Chapter 1
Tools and Technologies for Building Clouds

Hai Jin, Shadi Ibrahim, Tim Bell, Li Qi, Haijun Cao, Song Wu, and Xuanhua Shi

Abstract With cloud computing growing in popularity, tools and technologies are emerging to build, access, manage, and maintain the clouds. These tools need to manage the huge number of operations within a cloud transparently and without service interruptions. Cloud computing promises lower costs, faster implementation, and more flexibility using mixtures of technologies, and the associated tools are critical for achieving this.

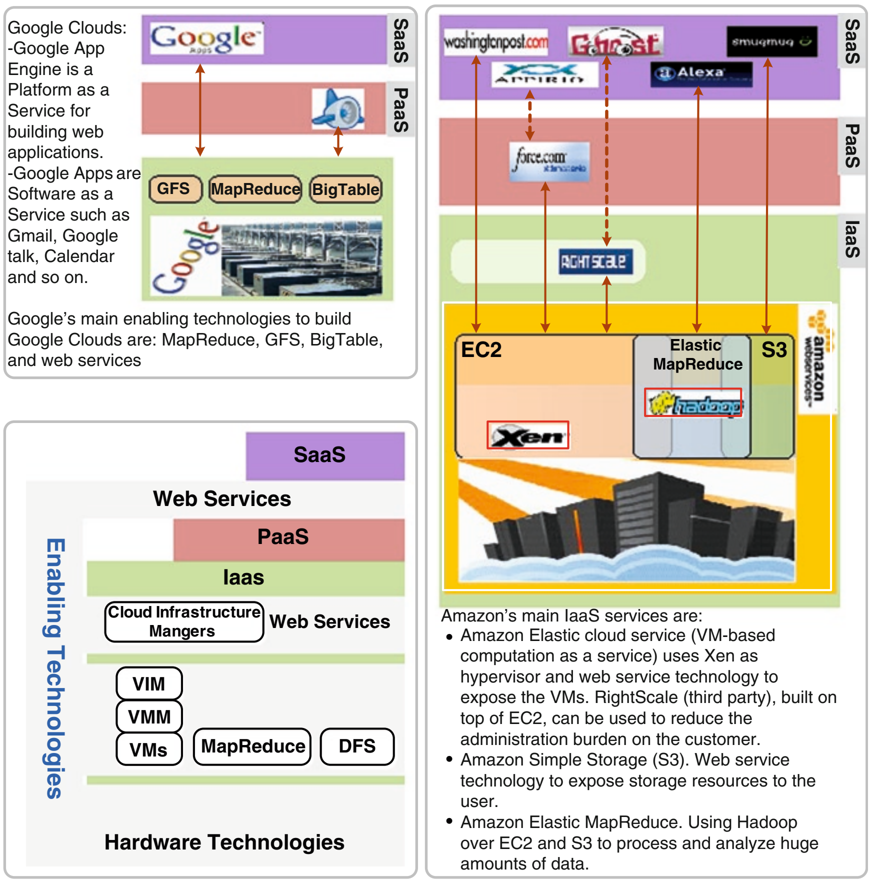
In this chapter, we survey several state-of-the-art techniques for building clouds, starting with virtualization technology. We briefly introduce virtual machines (VMs) and their main features. Then, we introduce the main tools to manage VMs (hypervisors and virtual infrastructure managers) as well as the major technologies used to manage VMs in a public cloud. We then present *MapReduce*, a powerful model that makes it easier to write programs that take advantage of the power of cloud computing. We conclude by examining four web services tools and technologies that are built for cloud computing.

1.1 Introduction

Computing is being transformed by a new model, *cloud computing*. In this model, data and computation are operated somewhere in a "cloud," which is some collection of data centers owned and maintained by a third party.

Cloud computing refers to the hardware, systems software, and applications delivered as services over the Internet. When a cloud is made available in a pay-as-you-go manner to the general public, we call it a *Public Cloud*. The term *Private Cloud* is used when the cloud infrastructure is operated solely for a business or an organization. A composition of the two types (private and public) is called a *Hybrid*

H. Jin (✉)
Services Computing Technology and System Lab, Cluster and Grid Computing Lab,
Huazhong University of Science and Technology, 430074, Wuhan, China
e-mail: hjin@hust.edu.cn



Google Clouds:
-Google App Engine is a Platform as a Service for building web applications.
-Google Apps are Software as a Service such as Gmail, Google talk, Calendar and so on.

Google's main enabling technologies to build Google Clouds are: MapReduce, GFS, BigTable, and web services

Amazon's main IaaS services are:

- Amazon Elastic cloud service (VM-based computation as a service) uses Xen as hypervisor and web service technology to expose the VMs. RightScale (third party), built on top of EC2, can be used to reduce the administration burden on the customer.
- Amazon Simple Storage (S3). Web service technology to expose storage resources to the user.
- Amazon Elastic MapReduce. Using Hadoop over EC2 and S3 to process and analyze huge amounts of data.

Fig. 1.1 Cloud services and enabling technologies, using Amazon and Google systems as examples

Cloud, where a private cloud is able to maintain high service availability by scaling up their system with externally provisioned resources from a public cloud when there are rapid workload fluctuations or hardware failures.

In general, cloud providers fall into three categories (shown in Fig. 1.1):

- Infrastructure as a Service (IaaS): offering web-based access to storage and computing power. The consumer does not need to manage or control the underlying cloud infrastructure but has control over the operating systems, storage, and deployed applications.
- Platform as a Service (PaaS): giving developers the tools to build and host web applications (e.g., APPRIO [1], a software as a service provider, is built using the Force.com [2] platform while the infrastructure is provided by the Amazon Web Service [3]).

- Software as a Service (SaaS): applications that are accessible from various client devices through a thin client interface such as a web browser.

The shift toward cloud computing is driven by many factors including ubiquity of access (all you need is a browser), ease of management (no need for user experience improvements as no configuration or backup is needed), and less investment (affordable enterprise solution deployed on a pay-per-use basis for the hardware, with systems software provided by the cloud providers) [4]. Furthermore, cloud computing offers many advantages to vendors, such as easily managed infrastructure because the data center has homogeneous hardware and system software. Moreover, they are under the control of a single, knowledgeable entity.

1.1.1 Cloud Services and Enabling Technologies

For the purposes of this chapter, we define cloud computing as data centers plus a layer of system software services designed to support the creation and scalable deployment of application services. Our goal here is to examine the tools and technologies used to build these clouds.

The data center hardware consists of thousands of individual computing nodes with their corresponding networking and storage subsystems, power distribution and conditioning equipment, and extensive cooling systems. Such data centers currently power the services offered by companies such as Google, Amazon, Yahoo, and Microsoft's online services division.

Cloud services (remote data and computation) are exposed as simple and user-friendly **web services**. For example, Microsoft's ADO.NET (originally called *Astoria*) [5] provides the tools to expose any data object from a collection, stored in a database or other form, as a URI to an encoded form using a standard such as JSON or ATOM representation, and Google's AppEngine [6] provides a way to deploy a remote Python script that becomes a web service that can access data in their *BigTable* database system.

To deliver highly available and flexible services (i.e., computation as a service), and owing to the maturity of *virtualization technology*, *Virtual Machines* (VMs) are used as a standard for object deployment in the cloud. VMs decouple the computing infrastructure from the physical infrastructure. In addition, VMs allow the customization of the platform to suit the needs of the end-user. For example, in the Amazon Elastic Compute Cloud (EC2) [7], the customer selects his/her preferred VM image (virtual appliance) from a list of various versions of Linux and Windows servers configured with different web servers and databases. Alternatively, they can customize a system to best meet their needs and deploy the new application in the VM. Amazon provides a basic set of web services that can be used to deploy the VM, create an instance, and secure it. Multiple instances can be created to support demand as needed, although this requires more system administration and management. Thus, some organizations have developed *virtual infrastructure* tools to manage and monitor VMs in a pool of distributed resources (e.g., Enomaly

[8] and OpenNebula [9]). In addition, third-party application hosting framework service companies, like RightScale [10] and Elastra [11], have emerged to provide higher-level application deployment tools on top of EC2, thereby reducing the administration burden on the customer.

Because of the huge amount of data stored by a cloud, efficient processing and analysis of data has become a challenging issue. The Google *MapReduce* [12] model has proven to be an efficient approach for data-intensive cloud computing (e.g., Google uses its MapReduce framework to process 20 petabytes of data per day). MapReduce has been advocated as a good basis for data center computers in general [13].

1.2 Virtualization Technology

Virtualization is the idea of partitioning or dividing the resources of a single server into multiple segregated VMs. Virtualization technology has been proposed and developed over a relatively long period. The earliest use of VMs was by IBM in 1960, intended to leverage investments in expensive mainframe computers [14]. The idea was to enable multitasking – running multiple applications and processes for different users simultaneously. Robert P. Goldberg described the need for virtual machines in 1974: “Virtual machine systems were originally developed to correct some of the shortcomings of the typical third generation architectures and multiprogramming operating systems – e.g., OS/360” [15]. During the 1980s and 1990s, the prevailing approach to computing was distributed systems, client-server applications, and the inexpensive x86 server [14]. Recently, owing to the rapid growth in IT infrastructure, we have seen the emergence of multicore processors and a wide variety of hardware, operating systems, and software. In this environment, virtualization has had a resurgence of popularity. Virtualization can provide dramatic benefits for a computing system, including increased utilization, energy saving, rapid deployment, improved maintenance capability, isolation, and encapsulation. Moreover, virtualization enables applications to migrate from one server to another while they are still running, without downtime, providing flexible workload management, and high availability during planned maintenance or unplanned events [16–22].

There are numerous reasons that virtualization is effective in practical scenarios, for example [23,24]:

- Server and application consolidation: under virtualization, we can run multiple applications at the same time on the same server, resulting in more efficient utilization of resources.
- Configurability: virtualization allows dynamic configuration and bundling of resources for a wider variety of applications than could be achieved at the hardware level – different applications require different resources (some requiring more storage, others requiring more computing).

- Increased application availability: VM checkpointing and migration allow quick failure recovery from unplanned outages with no interruption in service.
- Improved responsiveness: resource provisioning, monitoring, and maintenance can be automated, and common resources can be cached and reused.

1.2.1 Virtual Machines

A VM is a software implementation of a machine (i.e., a computer) that executes programs like a physical machine [25]. This differs from a *process* VM, which is designed to run a single program, such as the Java Runtime Environment (JRE). A *system* VM provides a complete system platform that supports the execution of a complete operating system (OS).

The VM lifecycle has six phases: create, suspend, resume, save, migrate, and destroy. Multiple VMs can run simultaneously in the same physical node. Each VM can have a different OS, and a *Virtual Machine Monitor* (VMM) is used to control and manage the VMs on a single physical node. A VMM is often referred to as a *hypervisor*. Above this level, *Virtual Infrastructure Managers* (VIMs) are used to manage, deploy, and monitor VMs on a distributed pool of resources (cluster or data center). In addition, *Cloud Infrastructure Managers* (CIMs) are web-based management solutions on the top of IaaS providers (see Fig. 1.2).

1.2.2 Virtualization Platforms

Virtualization technology has been developed to best utilize computing capacity. Server virtualization has been described as follows: “In most cases, server virtualization

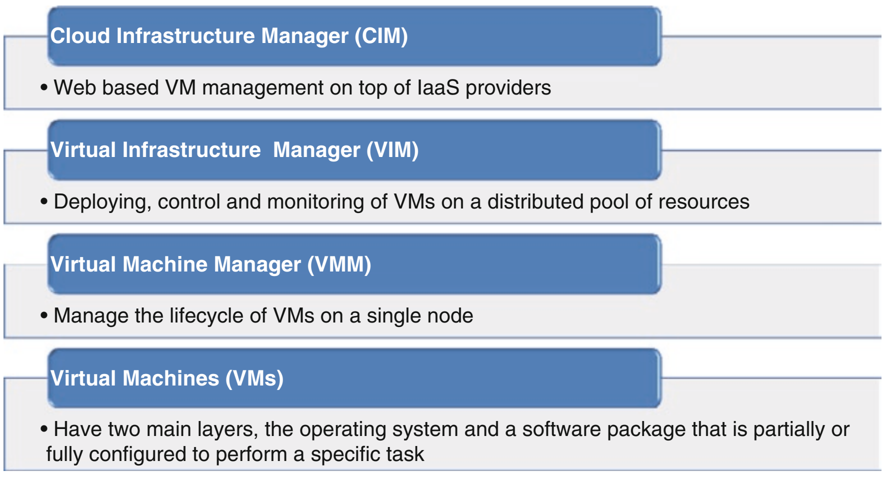

Cloud Infrastructure Manager (CIM)
• Web based VM management on top of IaaS providers
Virtual Infrastructure Manager (VIM)
• Deploying, control and monitoring of VMs on a distributed pool of resources
Virtual Machine Manager (VMM)
• Manage the lifecycle of VMs on a single node
Virtual Machines (VMs)
• Have two main layers, the operating system and a software package that is partially or fully configured to perform a specific task

Fig. 1.2 Different layers of VM management tools and technologies

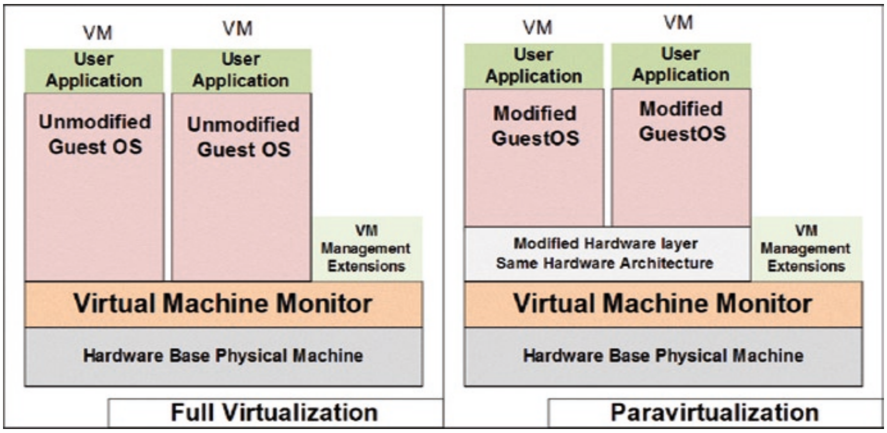


Fig. 1.3 A comparison between full virtualization and paravirtualization VM hypervisors [33]

is accomplished by the use of a hypervisor (VMM) to logically assign and separate physical resources. The hypervisor allows a guest operating system, running on the virtual machine, to function as if it were solely in control of the hardware, unaware that other guests are sharing it. Each guest operating system is protected from the others and is thus unaffected by any instability or configuration issues of the others" [26].

Virtualization methods can be classified into two categories according to whether or not the guest OS kernel needs to be modified, as shown in Fig. 1.3: (1) *full virtualization* (supported by VMware [27], Xen [28], KVM [29], and Microsoft Hyper-V [30], etc.), and (2) *paravirtualization* (currently supported only by Xen). Full virtualization emulates the entire hardware environment by utilizing hardware virtualization support, binary code translation, or binary code rewriting, and thus the guest OS does not need to modify its kernel. Having full virtualization is important for running non-open-source operating system such as Windows, because it is too difficult to modify the Windows kernel without source code. Paravirtualization requires the guest OS kernel to be modified to become aware of the hypervisor. Because it need not emulate the entire hardware environment, paravirtualization can attain better performance than full virtualization.

In paravirtualized architecture, OS-level information about the VM can be passed explicitly from the OS to the VMM, and this is done in practice to some extent [31,32]. Any explicit information supplied by a paravirtualized OS is guaranteed to match what is available inside the OS. However, in some important environments, the explicit approach is less valuable, and because paravirtualization requires OS-level modification, that functionality cannot be deployed in VMMs running beneath legacy or closed-source operating systems anyway.

Table 1.1 compares some of the most relevant commercial and open-source software (OSS) technologies for server virtualization, showing the main trade-off between the product's performances.

Table 1.1 Comparison of some of the most relevant commercial and open-source software (OSS) tools for server virtualization [34]

VMM	Type	Highlights	Guest performance	License
KVM [29]	Full virtualization	Assigns every VM as a regular Linux process	Close to native	Open source
Xen [28]	Paravirtualization	Supports VM migration on fly	Native	Open source
VMware [27]	Full virtualization	Provides a mature product family to manage virtual infrastructure	Close to native	Commercial
Microsoft hyper-V [30]	Full virtualization	Able to trap guest calls	Close to native	Commercial

1.2.3 Virtual Infrastructure Management

A *Virtual Infrastructure Manager* (VIM) is responsible for the efficient management of a virtual infrastructure as a whole, by providing basic functionality for deploying, controlling, and monitoring VMs on a distributed pool of resources. This is done by communicating with their VMMs. The major issues being addressed by the cloud community are:

- Improving the distributed and efficient management of the virtual infrastructure as a whole (i.e., deployment, control, and monitoring)
- Providing self-provisioning of the virtual infrastructure
- Improving the integrity and interoperability of the different virtualization technologies (different hypervisors such as Xen, VMware) as well as the different cloud providers
- Providing administrators with a uniform user-friendly environment that enables access to a wider range of physically distributed facilities improving productivity

Accordingly, many organizations have introduced virtual infrastructure management tools as shown in Table 1.2.

In addition to specific systems such as those listed in the table, open standard organizations such as OGF and DMTF contribute many standards for remote management of cloud computing infrastructures. The scope of the specifications covers all high-level functionality required for the life-cycle management of VMs. Some of these standards have been widely adopted to construct grid and cloud systems, such as the Open Grid Forum *Open Cloud Computing Interface* (OCCI) [40], The *Open Virtualization Format* (OVF) [41], and the virtualization API (*libvirt*) [42].

Table 1.2 Comparison of some of the most relevant commercial and open-source software (OSS) tools for virtual infrastructure management

System name	Brief description	VM hypervisor	Cloud type
Enomaly [8]	A programmable virtual cloud infrastructure for small, medium, and large businesses. Their Elastic Computing Platform (ECP) helps users to design, deploy, and manage virtual applications in the cloud, and also significantly reduces administrative and systems workload. A browser-based dashboard enables IT personnel to simply and efficiently plan deployments, automate VM scaling and load-balancing, and analyze, configure, and optimize cloud capacity.	Xen, KVM	Private and public
Eucalyptus [35]	"Elastic Utility Computing Architecture Linking Your Programs To Useful Systems" – is an open-source software infrastructure for implementing cloud computing on clusters. The current interface to Eucalyptus is compatible with Amazon's EC2, S3, and EBS interfaces, but the infrastructure is designed to support multiple client-side interfaces.	Xen, KVM, VMware	Private and public
Nimbus [36]	Nimbus has been developed in part within the Globus Toolkit 4 framework and provides interfaces to VM management functions based on the WSRF set of protocols. There is also an alternative implementation implementing Amazon EC2 WSDL.	Xen	Private and public
Open Nebula [9]	Orchestrates storage, network, and virtualization technologies to enable the dynamic placement of multitier services (groups of interconnected VMs) on distributed infrastructures, combining both data center resources and remote cloud resources, according to allocation policies.	Xen, KVM, VMWare	Private, hybrid, and public cloud (EC2, Elastic Hosts[37])
Usher [38]	The design philosophy of Usher is to provide an interface whereby users and administrators can request VM operations (e.g., start, stop, migrate, etc.) while delegating administrative tasks for these operations out to smart plug-ins. Usher's implementation allows for arbitrary action to be taken for nearly any event in the system.	Xen	Virtual cluster
VNIX [39]	With VNIX, administrators can deploy various VMs rapidly and easily on computing nodes, and manage them with related configuration from a single easy-to-use management console. In addition, VNIX implements several specialized features, involving easy monitoring, fast deploying, and autoconfiguring.	Xen	Cluster

1.2.4 Cloud Infrastructure Manager

A *Cloud Infrastructure Manager* (CIM) is a web-based solution focused on deploying and managing services (deploying, monitoring, and maintaining the VMs) on top of *Infrastructure as a Service* (IaaS) clouds. Third-party application-hosting framework service companies provide higher-level application deployment tools on top of IaaS. Some of these solutions are listed in Table 1.3.

Table 1.3 Cloud infrastructure management solutions

System name	Brief description	Pricing	Cloud provider	Users
Rightscale [11]	Rightscale is a cloud management environment, cloud-ready server template and best-practice deployment library, adaptable automation engine, and multi-cloud engine.	Starting at US$500, monthly fee	Amazon web services, GoGrid, FlexiScale	G.ho.st, Animoto, and MeDeploy [43]
Elastra [12]	Elastra's main features are: application infrastructure modeling, federated hybrid cloud management, lifecycle orchestration, and deployment management.	Pricing not published	AWS	
Kaavo [44]	IMOD is for Application-Centric Management of virtual resources in the clouds. It provides easy-to-use web interface for deploying, running, and managing complex multiserver n-tier applications in the cloud.	Pricing not published	EC2	The 451 Group and Infoworld [45]
CohesiveFT [46]	PN-Cubed is a commercial solution that enables customer control in a cloud, across multiple clouds, and between private infrastructure and the clouds.	Starting with US$5,000 per year	EC2, Elastic hosts	

1.3 The MapReduce System

Google's MapReduce [12] is a programming model that demonstrates a simpler way to develop data-intensive applications for large distributed systems. It can be leveraged to utilize the resources available through a cloud.

The MapReduce [12] system runs on top of the *Google File System* (GFS) [47], within which data is loaded, partitioned into chunks, and each chunk replicated. Data processing is co-located with data storage: when a file needs to be processed, the job scheduler consults a storage metadata service to get the host node for each chunk, and then schedules a *map* process on that node, so that data locality is exploited efficiently. At the time of writing, because of its remarkable features including simplicity, fault tolerance, and scalability, MapReduce is by far the most powerful realization of data-intensive cloud computing programming. It is often advocated as an easier-to-use, efficient, and reliable replacement for the traditional programming model of moving the data to the computation.

The MapReduce abstraction is inspired by the Map and Reduce functions, which are commonly used in the functional languages such as Lisp [12]. Users express the computation using two functions, map and reduce, which can be carried out on subsets of the data in a highly parallel manner. The runtime system is responsible for parallelizing and fault handling. The steps of the process are as follows. They are illustrated by the widely used "wordcount" example in Fig. 1.4:

- The input is read (typically from a distributed file system) and broken up into key/value pairs. The key identifies the subset of data, and the value will have computation performed on it. (In the example, the keys are each input word read from files A, B, and C, and the values are all a count of one.) The map function maps this data into sets of key/value pairs that can be distributed to different processors.

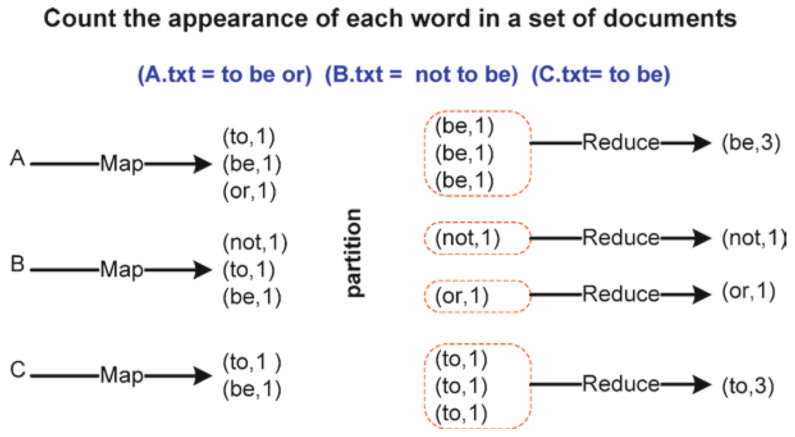


Fig. 1.4 Illustrate the *Map* and *Reduce* functions using the Wordcount Example

- The pairs are partitioned into groups for processing, and are sorted according to their key as they arrive for reduction. (In the example, the pairs are now grouped according to the key.)
- The key/value pairs are reduced, once for each unique key in the sorted list, to produce a combined result. (In this example, this will be the count of each word).

MapReduce has been applied widely in various fields including data- and compute-intensive applications, machine learning, and multicore programming. Moreover, many implementations have been developed in different programming languages for various purposes.

The popular open source implementation of MapReduce, Hadoop [48], was developed primarily by Yahoo, where it processes hundreds of terabytes of data on at least 10,000 cores [49], and is now used by other companies, including Facebook, Amazon, Last.fm, and the *New York Times* [50]. Research groups from enterprises and academia are starting to study the MapReduce model as a better fit for cloud computing, and explore the possibilities of adapting it for more applications.

1.3.1 Hadoop MapReduce Overview

The Hadoop common [48], formerly called the Hadoop core, includes filesystem, RPC (remote procedure call), and serialization libraries, and provides the basic services for building a cloud computing environment with commodity hardware. The two main subprojects are the MapReduce framework and the Hadoop Distributed File System (HDFS).

The HDFS is a distributed file system designed to run on commodity hardware. HDFS is highly fault-tolerant and so can be deployed on low-cost hardware. HDFS provides high throughput access to application data and is suitable for applications that have large data sets. The Hadoop MapReduce framework is highly reliant on its shared file system, and it comes with plug-ins for HDFS, CloudStore [51], and the Amazon Simple Storage Service (S3).

The MapReduce framework consists of a single master JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves (i.e., it queries the HDFS master Namenode about data block locations and assigns each task to the TaskTracker that is closest to where the data to be processed is physically stored), monitoring them, and re-executing any failed tasks. The slaves execute the tasks as directed by the master.

1.4 Web Services

To support cloud computing infrastructure efficiently, and to express business models easily, designers and developers need a group of web services technologies to construct a real, user-friendly, and content-rich set of applications on the top of their clouds. This section introduces four fundamental tools and technologies, which can be employed to construct cloud applications viewed at the infrastructure, architecture, and presentational level. These technologies are: Remote Procedure Call (RPC), Service-Oriented Architecture (SOA), Representational State Transfer (REST), and Mashup.

1.4.1 RPC (Remote Procedure Call)

Reliable and stable communications among cloud resources are fundamental to the infrastructure, and thus are an important consideration. Remote Procedure Call (RPC) has proven to be an efficient mechanism for implementing the client-server model in a distributed computing environment. It was proposed initially by Sun Microsystems as a great advancement in comparison with sockets (e.g., the programmer is not concerned with the underlying communications, since they are embedded inside the RPC). In RPC, the client must know what features the server provides, which are indicated by a service definition, written in IDL (Interface Description Language). An RPC call is a synchronous operation that suspends the calling program until the results of the call are returned. When an RPC is compiled, a stub is included in the compiled code that represents the remote service. When the program runs, it calls the stub, which knows where the operation is and how to reach the service. The stub will send the message through the network to the server. The result of the procedure is returned to the client in the same way.

Many commercial products built over the RPC mechanism have been practically proven as efficient and convenient to construct enterprise applications.

In 2002, Microsoft released the .NET Remoting [52], which was incrementally evolved from DCOM and Active X, to support. NET applications intercommunicating in a loosely coupled environment. Similar to RPC stubs, .NET Remoting initializes the “Channel” objects to proxy the remote calls. To improve the transparency and convenience, the procedure of serialization and marshalling will be completed automatically by .NET runtime. Each .NET Remoting object is identified as a unique URL and safely accessed by clients remotely.

Extending from Java Remote Method Invocation (RMI) [53], Java community presents a complete specification J2EE [54] to standardize the communications among loosely coupled Java components. The enhancements include Enterprise Java Beans (EJB), connectors, servlets, and portlets. The complete J2EE structure of specifications helps designers to easily construct business logic and assists developers in clearly implementing them. Although .Net Remoting and J2EE have been widely adopted by the industry, RPC mechanism is not feasible to construct Cloud applications. One of the problems with RPC is that RPC implementations, as shown in Table 1.4, can be incompatible with each other. To use one of the possible implementations of RPC will result in a high dependence on the particular RPC.

Table 1.4 Web service toolkits comparisons

	Age	Dep.	Transport	Key Tech.	Categories	Implementations
RPC	1974	–	TCP/IP	Stubs, IDL	Infrastructure, IaaS	Java RMI [52], XML RPC, .Net Remoting [53], RPyC, CORBA
SOA	1998	WS-RPC	HTTP,FTP, SMTP	WSDL UDDI SOAP	Architecture level, PaaS	IBM Websphere, Microsoft .Net IIS, Weblogic
REST	2000	HTTP	HTTP,FTP, SMTP	Web-oriented	Architecture level, DaaS	RIP, Rails, Restlet, Jboss RESTEasy, Apache CXF, Symfony
MASHUP	2000 later	REST SOA RSS	HTTP	Web-oriented (Web 2.0)	Application level, SaaS	Google Mashup editor, JackBe, Mozilla Ubiquity

1.4.2 SOA (Service-Oriented Architecture)

The goal of a Service-Oriented Architecture (SOA) [55,56] is to composite together fairly large chunks of functionality to form service-oriented applications, which are almost entirely built from the existing software services. SOA hired a bunch of open standards (1) to wrap the components in different localized runtime environment (e.g., in Java or .NET); (2) to enable different clients including pervasive devices free access; (3) to reuse the existing components to compose more services. This significantly reduces development costs and helps designers and developers to concentrate more on business models and their internal logic.

SOAs use several communication standards based on XML to enhance the interoperability among application systems. As the atomic access point inside an SOA, the web services are formally defined by three kernel standards: Web Service Description Language (WSDL), Simple Object Access Protocol (SOAP), and Universal Description Discovery and Integration (UDDI). Normally, the functional interfaces and parameters of specific services are described using the WSDL. Web services exchange messages are encoded in the SOAP messaging framework and transported over HTTP or other internet protocols (SMTP, FTP, and so forth). A typical web service lifecycle envisions the following scenario: A service provider publishes the WSDL description of their service in a UDDI, a registry that permits Universal Description Discovery and Integration of web services. Subsequently, service requesters can inspect the UDDI and locate/discover web services that are of interest. Using the information provided by the WSDL description, they can directly invoke the corresponding web service. Further, several web services can be composed to achieve more complex functionality. All the invocation procedures are similar to RPC except that the communications and deployments are described in open standards.

Moreover, the open standards organizations such as W3C, OASIS, and DMTF contribute many higher-level standards to help different users construct their reusable, interoperable, and discoverable services and applications. Some of these standards were widely adopted to construct grid and cloud systems, such as Web Services Resources Framework (WSRF) [57], Web Services Security (WS-Security) [58], Web Services Policy (WS-Policy) [59], and so on.

1.4.3 REST (Representative State Transfer)

REST [60] is an architectural style that Roy T. Fielding, now chief scientist at Day Software, first defined in his doctoral thesis. REST stipulates mechanisms for defining and accessing resources in specific distributed systems such as the web. In a REST implementation, resources are addressed via uniform resource identifiers (URIs). That is, a given URI is used to access the representational state of a resource, and also to modify that resource. For example, web URLs can be used to give descriptive information about resources, and consumers then need to know only the URL to read the information. Furthermore, an authorized user can also modify the information if needed.

REST defines three architectural entities as follows [60–62]:

- Data elements: resource identifiers such as URIs and URLs, and resource representations, such as HTML documents, images, and XML documents
- Components: Origin servers, gateways, proxies, and user agents
- Connectors: Clients, servers, and caches

The representational state for resources in an HTTP-based REST system should be accessed using the standard HTTP methods.

A simple breakdown of these methods is as follows: GET is used to transfer the current representational state of a resource from a server to a client; PUT is used to transfer the modified representational state of a resource from the client to the server; POST is used to transfer the new representational state of a resource from the client to the server; and DELETE is used to transfer information needed to change a resource to a deleted representational state.

1.4.4 Mashup

A mashup has been defined in Wikipedia [63] as "a web page or application that combines data or functionality from two or more external sources to create a new service. To be more precise, Mashup technology concentrates on the following tasks

[64]: (1) Deep access to existing enterprise services and data/content repositories; (2) SaaS-style web-based Mashup assembly and use; (3) Assembly models that are truly end-user friendly with very little training required; and last, but certainly not least, (4) a credible management and maintenance plan for IT departments that must support a flood of public end-user built and integrated apps."

Mashup is concerned with the API (application) level. When building Mashups, the developer is always dependent on the providers of the services. As shown in the figure, Mashup requires that the XMLHttpRequest is made to third-party domains. By compositing services and data from SOA, REST, RSS, ATOM, and other RPC-like web servers, a Mashup API can conveniently bind the data with AJAX scripts to deliver a service to end-users.

Some Mashup editors have been implemented to help developers easily construct Web 2.0 and cloud-oriented applications; currently two are available, Google Mashup Editor [65] and Mozilla Ubiquity [66].

1.4.5 Web Services in Practice

All the aforementioned web services tools and technologies have been widely implemented by industry and open-source organizations. Table 1.4 also lists their main attributes in terms of when they were proposed, dependencies, transport mechanism, key technology, categories, and implementations. Understanding these features can help developers to quickly adopt the appropriate technologies and develop their clouds effectively.

1.5 Conclusions

This chapter has presented the main tools and technologies for building and operating clouds. Virtualization technology is foundational to cloud computing because it provides a safe and flexible platform using VMs, VM Monitors, Virtual Infrastructure, and Cloud Infrastructure Managers. Virtualization technology is still developing rapidly, and some of the limitations that currently exist are likely to be addressed as virtualization technology becomes more mature. We have also presented the MapReduce programming model, which is a particularly useful approach for processing huge amounts of data because the computation is close to the data.

Finally, we have reviewed a number of different web services technologies that provide an easy interface for users to configure and access cloud resources.

Cloud computing is a powerful way to provide computing resources, and the tools for creating and maintaining cloud systems and their services are becoming increasingly flexible and easy to use, providing users with easy on-demand access to massive computing power and storage that previously would only have been available to extremely well-resourced organizations.

Acknowledgments This work is supported by National 973 Key Basic Research Program under grant No.2007CB310900, NSFC under grants No.60673174 and No.60973037, Program for New Century Excellent Talents in University under Grant NCET-07-0334, Information Technology Foundation of MoE and Intel under grant MoE-INTEL-09-03, and National High-Tech R&D Plan of China under grant No.2006AA01A115.

References

1. APPRIO Homepage (2009) http://www.appirio.com/
2. Force.com Homepage (2009) http://www.salesforce.com/platform/
3. Amazon Web Services (2009) http://aws.amazon.com/
4. Barroso LA, Urs Hölzle U (2009) The datacenter as a computer: an introduction to the design of warehouse-scale machines. Morgan & Claypool, USA
5. ADO.NET Data Service (formally Astroia) (2009) http://msdn.microsoft.com/en-us/data/bb931106.aspx
6. Google AppEngine (2009) http://code.google.com/appengine/
7. Amazon Elastic Cloud Computing (2009) http://aws.amazon.com/ec2/
8. Enomaly Elastic Computing (2009) http://www.enomaly.com/
9. Open Nubela Homepage (2009) http://www.opennebula.org/
10. Rightscale Homepage (2009) http://www.rightscale.com/
11. Elastra Manage ComplexITy Homepage (2009) http://www.elastra.com/
12. Dean J, Ghemawat S (2008) MapReduce: simplified data processing on large clusters. Commun ACM 51(1):107–113
13. Patterson DA (2008) Technical perspective: the data center is the computer. Commun ACM 51(1):105
14. Vmware (2009) http://www.vmware.com/virtualization/history.html
15. Goldberg RP (1974) Survey of virtual machine research. IEEE Comput Mag 7(6):34–45
16. Waldspurger CA (December 2002) Memory resource management in VMware ESX server. In: Proceedings of the 5th symposium on operating systems design and implementation (OSDI ’02), Boston, MA
17. Fraser K, Hand S, Neugebauer R, Pratt I, Warfield A, Williamson M (2004) Safe hardware access with the Xen virtual machine monitor. In: OASIS ASPLOS 2004 workshop
18. Clark C, Fraser K, Hand S, Hansen JG, Jul E, Limpach C, Pratt L, Warfield A (2005) Live migration of virtual machines. In: Proceedings of the 2nd symposium on networked systems design and implementation (NSDI ’05), Boston, MA
19. Garfinkel T, Pfaff B, Chow J, Rosenblum M, Boneh D (2003) Terra: a virtual machine-based platform for trusted computing. In: Proceedings of the 19th ACM symposium on operating systems principles (SOSP ’03), Bolton Landing (Lake George), New York
20. Bressoud TC, Schneider FB (1995) Hypervisor based fault tolerance. In: Proceedings of the fifteenth ACM symposium on operating systems principles, ACM Press, pp 1–11
21. Petrini F, Kerbyson DJ, Pakin S (2003) The case of the missing supercomputer performance: achieving optimal performance on the 8,192 processors of ASCI Q. In: Proceedings of SC ’03, Washington, DC, USA
22. Koch K (2002) How does ASCI actually complete multi-month 1000-processor milestone simulations? In: Proceedings of the conference on high speed computing
23. Foster I, Zhao Y, Raicu I, Lu S (2008) Cloud computing and grid computing 360-degree compared. In: the Proceedings of the grid computing environments workshop
24. Nanda S, Chiueh T (2005) A survey on virtualization technologies, RPE Report, February. www.ecsl.cs.sunysb.edu/tr/TR179.pdf
25. Virtual Machine (Wikipedia) (2009) http://en.wikipedia.org/wiki/Virtual_machine

26. IBM white paper (2009) Seeding the Clouds: Key Infrastructure Elements for Cloud Computing. ftp://ftp.software.ibm.com/common/ssi/sa/wh/n/oiw03022usen/OIW03022USEN.PDF
27. VMware – Virtual Infrastructure Software (2009) http://www.vmware.com.
28. Xen Homepage (2009) http://www.xen.org/.
29. Kernel-based Virtual Machine (2009) http://kvm.qumranet.com.
30. Microsoft Hyper-V (2009) http://www.microsoft.com/hyper-v-server/en/us/default.aspx
31. Pratt I, Warfield A, Barham P, Neugebauer R (2003) Xen and the art of virtualization. In Proceedings of the 19th ACM symposium on operating systems principles (SOSP '03), Bolton Landing (Lake George), New York
32. Whitaker A, Shaw M, Gribble SD (2002) Scale and performance in the Denali isolation kernel. In: Proceedings of the 5th symposium on operating systems design and implementation (OSDI'02), Boston, MA
33. MSDN Architecture Center, Mapping Applications to the Cloud (2009) http://msdn.microsoft.com/en-us/library/dd430340.aspx
34. Comparison of platform virtual machines (Wikipedia) (2009) http://en.wikipedia.org/wiki/Comparison_of_platform_virtual_machines
35. Eucalyptus system Homepage (2009) http://www.eucalyptus.com/
36. Nimbus Homepage (2009) http://workspace.globus.org/
37. Elastic Hosts Homepage (2009) http://www.elastichosts.com/
38. McNett M, Gupta D, Vahdat A, Voelker GM (2007) Usher: an extensible framework for managing clusters of virtual machines. 21st Large installation system administration conference
39. Shi XH, Tan H, Wu S, Jin H (2008) VNIX: managing virtual machines on clusters, pp 155–162. Japan-China joint workshop on frontier of computer science and technology
40. OGF Open Cloud Computing Interface Working Group (2009) http://www.occi-wg.org/doku.php
41. VMan Initiative (2009) http://www.dmtf.org/initiatives/vman_initiative/
42. libvirt: The virtualization API (2009) http://libvirt.org/
43. RightScale – Testimonials (2009) http://www.rightscale.com/customers/
44. Kaavo Homepage (2009) http://www.kaavo.com/home
45. Kaavo – Testimonials (2009) http://www.kaavo.com/testimonials
46. CohesiveFT Homepage (2009) http://www.cohesiveft.com/
47. Ghemawat S, Gobioff H, Leung ST (2003) The google file system. In: the proceedings of the 19th ACM symposium on operating systems principles, Lake George, New York
48. Hadoop (2009) http://lucene.apache.org/
49. Yahoo! (2009) Yahoo! Developer Network. http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html. Accessed September 2009
50. Hadoop Credits Page (2009) http://hadoop.apache.org/core/credits.html. Accessed September 2009
51. CloudStore (Formely Kosmos File System) (2009) http://kosmosfs.sourceforge.net/
52. .NET Remoting, http://en.wikipedia.org/wiki/.NET_Remoting
53. Java RMI, http://en.wikipedia.org/wiki/Java_RMI
54. J2EE, http://en.wikipedia.org/wiki/J2EE
55. Service Oriented Architecture (Wikipedia) (2009) http://en.wikipedia.org/wiki/Service-oriented_architecture
56. Service-architecture – Service-oriented architecture (SOA) definition (2009) http://www.service-architecture.com/web-services/articles/service-oriented_architecture_soa_definition.html
57. WSRF (2009) http://www.oasis-open.org/committees/wsrf/
58. WS-Security (2009) http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=wss
59. WS-Policy (2009) http://www.w3.org/Submission/WS-Policy/
60. Goth G (2004) Critics say web services need a REST. IEEE Distribut Syst Online 5(12): 1–1
61. Vinoski S (2008) RESTful web services development checklist. IEEE Internet Comput 12(6):94–96
62. Vinoski S ((2007) REST eye for the SOA guy. IEEE Internet Comput 11(1):82–84

63. Mashup (web application hybrid), http://en.wikipedia.org/wiki/Mashup_(web_application_hybrid)
64. Webmashup.com blog, http://www.webmashup.com/blog/category/learn/ (accessed 1 October 2009)
65. Google Mashup Editor (2009) http://en.wikipedia.org/wiki/Google_Mashup_Editor
66. Mozilla Ubiquity (2009) http://ubiquity.mozilla.com/