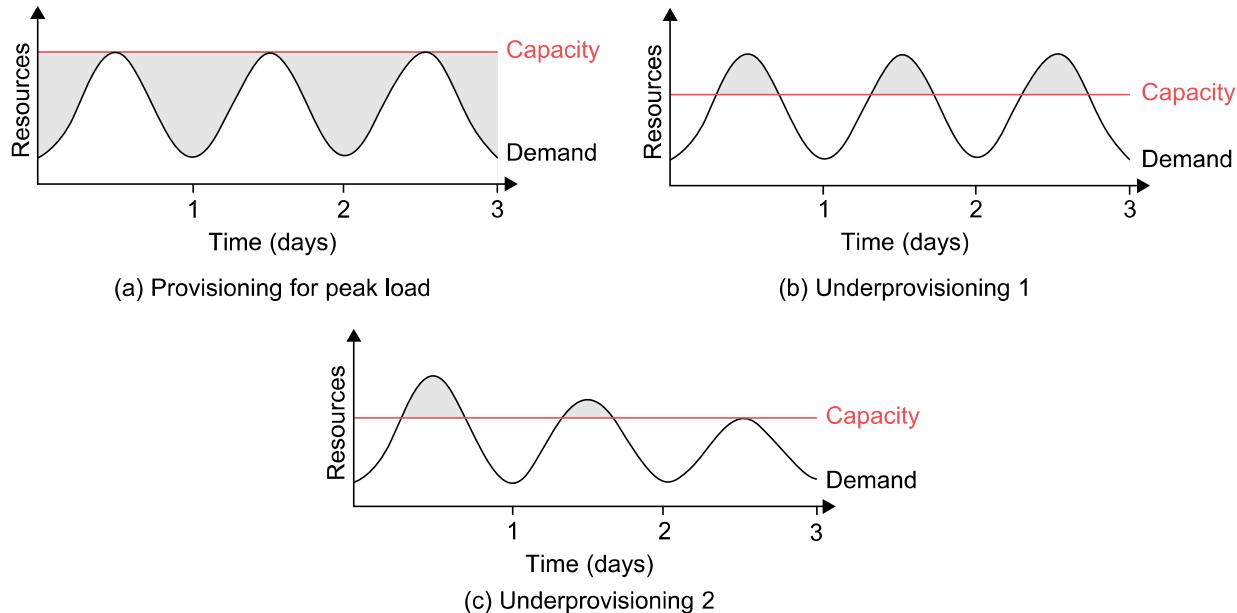


FIGURE 4.24

Three cases of cloud resource provisioning without elasticity: (a) heavy waste due to overprovisioning, (b) underprovisioning and (c) under- and then overprovisioning.

(Courtesy of Armbrust, et al., UC Berkeley, 2009 [4])

on Internet traffic monitored. We characterize these resource provisioning methods as follows (see [Figure 4.25](#)).

4.5.2.3 Demand-Driven Resource Provisioning

This method adds or removes computing instances based on the current utilization level of the allocated resources. The demand-driven method automatically allocates two Xeon processors for the user application, when the user was using one Xeon processor more than 60 percent of the time for an extended period. In general, when a resource has surpassed a threshold for a certain amount of time, the scheme increases that resource based on demand. When a resource is below a threshold for a certain amount of time, that resource could be decreased accordingly. Amazon implements such an auto-scale feature in its EC2 platform. This method is easy to implement. The scheme does not work out right if the workload changes abruptly.

The x-axis in [Figure 4.25](#) is the time scale in milliseconds. In the beginning, heavy fluctuations of CPU load are encountered. All three methods have demanded a few VM instances initially. Gradually, the utilization rate becomes more stabilized with a maximum of 20 VMs (100 percent utilization) provided for demand-driven provisioning in [Figure 4.25\(a\)](#). However, the event-driven method reaches a stable peak of 17 VMs toward the end of the event and drops quickly in [Figure 4.25\(b\)](#). The popularity provisioning shown in [Figure 4.25\(c\)](#) leads to a similar fluctuation with peak VM utilization in the middle of the plot.

4.5.2.4 Event-Driven Resource Provisioning

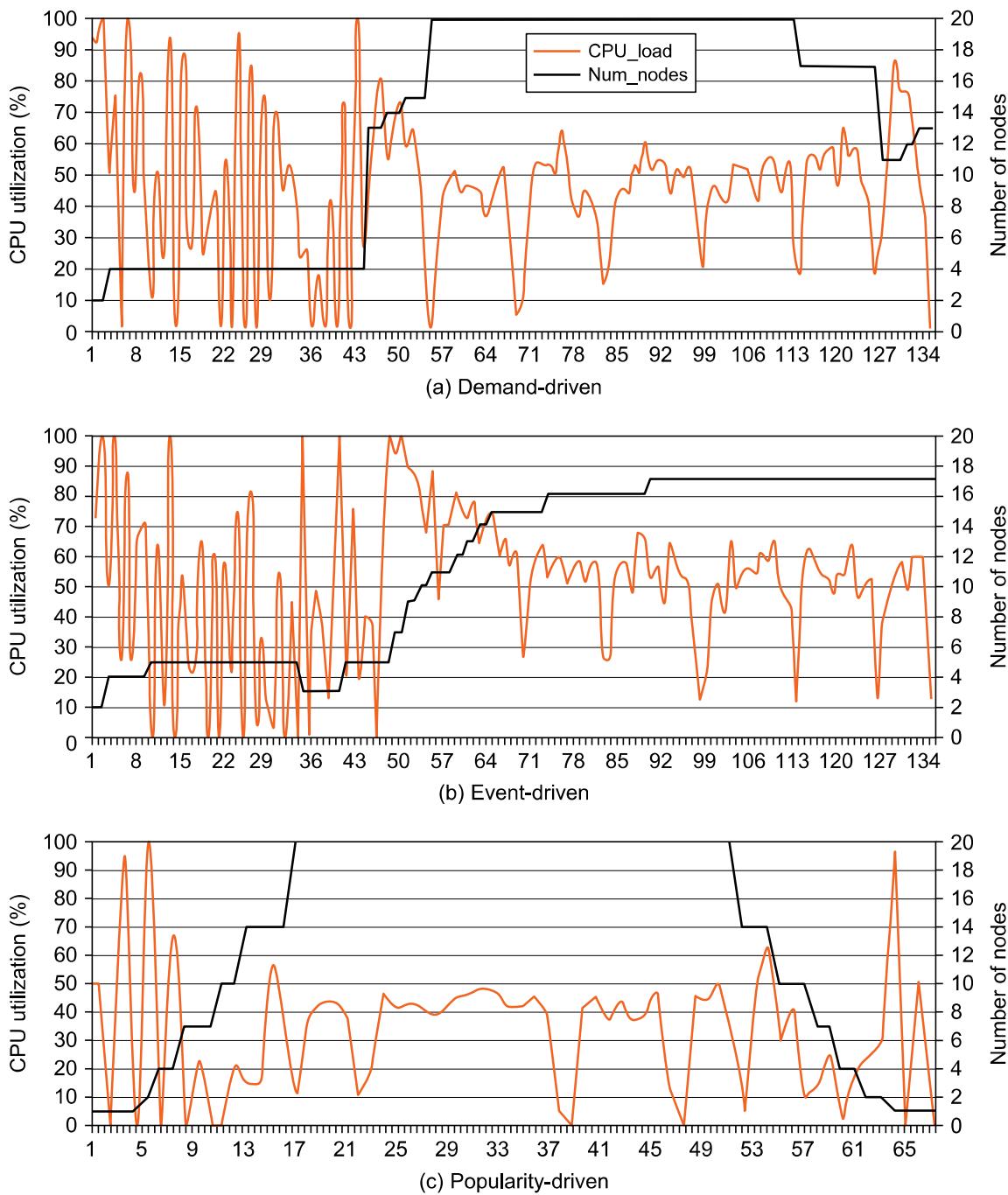
This scheme adds or removes machine instances based on a specific time event. The scheme works better for seasonal or predicted events such as Christmastime in the West and the Lunar New Year in the East. During these events, the number of users grows before the event period and then decreases during the event period. This scheme anticipates peak traffic before it happens. The method results in a minimal loss of QoS, if the event is predicted correctly. Otherwise, wasted resources are even greater due to events that do not follow a fixed pattern.

4.5.2.5 Popularity-Driven Resource Provisioning

In this method, the Internet searches for popularity of certain applications and creates the instances by popularity demand. The scheme anticipates increased traffic with popularity. Again, the scheme has a minimal loss of QoS, if the predicted popularity is correct. Resources may be wasted if traffic does not occur as expected. In [Figure 4.25\(c\)](#), EC2 performance by CPU utilization rate (the dark curve with the percentage scale shown on the left) is plotted against the number of VMs provisioned (the light curves with scale shown on the right, with a maximum of 20 VMs provisioned).

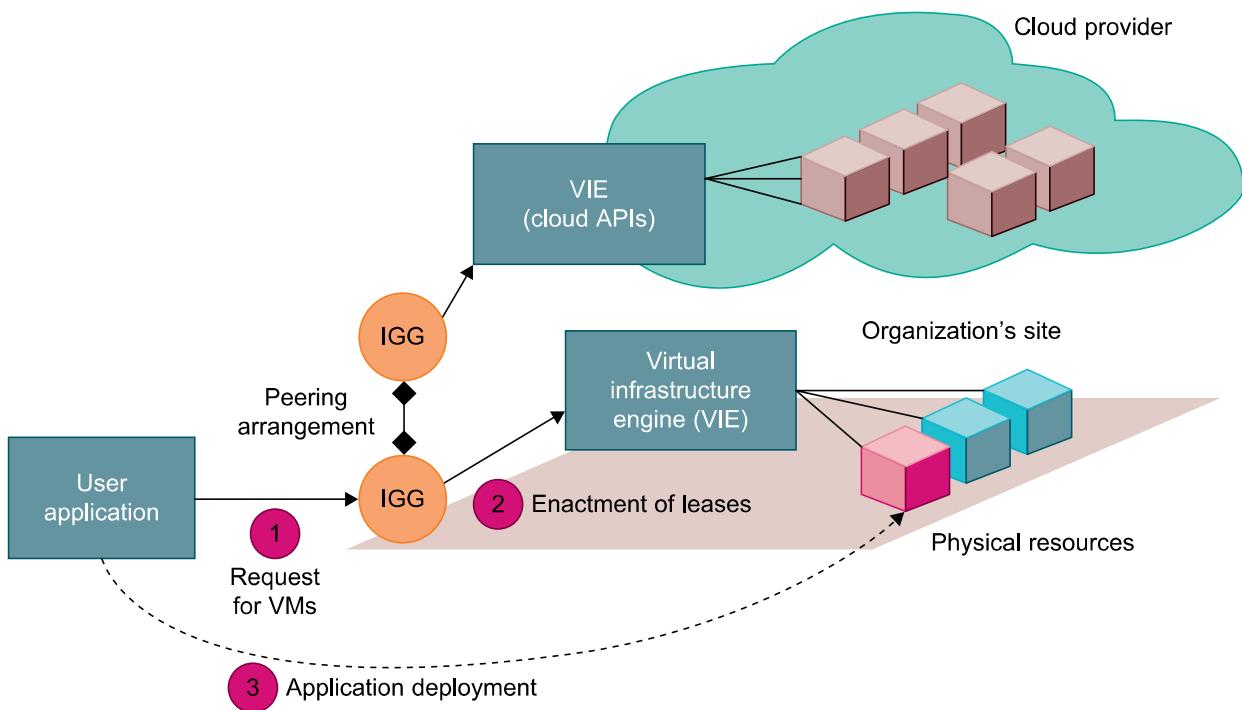
4.5.2.6 Dynamic Resource Deployment

The cloud uses VMs as building blocks to create an execution environment across multiple resource sites. The InterGrid-managed infrastructure was developed by a Melbourne University group [19]. Dynamic resource deployment can be implemented to achieve scalability in performance. The InterGrid is a Java-implemented software system that lets users create execution cloud environments on top of all participating grid resources. Peering arrangements established between gateways enable the allocation of resources from multiple grids to establish the execution environment. In [Figure 4.26](#), a scenario is illustrated by which an *intergrid gateway (IGG)* allocates resources from a local cluster to

**FIGURE 4.25**

EC2 performance results on the AWS EC2 platform, collected from experiments at the University of Southern California using three resource provisioning methods.

(Courtesy of Ken Wu, USC)

**FIGURE 4.26**

Cloud resource deployment using an IGG (intergrid gateway) to allocate the VMs from a Local cluster to interact with the IGG of a public cloud provider.

(Courtesy of Constanzo, et al. © IEEE [21])

deploy applications in three steps: (1) requesting the VMs, (2) enacting the leases, and (3) deploying the VMs as requested. Under peak demand, this IGG interacts with another IGG that can allocate resources from a cloud computing provider.

A grid has predefined peering arrangements with other grids, which the IGG manages. Through multiple IGGs, the system coordinates the use of InterGrid resources. An IGG is aware of the peering terms with other grids, selects suitable grids that can provide the required resources, and replies to requests from other IGGs. Request redirection policies determine which peering grid InterGrid selects to process a request and a price for which that grid will perform the task. An IGG can also allocate resources from a cloud provider. The cloud system creates a virtual environment to help users deploy their applications. These applications use the distributed grid resources.

The InterGrid allocates and provides a *distributed virtual environment* (DVE). This is a virtual cluster of VMs that runs isolated from other virtual clusters. A component called the *DVE manager* performs resource allocation and management on behalf of specific user applications. The core component of the IGG is a scheduler for implementing provisioning policies and peering with other gateways. The communication component provides an asynchronous message-passing mechanism. Received messages are handled in parallel by a thread pool.

4.5.2.7 Provisioning of Storage Resources

The data storage layer is built on top of the physical or virtual servers. As the cloud computing applications often provide service to users, it is unavoidable that the data is stored in the clusters of the cloud provider. The service can be accessed anywhere in the world. One example is e-mail systems. A typical large e-mail system might have millions of users and each user can have thousands of e-mails and consume multiple gigabytes of disk space. Another example is a web searching application. In storage technologies, hard disk drives may be augmented with solid-state drives in the future. This will provide reliable and high-performance data storage. The biggest barriers to adopting flash memory in data centers have been price, capacity, and, to some extent, a lack of sophisticated query-processing techniques. However, this is about to change as the I/O bandwidth of solid-state drives becomes too impressive to ignore.

A distributed file system is very important for storing large-scale data. However, other forms of data storage also exist. Some data does not need the namespace of a tree structure file system, and instead, databases are built with stored data files. In cloud computing, another form of data storage is (Key, Value) pairs. Amazon S3 service uses SOAP to access the objects stored in the cloud. **Table 4.8** outlines three cloud storage services provided by Google, Hadoop, and Amazon.

Many cloud computing companies have developed large-scale data storage systems to keep huge amount of data collected every day. For example, Google's GFS stores web data and some other data, such as geographic data for Google Earth. A similar system from the open source community is the Hadoop Distributed File System (HDFS) for Apache. Hadoop is the open source implementation of Google's cloud computing infrastructure. Similar systems include Microsoft's Cosmos file system for the cloud.

Despite the fact that the storage service or distributed file system can be accessed directly, similar to traditional databases, cloud computing does provide some forms of structure or semistructure database processing capability. For example, applications might want to process the information contained in a web page. Web pages are an example of semistructural data in HTML format. If some forms of database capability can be used, application developers will construct their application logic more easily. Another reason to build a database-like service in cloud computing is that it will be quite convenient for traditional application developers to code for the cloud platform. Databases are quite common as the underlying storage device for many applications.

Thus, such developers can think in the same way they do for traditional software development. Hence, in cloud computing, it is necessary to build databases like large-scale systems based on data

Table 4.8 Storage Services in Three Cloud Computing Systems

| Storage System | Features |
|--------------------------------------|---|
| GFS: Google File System | Very large sustainable reading and writing bandwidth, mostly continuous accessing instead of random accessing. The programming interface is similar to that of the POSIX file system accessing interface. |
| HDFS: Hadoop Distributed File System | The open source clone of GFS. Written in Java. The programming interfaces are similar to POSIX but not identical. |
| Amazon S3 and EBS | S3 is used for retrieving and storing data from/to remote servers. EBS is built on top of S3 for using virtual disks in running EC2 instances. |

storage or distributed file systems. The scale of such a database might be quite large for processing huge amounts of data. The main purpose is to store the data in structural or semi-structural ways so that application developers can use it easily and build their applications rapidly. Traditional databases will meet the performance bottleneck while the system is expanded to a larger scale. However, some real applications do not need such strong consistency. The scale of such databases can be quite large. Typical cloud databases include BigTable from Google, SimpleDB from Amazon, and the SQL service from Microsoft Azure.

4.5.3 Virtual Machine Creation and Management

In this section, we will consider several issues for cloud infrastructure management. First, we will consider the resource management of independent service jobs. Then we will consider how to execute third-party cloud applications. Cloud-loading experiments are used by a Melbourne research group on the French Grid'5000 system. This experimental setting illustrates VM creation and management. This case study example reveals major VM management issues and suggests some plausible solutions for workload-balanced execution. Figure 4.27 shows the interactions among VM managers for cloud creation and management. The managers provide a public API for users to submit and control the VMs.

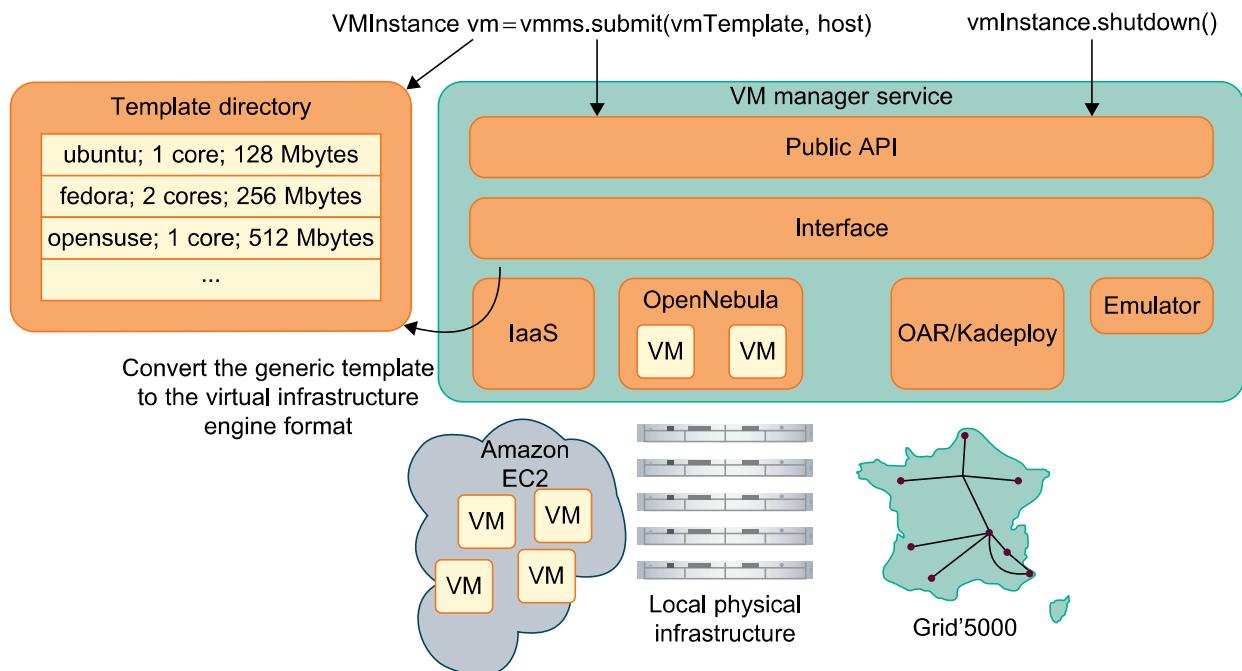


FIGURE 4.27

Interactions among VM managers for cloud creation and management; the manager provides a public API for users to submit and control the VMs.

(Courtesy of Constanzo, et al. © IEEE [21])

4.5.3.1 Independent Service Management

Independent services request facilities to execute many unrelated tasks. Commonly, the APIs provided are some web services that the developer can use conveniently. In Amazon cloud computing infrastructure, SQS is constructed for providing a reliable communication service between different providers. Even the endpoint does not run while another entity has posted a message in SQS. By using independent service providers, the cloud applications can run different services at the same time. Some other services are used for providing data other than the compute or storage services.

4.5.3.2 Running Third-Party Applications

Cloud platforms have to provide support for building applications that are constructed by third-party application providers or programmers. As current web applications are often provided by using Web 2.0 forms (interactive applications with Ajax), the programming interfaces are different from the traditional programming interfaces such as functions in runtime libraries. The APIs are often in the form of services. Web service application engines are often used by programmers for building applications. The web browsers are the user interface for end users.

In addition to gateway applications, the cloud computing platform provides the extra capabilities of accessing backend services or underlying data. As examples, GAE and Microsoft Azure apply their own cloud APIs to get special cloud services. The WebSphere application engine is deployed by IBM for Blue Cloud. It can be used to develop any kind of web application written in Java. In EC2, users can use any kind of application engine that can run in VM instances.

4.5.3.3 Virtual Machine Manager

The VM manager is the link between the gateway and resources. The gateway doesn't share physical resources directly, but relies on virtualization technology for abstracting them. Hence, the actual resources it uses are VMs. The manager manages VMs deployed on a set of physical resources. The VM manager implementation is generic so that it can connect with different VIEs. Typically, VIEs can create and stop VMs on a physical cluster. The Melbourne group has developed managers for OpenNebula, Amazon EC2, and French Grid'5000. The manager using the OpenNebula OS (www.opennebula.org) to deploy VMs on local clusters.

OpenNebula runs as a daemon service on a master node, so the VMM works as a remote user. Users submit VMs on physical machines using different kinds of hypervisors, such as Xen (www.xen.org), which enables the running of several operating systems on the same host concurrently. The VMM also manages VM deployment on grids and IaaS providers. The InterGrid supports Amazon EC2. The connector is a wrapper for the command-line tool Amazon provides. The VM manager for Grid'5000 is also a wrapper for its command-line tools. To deploy a VM, the manager needs to use its template.

4.5.3.4 Virtual Machine Templates

A *VM template* is analogous to a computer's configuration and contains a description for a VM with the following static information:

- The number of cores or processors to be assigned to the VM
- The amount of memory the VM requires
- The kernel used to boot the VM's operating system

- The disk image containing the VM's file system
- The price per hour of using a VM

The gateway administrator provides the VM template information when the infrastructure is set up. The administrator can update, add, and delete templates at any time. In addition, each gateway in the InterGrid network must agree on the templates to provide the same configuration on each site. To deploy an instance of a given VM, the VMM generates a descriptor from the template. This descriptor contains the same fields as the template and additional information related to a specific VM instance. Typically the additional information includes:

- The disk image that contains the VM's file system
- The address of the physical machine hosting the VM
- The VM's network configuration
- The required information for deployment on an IaaS provider

Before starting an instance, the scheduler gives the network configuration and the host's address; it then allocates MAC and IP addresses for that instance. The template specifies the disk image field. To deploy several instances of the same VM template in parallel, each instance uses a temporary copy of the disk image. Hence, the descriptor contains the path to the copied disk image. The descriptor's fields are different for deploying a VM on an IaaS provider. Network information is not needed, because Amazon EC2 automatically assigns a public IP to the instances. The IGG works with a repository of VM templates, called the *VM template directory*.

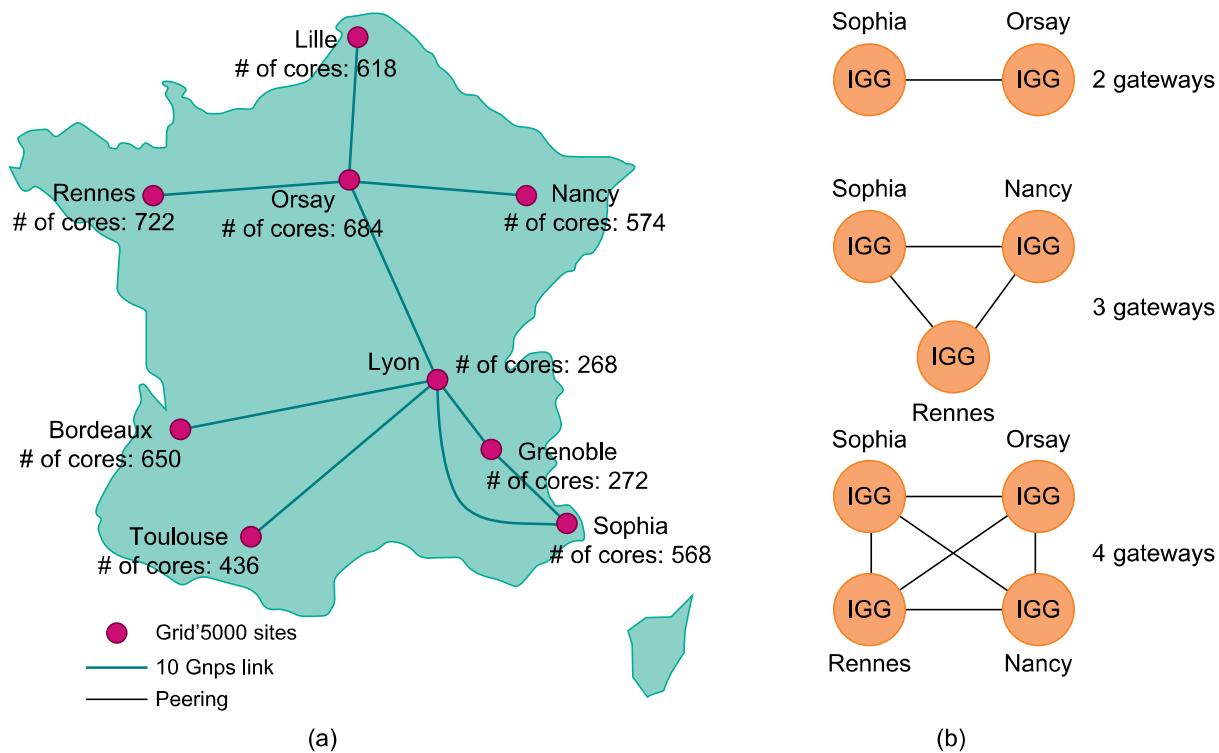
4.5.3.5 Distributed VM Management

Figure 4.30 illustrates the interactions between InterGrid's components. A distributed VM manager makes requests for VMs and queries their status. This manager requests VMs from the gateway on behalf of the user application. The manager obtains the list of requested VMs from the gateway. This list contains a tuple of public IP/private IP addresses for each VM with Secure Shell (SSH) tunnels. Users must specify which VM template they want to use and the number of VM instances needed, the deadline, the wall time, and the address for an alternative gateway.

The local gateway tries to obtain resources from the underlying VIEs. When this is impossible, the local gateway starts a negotiation with any remote gateways to fulfill the request. When a gateway schedules the VMs, it sends the VM access information to the requester gateway. Finally, the manager configures the VM, sets up SSH tunnels, and executes the tasks on the VM. Under the peering policy, each gateway's scheduler uses conservative backfilling to schedule requests. When the scheduler can't start a request immediately using local resources, a redirection algorithm will be initiated.

Example 4.6 Experiments on an InterGrid Test Bed over the Grid'5000

The Melbourne group conducted two experiments to evaluate the InterGrid architecture. The first one evaluates the performance of allocation decisions by measuring how the IGG manages load via peering arrangements. The second considers its effectiveness in deploying a bag-of-tasks application. The experiment was conducted on the French experimental grid platform Grid'5000. Grid'5000 comprises 4,792 processor cores on nine grid sites across France. Each gateway represents one Grid'5000 site, as shown in Figure 4.28.

**FIGURE 4.28**

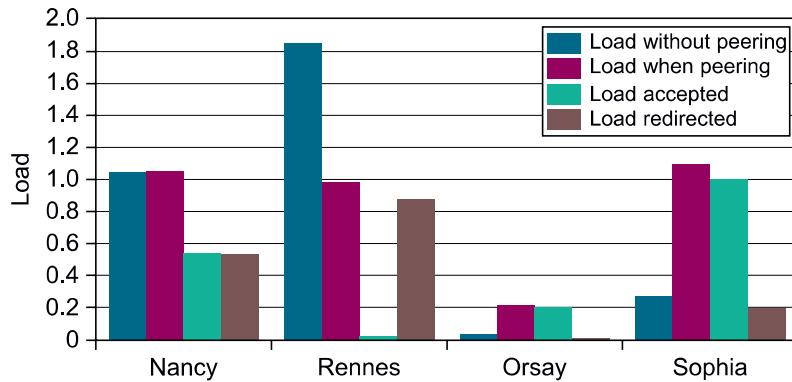
The InterGrid test bed over the French Grid'5000 located in nine cities across France.

(Courtesy of Constanzo, et al. © IEEE [21])

To prevent the gateways from interfering with real Grid'5000 users, emulated VM managers were implemented to instantiate fictitious VMs. The number of emulated hosts is limited by the core number at each site. A balanced workload was configured among the sites. The maximum number of VMs requested does not exceed the number of cores in any site. The load characteristics are shown in Figure 4.29 under a four-gateway scenario. The teal bars indicate each grid site's load. The magenta bars show the load when gateways redirect requests to one another. The green bars correspond to the amount of load each gateway accepts from other gateways. The brown bars represent the amount of load that is redirected. The results show that the loading policy can balance the load across the nine sites. Rennes, a site with a heavy load, benefits from peering with other gateways as the gateway redirects a great share of its load to other sites.

4.5.4 Global Exchange of Cloud Resources

In order to support a large number of application service consumers from around the world, cloud infrastructure providers (i.e., IaaS providers) have established data centers in multiple geographical locations to provide redundancy and ensure reliability in case of site failures. For example, Amazon

**FIGURE 4.29**

Cloud loading results at four gateways at resource sites in the Grid'5000 system.

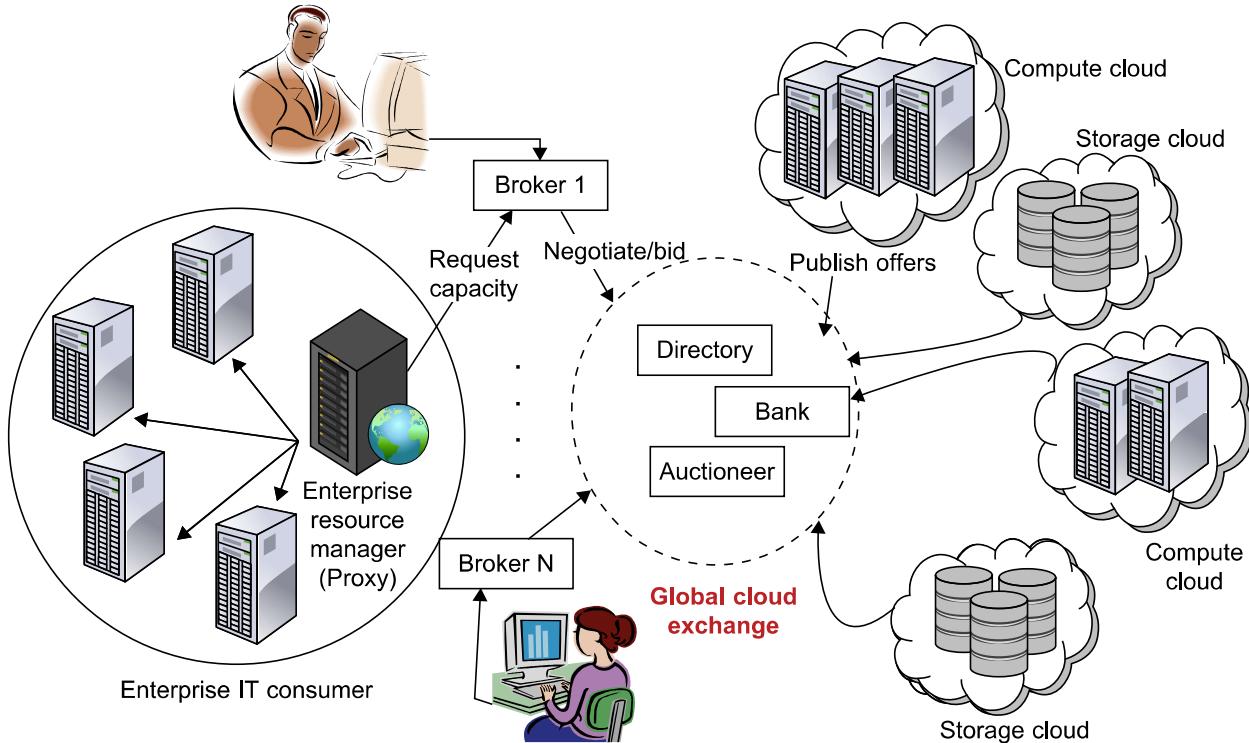
(Courtesy of Constanzo, et al. © IEEE [21])

has data centers in the United States (e.g., one on the East Coast and another on the West Coast) and Europe. However, currently Amazon expects its cloud customers (i.e., SaaS providers) to express a preference regarding where they want their application services to be hosted. Amazon does not provide seamless/automatic mechanisms for scaling its hosted services across multiple geographically distributed data centers.

This approach has many shortcomings. First, it is difficult for cloud customers to determine in advance the best location for hosting their services as they may not know the origin of consumers of their services. Second, SaaS providers may not be able to meet the QoS expectations of their service consumers originating from multiple geographical locations. This necessitates building mechanisms for seamless federation of data centers of a cloud provider or providers supporting dynamic scaling of applications across multiple domains in order to meet QoS targets of cloud customers. Figure 4.30 shows the high-level components of the Melbourne group's proposed InterCloud architecture.

In addition, no single cloud infrastructure provider will be able to establish its data centers at all possible locations throughout the world. As a result, cloud application service (SaaS) providers will have difficulty in meeting QoS expectations for all their consumers. Hence, they would like to make use of services of multiple cloud infrastructure service providers who can provide better support for their specific consumer needs. This kind of requirement often arises in enterprises with global operations and applications such as Internet services, media hosting, and Web 2.0 applications. This necessitates federation of cloud infrastructure service providers for seamless provisioning of services across different cloud providers. To realize this, the Cloudbus Project at the University of Melbourne has proposed InterCloud architecture [12] supporting brokering and exchange of cloud resources for scaling applications across multiple clouds.

By realizing InterCloud architectural principles in mechanisms in their offering, cloud providers will be able to dynamically expand or resize their provisioning capability based on sudden spikes in workload demands by leasing available computational and storage capabilities from other cloud

**FIGURE 4.30**

Inter-cloud exchange of cloud resources through brokering.

(Courtesy of R. Buyya, et al., University of Melbourne [12])

service providers; operate as part of a market-driven resource leasing federation, where application service providers such as [Salesforce.com](#) host their services based on negotiated SLA contracts driven by competitive market prices; and deliver on-demand, reliable, cost-effective, and QoS-aware services based on virtualization technologies while ensuring high QoS standards and minimizing service costs. They need to be able to utilize market-based utility models as the basis for provisioning of virtualized software services and federated hardware infrastructure among users with heterogeneous applications.

They consist of client brokering and coordinator services that support utility-driven federation of clouds: application scheduling, resource allocation, and migration of workloads. The architecture cohesively couples the administratively and topologically distributed storage and compute capabilities of clouds as part of a single resource leasing abstraction. The system will ease the cross-domain capability integration for on-demand, flexible, energy-efficient, and reliable access to the infrastructure based on virtualization technology [6,75].

The *Cloud Exchange (CEx)* acts as a market maker for bringing together service producers and consumers. It aggregates the infrastructure demands from application brokers and evaluates them against the available supply currently published by the cloud coordinators. It supports trading of cloud services based on competitive economic models such as commodity markets and auctions. CEx allows participants to locate providers and consumers with fitting offers. Such markets enable

services to be commoditized, and thus will pave the way for creation of dynamic market infrastructure for trading based on SLAs. An SLA specifies the details of the service to be provided in terms of metrics agreed upon by all parties, and incentives and penalties for meeting and violating the expectations, respectively. The availability of a banking system within the market ensures that financial transactions pertaining to SLAs between participants are carried out in a secure and dependable environment.

4.6 CLOUD SECURITY AND TRUST MANAGEMENT

Lacking trust between service providers and cloud users has hindered the universal acceptance of cloud computing as a service on demand. In the past, trust models have been developed to protect mainly e-commerce and online shopping provided by eBay and Amazon. For web and cloud services, trust and security become even more demanding, because leaving user applications completely to the cloud providers has faced strong resistance by most PC and server users. Cloud platforms become worrisome to some users for lack of privacy protection, security assurance, and copyright protection. Trust is a social problem, not a pure technical issue. However, the social problem can be solved with a technical approach.

Common sense dictates that technology can enhance trust, justice, reputation, credit, and assurance in Internet applications. As a virtual environment, the cloud poses new security threats that are more difficult to contain than traditional client and server configurations. To solve these trust problems, a new data-protection model is presented in this section. In many cases, one can extend the trust models for P2P networks and grid systems to protect clouds and data centers.

4.6.1 Cloud Security Defense Strategies

A healthy cloud ecosystem is desired to free users from abuses, violence, cheating, hacking, viruses, rumors, pornography, spam, and privacy and copyright violations. The security demands of three cloud service models, IaaS, PaaS, and SaaS, are described in this section. These security models are based on various SLAs between providers and users.

4.6.1.1 Basic Cloud Security

Three basic cloud security enforcements are expected. First, facility security in data centers demands on-site security year round. Biometric readers, CCTV (close-circuit TV), motion detection, and man traps are often deployed. Also, network security demands fault-tolerant external firewalls, intrusion detection systems (IDSe), and third-party vulnerability assessment. Finally, platform security demands SSL and data decryption, strict password policies, and system trust certification. [Figure 4.31](#) shows the mapping of cloud models, where special security measures are deployed at various cloud operating levels.

Servers in the cloud can be physical machines or VMs. User interfaces are applied to request services. The provisioning tool carves out the systems from the cloud to satisfy the requested service. A security-aware cloud architecture demands security enforcement. Malware-based attacks such as network worms, viruses, and DDoS attacks exploit system vulnerabilities. These attacks compromise system functionality or provide intruders unauthorized access to critical information.