validating small pieces of work by them iteratively. These agile process models work better for most of the software projects as changes are inevitable and responding to the change is key to the success of a project.

Since agile development was invented in the mid-1990s, it has revolutionized how software is created by emphasizing short development cycles based on fast customer feedback [32]. As the developers are looking for shorter time period, major new releases are delivered on time. Developers using this methodology call the process *continuous improvement*. But for much of its history, agile development was missing a crucial component: a development platform that supports the rapid development cycles that make the methodology work. In traditional software environments, new software distribution is an ordeal that requires patches, reinstallation, and help from the support team. In such an environment, months or even years are needed to get a new distribution into the hands of users. Incorporating their feedback into the next release then requires comparable time.

6.9.4 Advantages of Agile Model

Agile software process offers the following advantages compared to traditional software development models [33]:

1. *Faster time to market*: Since the software is developed using lesser time in the agile process model, it reduces the time an organization takes to launch the product into the market.
2. *Quick ROI*: Since an organization is able launch the product in lesser time, it generates quick ROI.
3. *Shorter release cycles*: Agile process ensures that the software product is released in shorter cycles compared to traditional software development models.
4. *Better quality*: Since the agile development model ensures the maximum interaction among the stakeholders during the entire process of development, it increases the overall quality of the product.
5. *Better adaptability and responsiveness to business changing requirements*: Since the agile process model is adaptive to incorporate changes in the requirements any time during the development process, it increases the responsiveness to changing requirements of the business.
6. *Early detection of failure/failing projects*: Agile process model involves the maximum interaction among the stakeholders, and the testing phase is not delayed till the entire software development process is complete. This helps in the early detection of failure/failing projects.

6.9.5 How Cloud Meets Agile Process?

The cloud development use case encompasses the flow of defects/requirements through phases of development/builds/tests and back to submission of new requirements or defects by various stakeholders. Automation at any point possible is a key capability, including the ability to *turn on* and *rip down* virtual or physical systems as needed, in a cloud. Continuous integration is a key concept to agile practices. It is based on the philosophy of why wait until the end of the project to see if all pieces of the system will work? Every few hours the system should be fully integrated, but tested with all the latest changes, so the adjustments can be made [34].

It is here that cloud computing makes a substantial difference [32]. Cloud computing eliminates the cumbersome distribution requirements that can bring agile development to a crawl. There are no patches to distribute and no reinstallations needed. With cloud computing, new distributions are installed on hosted servers and made available to users immediately. As a result, it is possible that the application being run today was modified just the night before. One of the best examples of bringing together agile development and cloud computing is the experience of Salesforce.com where, in late 2006, the R&D team moved to agile development.

6.9.5.1 Six Ways the Cloud Enhances Agile Software Development

Cloud computing and virtualization allow the creation of VMs and use of cloud-based services for project management, issue management, and software builds with automated testing. This, in turn, encourages agile development in six key ways. Cloud computing and virtualization make it easy for agile development teams to seamlessly combine multiple development, test, and production environments with other cloud services. Here are six important ways in which cloud computing and virtualization enhance agile software development [35]:

1. *Cloud computing provides an unlimited number of testing and staging servers*: When agile development is used without virtualization or clouds, development teams are limited to one physical server per development, staging, and production server need. However, when VMs or cloud instances are used, development teams have practically an unlimited number of servers available to them. They do not need to wait for physical servers to become free to begin or continue their work.
2. *It turns agile development into a truly parallel activity*: Even in agile development, a developer may experience delays in provisioning server instances and in installing necessary underlying platforms such as database software. Agile development teams can provision the servers they need quickly themselves, rather than wait for IT operations to do it for them.

3. *It encourages innovation and experimentation*: Being able to spawn as many instances as needed enables agile development groups to innovate. If a feature or a story looks interesting, a team can spawn a development instance quickly to code it and test it out. There is no need to wait for the next build or release, as is the case when a limited number of physical servers are available. When adding cloud computing to agile development, builds are faster and less painful, which encourages experimentation.
4. *It enhances continuous integration and delivery*: Having a large number of VMs available to the agile development group in its own cloud or on the public cloud greatly enhances the speed of continuous integration and delivery.
5. *It makes more development platforms and external services available*: Agile development groups may need to use a variety of project management, issue management, and, if continuous integration is used, automated testing environments. A number of these services are available as Software as a Service (SaaS) offerings in the cloud:
   a. Agile development can use a combination of virtualization, private clouds, and the public cloud at the IaaS level. Such offerings include Amazon Web Services, GoGrid, OpSource, and RackSpace Cloud.
   b. Then comes the use of PaaS instances such as the Oracle Database Cloud Service, the Google App Engine, and the Salesforce.com's platform (force.com), all of which include databases and language environments as services.
   c. Finally, there are a number of SaaS services that specifically assist agile development, including Saleforce.com, the Basecamp project management portal, and TestFlight, which provides hosted testing automation for Apple iOS devices.
6. *It eases code branching and merging*: In code refactoring efforts, current releases may need to be enhanced with minor enhancements and used in production, all while a major redesign of code is going on. Code branching is necessary in these cases. Code branching and merging involve juggling many versions of development and staging builds. With virtualization and cloud computing, buying or renting additional physical servers for these purposes can be avoided.

6.9.5.2 Case Study of Agile Development

Meanwhile, Salesforce.com's R&D leverages cloud computing to vastly speed up release cycles [32]. The company's cloud infrastructure helps it maintain a single, unified code base that geographically distributed development teams can use. Those teams are successfully combining agile development

and continuous integration/delivery with cloud computing. In reference [32], Salesforce.com finds that agile process model works better on cloud computing platform. Before the introduction of cloud computing, there was a gap or time interval between the releasing of software and getting feedback from the customer and now new software release can be uploaded to the server and used by the customer simultaneously. So, the agile development model can complement the benefits of software services hosted on the Internet. In the rapidly varying computing environment with web services and cloud platform, software design and development also involve various platforms, distributed web services, and geographically distributed enterprises [36].

Salesforce.com's R&D organization has benefitted in several ways from its transition to agile development [32]:

- Increased delivery rate and created a process that makes customers and R&D happy
- Increased time to market of major releases by 61%
- Achieved a Net Promoter Score of 94%, a good indicator of customer satisfaction
- Convinced 90% of the R&D team to recommend the methodology to colleagues inside and outside the company
- Increased productivity across the organization by 38%, as measured by the number of features produced per developer (a side benefit not anticipated as part of the original goals)

6.10 Programming Models

Programming models for cloud computing have become a research focus recently. Cloud computing promises to provide on-demand and flexible IT services, which goes beyond traditional programming models and calls for new ones [37]. Cloud platforms allow programmers to write applications that run in the cloud, or use services from the cloud, or both while abstracting the essence of scalability and distributed processing [38]. With the emergence of cloud as a nascent architecture, abstractions that support emerging programming models are needed. In recent years, cloud computing has led to the design and development of diverse programming models for massive data processing and computation-intensive applications.

Specifically, a programming model is an abstraction of the underlying computer system that allows for the expression of both algorithms and data structures [39]. In comparison, languages and APIs provide implementation of the abstractions and allow algorithms and data structures to be put into practice. A programming model exists independently of the choice of both the

programming language and the supporting APIs. Programming models are typically focused on achieving increased developer productivity, performance, and portability to other system designs. The rapidly changing nature of processor architectures and the complexity of designing a platform provide significant challenges for these goals. Several other factors are likely to impact the design of future programming models. In particular, the representation and management of increasing levels of parallelism, concurrency, and memory hierarchies, combined with the ability to maintain a progressive level of interoperability with today's applications, are of significant concern. Furthermore, the successful implementation of a programming model is dependent on exposed features of the runtime software layers and features of the OS [39].

Over the years, many organizations have built large-scale systems to meet the increasing demands of high storage and processing requirements of compute- and data-intensive applications [38]. With the popularity and demands on DCs, it is a challenge to provide a proper programming model that is able to support convenient access to large-scale data for performing computations while hiding all low-level details of physical environments. Cloud programming is about knowing what and how to program on cloud platforms. Cloud platforms provide the basic local functions that an application program requires. These can include an underlying OS and local support such as deployment, management, and monitoring.

6.10.1 Programming Models in Cloud

There are different programming models that are used for solving various compute- or data-intensive problems in cloud. The model to be selected depends on the nature of the problem and also on the QoS expected from the cloud environment. Some of the cloud programming models are discussed in the following subsections.

6.10.1.1 BSP Model

With the advantages on predictable performance, easy programming, and deadlock avoidance, the bulk synchronous parallel (BSP) model has been widely applied in parallel databases, search engines, and scientific computing. The BSP model can be adapted into the cloud environment [37]. The scheduling of computing tasks and the allocation of cloud resources are integrated into the BSP model. Recently, research on cloud computing programming models has made some significant progress, such as Google's MapReduce [40] and Microsoft's Dryad [41]. The BSP model is originally proposed by Harvard's Valiant. Its initial aim is to bridge parallel computation software and architecture. It offers the following advantages: firstly, its performance can be predicted; secondly, no deadlock occurs during message passing; and thirdly, it is easy to program. The BSP model can be used not only for data-intensive applications but also for computation-intensive and I/O-intensive applications.

6.10.1.2 MapReduce Model

Recently, many large-scale computer systems are built in order to meet the high storage and processing demands of compute- and data-intensive applications. MapReduce is one of the most popular programming models designed to support the development of such applications [42]. It was initially created by Google for simplifying the development of large-scale web search applications in DCs and has been proposed to form the basis of a *data center computer*. With the increasing popularity of DCs, it is a challenge to provide a proper programming model that is able to support convenient access to the large-scale data for performing computations while hiding all low-level details of physical environments. Among all the candidates, MapReduce is one of the most popular programming models designed for this purpose.

MapReduce is triggered by the map and reduce operations in functional languages, such as Lisp. This model abstracts computation problems through two functions: map and reduce. All problems formulated in this way can be parallelized automatically. Essentially, the MapReduce model allows users to write map/reduce components with functional-style code. These components are then composed as a dataflow graph to explicitly specify their parallelism. Finally, the MapReduce runtime system schedules these components to distributed resources for execution while handling many tough problems: parallelization, network communication, and fault tolerance.

A map function takes a key/value pair as input and produces a list of key/value pairs as output. A reduce function takes a key and associated value list as input and generates a list of new values as output. A MapReduce application is executed in a parallel manner through two phases. In the first phase, all map operations can be executed independently from each other. In the second phase, each reduce operation may depend on the outputs generated by any number of map operations. All reduce operations can also be executed independently similar to map operations.

The task execution is carried out in four stages: map, sort, merge, and reduce. The map phase is fed with a set of key/value pairs. For each pair, the mapper module generates a result. The sort and merge phases group the data to produce an array, in which each element is a group of values for each key. The reduce phase works on this data and applies the reduce function on it. The hash functions used in the map and reduce functions are user defined and varies with the application of the model. The overall computation is depicted in Figure 6.5.

MapReduce has emerged as an important data-parallel programming model for data-intensive computing [38]. However, most of the implementations of MapReduce are tightly coupled with the infrastructure. There have been programming models proposed that provide a high-level programming interface, thereby providing the ability to create distributed applications in an infrastructure-independent way. Simple API for Grid Applications

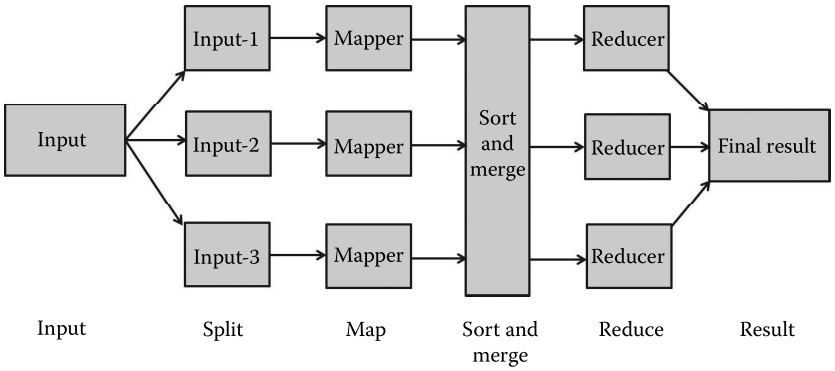

FIGURE 6.5
Computation of MapReduce. (From Jayaraj, A. et al., *Programming Models for Clouds*.)

(SAGA) [43] and Transformer [44] are examples of such models, which try to implement parallel models like MapReduce [42] and All-Pairs [45], taking considerable burden off the application developer.

6.10.1.3 SAGA

Although MapReduce has emerged as an important data-parallel programming model for data-intensive computing, most, if not all, implementations of MapReduce are tightly coupled to a specific infrastructure. SAGA is a high-level programming interface that provides the ability to create distributed applications in an infrastructure-independent way [38]. SAGA supports different programming models and concentrates on the interoperability on grid and cloud infrastructures. SAGA supports job submission across different distributed platforms, file access/transfer, and logical file, as well as checkpoint recovery and service discovery. SAGA API is written in C++ and supports other languages like Python, C, and Java. The runtime environment decision making is given support by the engine that loads relevant adaptors, as shown in Figure 6.6.

6.10.1.4 Transformer

Even though there are existing programming models based on C++ and Java in the industrial market, they suffer from certain shortcomings. First, the programmers have to master the bulky and complex APIs in order to use the model. Secondly, most programming models are designed for specific programming abstractions and created to address one particular kind of problem. There is an absence of a universal distributed software framework for processing massive datasets. To address the aforementioned shortcomings, a new framework called Transformer [38] is used, which supports diverse programming models and also is not problem specific.

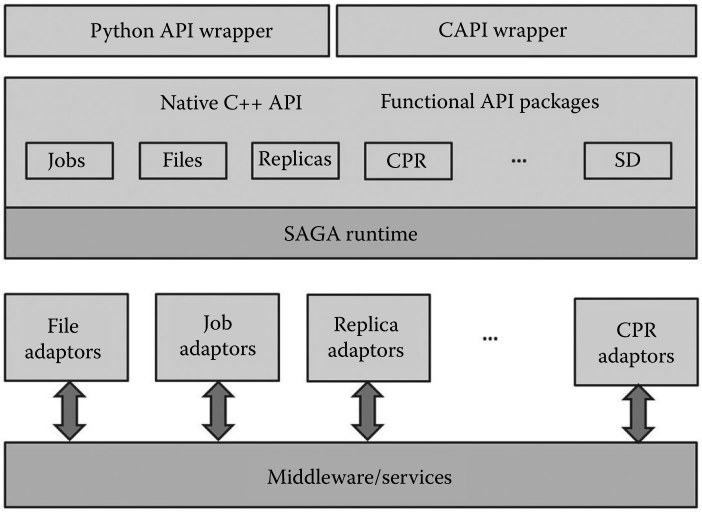

FIGURE 6.6
SAGA model. (From Jayaraj, A. et al., *Programming Models for Clouds*.)

Transformer is based on two concise operations: send and receive. Using the Transformer model, various models such as MapReduce [39], Dryad, and All-Pairs [40] can be built. The architecture of the Transformer model is divided into two layers: common runtime and model-specific systems, as shown in Figure 6.7. This is done to reduce coupling. Runtime system handles the tasks-like flow of data between machines and executes the tasks on different systems making use of send and receive functions from runtime API. Model-specific layer deals with particular model tasks like mapping, data partitioning, and data dependencies.

Transformer has a master/slave architecture. Every node has two communication components: a message sender and a message receiver. The

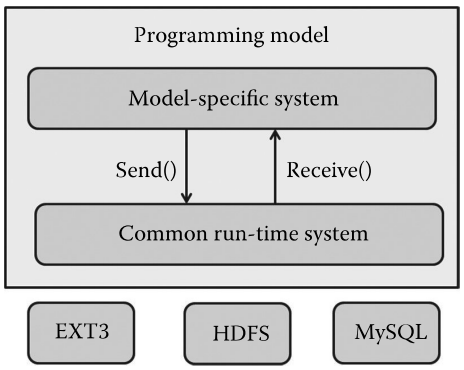

FIGURE 6.7
Transformer architecture. (From Jayaraj, A. et al., *Programming Models for Clouds*.)

master node issues commands for task execution on the slave nodes. The slave nodes return the status of execution when it is over. The fault-tolerance strategy is agile in nature. Failure is detected by the runtime system whereas fault recovery is handled by the model-specific layer. This involves rerunning the tasks or resending data. Transformer system is coded in Python. Communication between nodes is done using message-passing mechanism opposed to semaphores and conditions in threaded approach. Since the frequency of communication is high, asynchronous network programming is adopted, and moreover the message is serialized before sending it. Using the Transformer model, all three known parallel programming models, namely MapReduce, Dryad, and All-Pairs, are implemented.

6.10.1.5 Grid Batch Framework

Recently, an alternative to parallel computational models has been suggested that enables users to partition their data in a simplified manner while having the highest possible efficiency. The Grid Batch system has two fundamental data types [38]: table and indexed table. A table is a set of rows that are independent of each other. An indexed table has all the properties of a table in addition to having an index associated with each record.

The two major software components of the Grid Batch system are the Distributed File System (DFS) and the Job Scheduler. The DFS is responsible for storing and managing the files across all the nodes in the system. A file is broken down into many pieces and each of these pieces is stored on a separate node. The Job Scheduler constitutes of a master node and associated slave nodes. A job is broken down into many smaller tasks by the master node, and each of these tasks is distributed among the slave nodes. The basic map and reduce operators in the MapReduce system are extended in the Grid Batch model. These are map operator, distribute operator, join operator, Cartesian operator, recurse operator, and neighbor operator.

6.11 Pervasive Computing

Pervasive computing is a combination of technologies, such as Internet capabilities, voice recognition, networking, artificial intelligence, and wireless computing, used to make computing anywhere possible. Pervasive computing devices make day-to-day computing activities extremely easy to perform. The technology is moving beyond the PC to everyday devices with embedded technology and connectivity. Pervasive computing is also called ubiquitous computing, in which almost any device or material such as clothing, tools, appliances, vehicles, homes, human body, or even the coffee mug can be imbedded with chips to connect that object to an infinite network

of other devices. The goal of pervasive computing, which combines current network technologies with wireless computing, voice recognition, Internet capability, and artificial intelligence, is to create an environment where the connectivity of devices is achieved in such a way that the connectivity is unobtrusive and always available. Pervasive computing also has a number of prospective applications, which range from home care and health, to geographical tracking and intelligent transport systems.

The words pervasive and ubiquitous mean *existing everywhere*. Pervasive computing devices are completely connected and constantly available. Pervasive computing relies on the convergence of wireless technologies, advanced electronics, and the Internet. The goal of researchers working in pervasive computing is to create smart products that communicate unobtrusively. The products are connected to the Internet and the data they generate are easily available. An example of a practical application of pervasive computing is the replacement of old electric meters with smart meters. In the past, electric meters had to be manually read by a company representative. Smart meters report usage of electricity in real time over the Internet. They will also notify the power company when there is an outage and also send messages to display units in the home and regulate the water heater.

Hence, in pervasive computing, computing is made to appear everywhere and anywhere [46]. In contrast to desktop computing, pervasive computing can be done using any supporting device, at any location. The underlying technologies to support pervasive computing include Internet, advanced middleware, OS, mobile, sensors, microprocessors, new I/O and user interfaces, networks, mobile protocols, and location-based services. This paradigm is also named with different names like physical computing, the Internet of Things, and things that think, by considering the objects involved in it. In this case, the device used to access applications and information is almost irrelevant as various types of devices or platform can be used to perform the intended operation [47].

6.11.1 How Pervasive Computing Works?

The success of ubiquitous computing rests with the proper integration of various components that talk to each other and thereby behaving as a single connected system. Figure 6.8 shows the architecture of a ubiquitous computing stack [48]. At the bottom of the stack is a *physical* layer. Tiny sensors are attached (carried, worn, or embedded) to people, animals, machines, homes, cars, buildings, campuses, and fields. Sensors capture various bits of information from the immediate surroundings. Beyond the microphone and camera, multiple sensors such as GPS, accelerometer, and compass can be integrated into it.

Above the sensors lies the wireless communication infrastructure, which can be provided by the 802.11 family of networks. Together with mesh networks, such standards ensure the connectivity of sensors and devices.

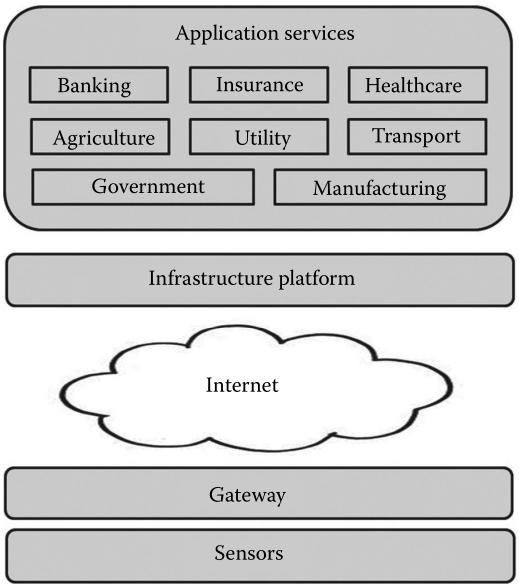

FIGURE 6.8
Pervasive computing stack. (From Perspectives, *TCS Consult. J.*, available at: http://sites.tcs.com/insights/perspectives/enterprise-mobility-ubiquitous-computing-beyond-mobility#.UzPo8fmSzCe.)

Another technology called ZigBee is a low-cost alternative for keeping multiple devices connected, allowing parent devices to wirelessly control child sensors. Near field communication (NFC) is yet another technology standard that leverages RFID and can be used for ubiquitous computing, especially in scenarios where non-battery-operated passive points are concerned. NFC-powered devices can also interact with one another.

The next level includes a range of application services. The data from the sensors and handheld devices are gathered, mined, and analyzed for patterns. The patterns help provide options to smart applications that proactively make changes to environments through smartphones, tablets, notebooks, or any other handheld devices or smart devices. An example could be that of a cardiac patient wearing a tiny monitor connected to a mobile device. An irregular ECG will trigger the mobile to alert the patient's doctor and emergency services.

6.11.2 How Pervasive Computing Helps Cloud Computing?

Nowadays, IT enterprises are adopting cloud computing in order to reduce the total cost involved and also to improve the QoS delivered to the customers. Cloud computing provides the opportunity to access the infrastructure, platform, and the software from the service providers on a

pay-per-use basis. Pervasive computing helps cloud computing by providing the ability to access the cloud resources anytime, anywhere and also through any device. Pervasive computing provides the necessary features such as ubiquitous computing, storage and archiving, social community–based applications, and business as well as nonbusiness applications in order for cloud computing to gain its full potential. Cloud computing is typically a client-server architecture, where the client can be any portable device like a laptop, phone, browser, or any other OS-enabled devices [49]. A main issue with these portable devices is the constraints they present in terms of storage, memory, processing, and battery lifetime. By storing data on the cloud, and interacting with the cloud through secure communication channels, all these constraints can be easily met.

Machine-to-machine (M2M) communication is important in cloud computing [47]. Such a communication scenario spans from the shop floor, to the DC, to the boardroom, as the devices carried along track the user's movements and activities and also interact with the other systems around. For example, an employee is currently in New York and he wants to discuss something with two colleagues. He requests an appointment using his mobile device, and based on his location data and that of his colleagues, and the timing of the meeting, backend systems automatically book him a conference room and set up a video link to a coworker out of town. Based on analytics and the title of the meeting, relevant documents are dropped into a collaboration space. The employee's device records the meeting to an archive and notes who has attended in person. And, this conversation is automatically transcribed, tagged, and forwarded to team members for review.

Wearable devices like Google Glass will also feed into the new workplace. The true power behind these applications is not in the devices themselves but in the analytic systems that back them. The backend systems of the applications combine the data collected from the various types of computing devices such as Google Glass, smartphones, an embedded GPS device in a palette, or a sensor in a car's engine. Such systems process the data and then turn it into useful information that is used for triggering the required actions. Different computing systems performing various activities are deployed with APIs so that the user can build applications that extract information from these multiple systems.

6.12 Operating System

An OS is a collection of softwares that manages the computer hardware resources and other programs in the computing system. It provides common services required by computer programs for their effective execution within the computing environment. The OS is an essential component of the system

software in a computer system as application programs usually require an OS for their interface with the hardware resources and other system programs. For hardware functions such as input and output, and memory allocation, the OS acts as an intermediary between programs and the computer hardware.

6.12.1 Types of Operating Systems

The different variants of OSs are the following:

1. *Network OSs*: A network operating system (NOS) is a computer OS that is designed primarily to support workstations, PCs that are connected on a LAN. An NOS provides features such as printer sharing, common file system and database sharing, application sharing, security mechanisms, and also the ability to manage a network name directory and other housekeeping functions of the network. Novell's NetWare and Microsoft's LAN Manager are examples of NOSs. In addition, some multipurpose OSs, such as Windows NT and Digital's OpenVMS, come with capabilities that enable them to be described as an NOS.
2. *Web OSs*: Web OSs are basically websites that replicate the desktop environment of modern OSs, all inside a web browser. They are installed onto web servers and live on the Internet. Thus, a user can access his virtual desktop from any device, anywhere, that is connected to the net. Web OSs are also called the dynamic computers. In this case, the applications, hard disk, and OSs are all present at the servers from where they are accessed. The web OS service provider manages the application and database accesses of the various users. The user is provided with a graphical user interface similar to the one available on a desktop PC, which can be used to access the data and the applications from the server. Google Chrome OS is an example of a web OS.
3. *Distributed OS*: A distributed OS is a software that is present over a collection of independent, networked, communicating, and physically separate computational nodes. Each individual node holds a specific software that is a subset of the global aggregate OS. Each subset consists of two distinct components of the distributed OS. The first one is a ubiquitous minimal kernel, or microkernel, that directly controls the node's hardware. The second one is a higher-level collection of system management components that coordinate the node's individual and collaborative activities. The microkernel and the management components work together. They support the distributed system's goal of integrating multiple resources and processing functionality into an efficient and stable system. To a user,

a distributed OS works in a manner similar to a single-node, monolithic OS. That is, although it consists of multiple nodes, it appears to the users and applications as a single-node OS.

4. *Embedded systems*: Embedded systems are OSs present in electronic devices used for various purposes in order to make them *smart* and more efficient. Embedded systems present in devices such as routers, for example, typically include a preconfigured web server, DHCP server, and some utilities for its effective networking operation, and they do not allow the installation of new programs in them. Examples of embedded OSs for routers include Cisco Internetwork Operating System (IOS), DD-WRT, and Juniper Junos. An embedded OS can also be found inside an increasing number of consumer gadgets including phones, personal digital assistance (PDA), and digital media player for the successful completion of their intended tasks.

6.12.2 Role of OS in Cloud Computing

In the 1970s, International Business Machines Corporation (IBM) released an OS called VM that allowed mainframe systems to have multiple virtual systems, or VMs on a single physical node [50]. The VM OS materialized shared access of mainframe systems to the next level by allowing multiple distinct computing environments to live in the same physical environment. Most of the basic functions of current virtualization software are inherited from this early VM OS. The virtualization software is now represented with the term *hypervisor*. Hardware virtualization is used to share the resources of the cloud providers effectively with the customers by generating an illusion of dedicated computing, storage, and networking on a computing infrastructure. The concept of virtualization is physically implemented using the hypervisor modules, and the operation and processing of hypervisors are materialized by the OSs. In other words, hypervisor modules are installed on top of OSs, which act as an interface between hardware units and hypervisor packages.

Using virtualization, multiple VMs are generated according to the requirement, and these VMs are made operational by individually installing OSs on each VM. Figure 6.9 shows the virtualization of a single hardware of the CSP to create different VMs, each installed with its own OS. Every VM runs custom OS or guest OS that has its own memory, CPU, and hard drives along with CD-ROMs, keyboards, and networking, despite the fact that all of those resources are shared among the VMs.

In addition, an OS such as Linux supports the necessary standards that enhance portability and interoperability across cloud environments [51]. OS platforms are designed to hide much of the complexity required to

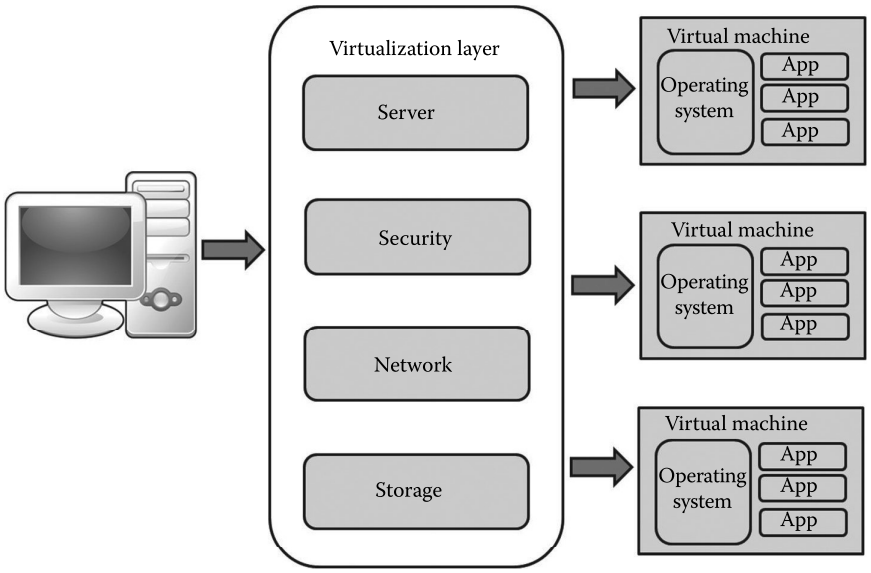

FIGURE 6.9
OS and virtualization. (From Steddum, J., A brief history of cloud computing, available at: http://blog.softlayer.com/tag/mainframe.)

support applications running in complex and federated environments. It needs to work effectively in the background in order to ensure that all the right resources (such as processing power, required memory, and storage) are allocated when needed. In addition, the OS implements the level of security and QoS to ensure that applications are able to access the resources needed to deliver an acceptable level of performance, in an efficient and secure manner.

One of the most important ways to support the underlying complexity of well-managed cloud computing resources is through the OS [51]. One of the most significant requirements for companies adopting cloud computing is the need to adopt a hybrid approach to computing. To do so, most organizations will continue to maintain their traditional DC to support complex mixed workloads. For example, an organization may choose a public cloud environment for development and test workloads, a private cloud for customer-facing web environments that deal with personal information, and a traditional DC for legacy billing and financial workloads. It is considered that hybrid cloud computing environments will be the norm for the future. Therefore, it is more important than ever for the OS to support and federate the various computing deployment models so that they appear to be a single system from a customer experience and a systems management perspective.

6.12.3 Features of Cloud OS

The elements required to create an operationally sophisticated hybrid cloud computing environment include the following [51]:

1. Well-defined interfaces that hide implementation details
2. Core security services
3. The ability to manage virtualization
4. Management of workloads to provide QoS and performance

These features are explained in the following subsections.

6.12.3.1 Well-Defined and Abstracted Interfaces

A cloud OS should provide the APIs that enable data and services interoperability across distributed cloud environments. Mature OSs provide a rich set of services to the applications so that each application does not have to invent important functions such as VM monitoring, scheduling, security, power management, and memory management. In addition, if APIs are built on open standards, it will help organizations avoid vendor lock-in and thereby creating a more flexible environment. For example, linkages will be required to bridge traditional DCs and public or private cloud environments. The flexibility of movement of data or information across these systems demands the OS to provide a secure and consistent foundation to reap the real advantages offered by the cloud computing environments. Also, the OS needs to make sure the right resources are allocated to the requesting applications. This requirement is even more important in hybrid cloud environments. Therefore, any well-designed cloud environment must have well-defined APIs that allow an application or a service to be plugged into the cloud easily. These interfaces need to be based on open standards to protect customers from being locked into one vendor's cloud environment.

6.12.3.2 Support for Security at the Core

Whether an organization is considering a public, private, or hybrid cloud environment, security is the most important foundation to ensure the protection for its assets. Security issues are exacerbated in a cloud environment since it is highly distributed, and also it involves a large variety of internal and external systems added or removed to/from the cloud dynamically. Therefore, the cloud environment has to protect the identity of the users and their information from external threats. To support the needs of the organizations, cloud security requires an integrated management capability within the OS that can track all IT assets in the context of how they are being used. This capability needs to ensure that the security meets an organization's compliance and governance requirements.

Both virtualization and multitenancy supports have to be implemented in a secure manner. As virtualization and multitenancy become the norm in cloud environments, it is critical that security be built-in at the core. When servers are virtualized, it makes the creation of a new image possible with little effort. This expansion of virtual images raises the risk of attack because it increases the possibility that a security flaw in the hypervisor can be exploited by a guest instance. It can expose both existing systems and other partners that interact with those systems to security threats. When security is implemented as a framework within the OS, it improves the overall security of both virtualized and nonvirtualized environments.

6.12.3.3 Managing Virtualized Workloads

Virtualization is fundamental to cloud computing because it breaks the traditional links between the physical server and the application. These virtualized environments are controlled and managed by a hypervisor. In essence, the hypervisor is an OS on the physical hardware and presents the core hardware abstraction and I/O instructions needed by the guests in its environment. Hence, the effective management of the hypervisor and the virtualized environments is critical to the success of cloud computing.

6.12.3.4 Management of Workloads

The cloud environment has to be designed in a manner that protects the individual customer's workloads. Hence, the prerequisite for effective cloud computing is the ability to isolate workloads of users from each other. The cloud OS should make sure that all the required resources are effectively managed in the cloud environment for the individual users.

6.12.4 Cloud OS Requirements

Other than the features of the cloud OS explained in the previous section, the major OS requirements in a cloud environment are given in the following [52]:

1. *The cloud OS must permit autonomous management of its resources*: The cloud OS should expose a consistent and unified interface that conceals whenever possible the fact that individual nodes are involved in its operations, and what those low-level operations are. It should support the autonomous management of the various cloud resources on behalf of its users and applications.
2. *Cloud OS operation must continue despite failure of nodes, clusters, and network partitioning*: Guaranteeing continued operation of the cloud management processes in these conditions involves mechanisms for

quickly detecting the failures and enacting appropriate measures for recovering from the failures. Several cloud libraries that implement common fault tolerance and state recovery features are provided for the customers to use.

3. *The cloud must support multiple types of applications*: Applications of different types such as high-performance computing, high data availability, and high network throughput should ideally coexist in a cloud environment and obtain from the system the resources that best match the application requirements.
4. *The cloud OS management system must be decentralized, scalable, and cost effective*: Moreover, apart from initial resource deployment, no human intervention should be required to expand the cloud resources. Likewise, user management should only entail the on-demand creation of user credentials, which are then automatically propagated throughout the cloud.
5. *The resources used in the cloud architecture must be accountable*: The resource usage of the various cloud customers should be monitored effectively in the cloud environment. This monitoring activity could be used for charging the customers for their resource access, and also for the security auditing (if needed). Moreover, dynamic billing schemes based on resource congestion could be an effective way for resource allocation.

6.12.5 Cloud-Based OS

Researchers are now aiming to go one step further and take the OS to the cloud with TransOS, a cross-platform, cloud-based OS [53]. The TransOS system code is stored on a cloud server and a minimal amount of code would be required to boot up the computer and connect it to the Internet. Featuring a graphical user interface, TransOS downloads specific pieces of code to perform the same kinds of tasks as a conventional OS, thereby allowing a bare bones terminal to perform tasks beyond the limitations of its hardware. The terminal would make a call to the relevant TransOS code as and when required, ensuring that the inactive OS is not hogging system resources when applications are being run. The TransOS manages all the networked and virtualized hardware and software resources and enables the users to select and run any service on demand. The TransOS could be adapted to platforms other than PCs such as mobile devices, factory equipment, and even domestic appliances.

In addition to keeping a lean machine, TransOS users can also store their documents and files in the cloud, much like Apple's iCloud, keeping their local storage free. With TransOS, users never have to worry about running the most up-to-date version of the OS or even maintain their own computer. With their OS, data, files, and settings stored in the cloud, any computer with an Internet connection (e.g., computers that are publicly available at

libraries and colleges) becomes just like the user's own machine. The files stored within the cloud can also be accessed anytime from any Internet-ready device, including smartphones and tablets.

6.13 Application Environment

An application development environment (ADE) is the hardware, software, and computing resources required for building software applications. ADE is a composite set of computing resources that provides an interface for application development, testing, deployment, integration, troubleshooting, and maintenance services. These are combined with the software engineering resources, such as a programming language's integrated development environment (IDE), reporting and analysis software, troubleshooting tools, and other performance evaluation software utilities.

6.13.1 Need for Effective ADE

As the mobile web application market matures, competitive pressures and user expectations will drive application developers to differentiate their product offerings by providing value-added features [54]. The standards-based application environment must ensure the interoperability of the application development components. In particular, it must enable customized content, extensible functionality and advanced user interfaces.

For the web to be ubiquitous, web access devices must be present almost everywhere. For this to occur, web access devices must become small and portable. As web-enabled devices evolve from today's desktop computers to such things as cellular telephones, car radios, and personal organizers, the challenge will be to provide a common application authoring environment across a diverse range of devices. The existing *standard* web application environment consists of HTML, JavaScript, and an ad hoc collection of standard graphics file formats, processed by an HTML browser. To incorporate multimedia content and extend the expressiveness or the functionality of a user interface, Java applets and browser plug-ins can be used. They require extensions that are often device specific and require special installation.

Hence, the web application environment must provide application developers the tools they need to develop innovative products, without sacrificing interoperability. Hence, an effective ADE should

- Support multiple content types, including multimedia content
- Support multimodal user interaction
- Provide a framework for the integration of new technologies as they become available

The application environment should support a standard extensibility framework. As new media types, user agents, or supplemental services emerge, they should be integrated into the environment in a backward-compatible manner without affecting the performance of any existing applications.

6.13.2 Application Development Methodologies

Today, two development methodologies are widely used in application development: distributed and agile developments [55].

6.13.2.1 Distributed Development

This is the natural by-product of the Internet and the phenomenon that not all coding geniuses live within commuting distance from the workplace. Distributed development is a global development that brings its own challenges with collaboration and code management. There are applications available for distributed code management such as git and Subversion. They are widely used in distributed environments.

6.13.2.2 Agile Development

This is where cloud development can really be much more than just *online*. Since cloud environments can be provisioned instantly and nearly any configuration can be copied and activated, the possibilities for instant developments and test environments are very attractive to developers. Cloud development can also boost agile development by coordinating collaboration, planned sprints, and emergency bug fixes. Deploying to the cloud is also very useful for agile development. Prereleases can be pushed out to customers' test machines on their cloud almost instantly. Even if the customer is not in a cloud environment yet, prereleases can be posted on a public cloud for the customer to access and test remotely before accepting delivery of the final release of the application. Toolsets that can help the agile management in the cloud include Code2Cloud, in conjunction with Tasktop and CollabNet.

6.13.3 Power of Cloud Computing in Application Development

Cloud computing has effectively solved the financial and infrastructural problems associated with developing custom applications for the enterprises as it eases the financial investment that was previously required to set up the sophisticated developer environment necessary to build, test, and deploy custom applications in-house [56]. As a result, the introduction of cloud platforms has enabled developers to solely focus on creating highly scalable modern applications. Further, the process of marketing these custom applications is less time consuming and more effective as a result of the flexibility provided by cloud computing services. When applications are run

in the cloud, they are accessed as a service—this is known as Software as a Service (SaaS). By utilizing SaaS, the companies can deliver services in a cost-effective and efficient manner. This process enables businesses to work in conjunction with partners to develop applications and quickly distribute them in the market.

The advantages of using cloud computing services over traditional software go beyond just the drop in costs. The traditional methods of developing custom applications that often took months to complete has now dropped to just weeks. With all the required software and tools available in the cloud, developers can work more efficiently and productively than they could if they were using traditional software, where, more often than not, additional components were required to develop a complete application. Today's heavily simplified approach of accessing applications online allows developers to produce comprehensive enterprise-level applications simply through a web browser, without the technical difficulties associated with traditional solutions.

Another main benefit of using cloud computing services for application development is the efficient use of resources. Applications that utilize virtualized IT services are generally more efficient and better equipped to meet user demands. The pay-per-use model of cloud computing services provides the clients with flexibility to spend according to their requirements and thus eliminates the unnecessary expenses. Also, cloud computing services allow delivering the applications on multiple devices; this allows companies to design their applications so that they are compatible with a range of devices.

6.13.3.1 Disadvantages of Desktop Development

Desktop development environments are becoming outdated, failing more often, and causing productivity issues for developers. The main issues with desktop environment are the following [57]:

1. *Complicated configuration management*: The substantial configuration management process for a developer's workspace turns developers into part-time system administrators, responsible for their own mini-DC running entirely on the desktop. This is time consuming, error prone, and challenging to automate. Many developers have multiple computers and are forced to repeat these tasks on each machine. There is no way to synchronize the configurations of components across different machines, and each machine requires similar hardware and OSs to operate the components identically.
2. *Decreased productivity*: Many IDEs are memory and disk hogs, with significant boot times. They are so resource-hungry that they can starve other applications and the net effect is less productivity due to a slower machine.

3. *Limited accessibility*: Normally, desktop developer workspaces are not accessible via mobile devices through the Internet. Developers who need remote access have to resort to some complex and slow solutions such as *GotoMyPC*.
4. *Poor collaboration*: These days, most developers work as part of a team, so communication and collaboration among the team members are critical for the success of the project. In the case of desktop IDEs, they must outsource collaboration to communication systems outside the developer's workflow, forcing developers to continuously switch between developing within the IDE and communicating with their team via other means. To solve these problems, it requires moving the entire development workspace into the cloud. The cloud-based environment is centralized, making it easy to share. Developers can invite others into their workspace to coedit, cobuild, or codebug and can communicate with one another in the workspace itself. The cloud can offer improvements in system efficiency, giving each individual workspace a configurable slice of the available memory and computing resources.

6.13.3.2 Advantages of Application Development in the Cloud

Cloud platforms reduce the overall development time of a software project [58]. This is largely due to the cloud platform's ability to streamline the development process, including the ability to quickly get the development assets online. Moreover, cloud platforms provide the ability to collaborate effectively on development efforts. Cloud-based development platforms in PaaS and IaaS public clouds such as Google, Amazon Web Services, Microsoft, and Salesforce.com offer cost savings and better QoS.

Some of the benefits of the application development in the cloud are given as follows:

- The ability to self-provision development and testing environments
- The ability to quickly get applications into production and to scale those applications as required
- The ability to collaborate with other developers, architects, and designers on the development of the application

6.13.4 Cloud Application Development Platforms

Application development, deployment, and runtime management have always been reliant on development platforms such as Microsoft's .NET, WebSphere, or JBoss, which have been deployed on premise traditionally [59]. In the cloud computing context, applications are generally deployed by cloud

providers to provide highly scalable and elastic services to as many end users as possible. Cloud computing infrastructure needs to support many users to access and utilize the same application services, with elastic allocation of resources. This has led to the enhancement in development platform technologies and architectures to handle performance, security, resource allocation, application monitoring, billing, and fault tolerance. Cloud provides the ADE as PaaS. There are several solutions available in the PaaS market, including Google App Engine, Microsoft Windows Azure, Force.com, and Manjrasoft Aneka.

6.13.4.1 Windows Azure

Windows Azure provides a wide array of Windows-based services for developing and deploying Windows-based applications on the cloud. It makes use of the infrastructure provided by Microsoft to host these services and scale them seamlessly. The Windows Azure Platform consists of SQL Azure and the .NET services. The .NET services comprise of access control services and .NET service bus. Windows Azure is a platform with shared multitenant hardware provided by Microsoft. Windows Azure application development mandates the use of SQL Azure for RDBMS functionality, because that is the only coexisting DBMS functionality accessible in the same hardware context as the applications.

6.13.4.2 Google App Engine

Google App Engine provides an extensible runtime environment for web-based applications developed with Java or Python, which leverage huge Google IT infrastructure. Google App Engine is offered by Google, Inc. Its key value is that developers can rapidly build web-based applications on their machine and deploy them on the cloud. Google App Engine provides developers with a simulated environment to build and test applications locally with any OS or any system that runs a suitable version of Python and Java language environments. Google uses the JVM with Jetty Servlet engine and Java Data Objects.

6.13.4.3 Force.com

Force.com is a development and execution environment and is the best approach for PaaS for developing customer relationship management (CRM)–based applications. With regard to the design of its platform and the runtime environment, it is based on the Java technology. The platform uses a proprietary programming language and environment called Apex code, which has a reputation for simplicity in learning and rapid development and execution.

6.13.4.4 Manjrasoft Aneka

Aneka is a distributed application platform for developing cloud applications. Aneka can seam together any number of Windows-based physical or virtual desktops or servers into a network of interconnected nodes that act as a single logical *application execution layer.* Aneka-based clouds can be deployed on a variety of hardware and OSs including several flavors of the Windows and Linux OS families. Aneka provides a flexible model for developing distributed applications and provides integration with external clouds such as Amazon EC2 and GoGrid. Aneka offers the possibility to select the most appropriate infrastructure deployment without being tied to any specific vendor, thus allowing enterprises to comfortably scale to the cloud as and when needed.

6.13.5 Cloud Computing APIs

APIs are provided by some of the CSPs for the development of cloud applications. Details of some of the APIs provided by the CSPs such as Rackspace, IBM, and Intel are given in the following [60].

6.13.5.1 Rackspace

Developers have access to the API documentation and software development kit (SDK) across all of Rackspace's services at their developer site, http://developer.rackspace.com. Thus, Rackspace provides developers with the tools and resources necessary to create new applications and services on top of their APIs.

6.13.5.2 IBM

IBM introduced new APIs, which can be found at the IBM developer site, www.ibm.com/developerworks/. The introduction of the new APIs focuses on arming developers with the tools and resources to build new products, applications, and services.

6.13.5.3 Intel

Intel has several SDKs that aimed at cloud computing developers. Intel has a cloud services platform beta where developers can download the SDK for identity-based and cross-platform services. The Intel Cloud Builders program brings together leading systems and software solutions vendors to provide the best practices and practical guidance on how to deploy, maintain, and optimize a cloud infrastructure based on Intel architecture. And for developers seeking to use public cloud infrastructure services, the Intel Cloud Finder makes it easier to select providers that meet a developer's requirements.

6.14 Summary

Cloud computing is dominating the IT industry worldwide today. More and more companies and organizations are adopting the cloud model these days. Even though cloud computing is a new service delivery model, the underlying technologies have been in existence for a long time. Cloud computing uses many of those technologies to achieve its established goals. This chapter focuses on the various technological drivers of cloud computing. It discusses about the basic enabling technologies of cloud computing such as SOA, hypervisors and virtualization, multicore technology, and memory and storage technologies. It also talks about the latest developments in Web 2.0 and Web 3.0, the advancements in the programming models, software development models, pervasive computing, OSs, and ADEs. It also explains how these technologies are related to the cloud model, helping the cloud in delivering quality services. The recent developments in each of these enabling technologies are highlighted with their advantages and characteristic features. The chapter explains as to how these underlying technologies are empowering the present cloud computing paradigm to deliver its services effectively. Also, the chapter presents how various stakeholders such as service providers and service consumers are benefitted from the features extended by these technologies.

Review Points

- *SOA*: Service-oriented architecture is a flexible set of design principles and standards used for systems development and integration. A properly implemented SOA-based system provides a loosely coupled set of services that can be used by the service consumers for meeting their service requirements within various business domains (see Section 6.2).
- *Hypervisor*: Hypervisors are software tools used to create virtual machines, and they produce the virtualization of various hardware resources such as CPU, storage, and networking devices. They are also called virtual machine monitor (VMM) or virtualization managers (see Section 6.3.2).
- *Multicore technology*: In the multicore technology, two or more CPUs are working together on the same chip. In this type of architecture, a single physical processor contains the core logic of two or more processors (see Section 6.4).
- *Storage as a Service*: Storage as a Service (STaaS) is a cloud business model in which a service provider rents space in its storage infrastructure to various cloud users (see Section 6.5.3).

- *Software-defined networking*: Software-defined networking (SDN) is an approach to networking in which control is decoupled from networking hardware and given to a software application called the controller (see Section 6.6.5).
- *Web 2.0*: Web 2.0 (or Web 2) is the popular term given to the advanced Internet technology and applications that include blogs, wikis, RSS, and social bookmarking (see Section 6.7).
- *Semantic web*: The semantic web is a vision of IT that allows data and information to be readily interpreted by machines, so that the machines are able to take contextual decisions on their own by finding, combining, and acting upon relevant information on the web (see Section 6.8.1.1).
- *Agile development model*: Agile model is a software development model where the software is developed in rapid, incremental cycles. The development results in tiny incremental releases and is based on previously built functionality and is carefully tested to ensure software quality (see Section 6.9.3).
- *MapReduce*: MapReduce is a popular programming model designed to support the development of compute- and data-intensive applications, which requires high storage and processing demands (see Section 6.10.1.2).
- *Pervasive computing*: Pervasive computing is a combination of technologies such as Internet capabilities, voice recognition, networking, artificial intelligence, and wireless computing used to make computing anywhere possible (see Section 6.11).
- *Web OS*: Web operating systems are basically websites that replicate the desktop environment of modern OSs, all inside a web browser (see Section 6.12.1).
- *Cloud API*: Cloud APIs are provided by the cloud service providers for the development of cloud applications (see Section 6.13.5).

Review Questions

1. What are the characteristic features of SOA that are used in the successful deployment of cloud computing?
2. What are the various approaches in virtualization? What are the roles played by the hypervisor and virtualization in cloud environment?
3. How can the multicore technologies be used to achieve the parallelism in cloud?

4. What are the latest technological developments to meet the storage requirements in cloud?
5. How does SDN relate to the cloud computing scenario?
6. What are the ways in which cloud computing relies on the concepts of Web 2.0 for its successful operation?
7. How do semantic web and web services contribute to the evolution of cloud computing?
8. Justify the decision to adopt the agile development model for software development. How can the cloud computing paradigm make the agile process effective?
9. What are the programming models used in cloud? Justify the answer by explaining the characteristic features of the models.
10. Explain the ways in which pervasive computing affects the cloud model.
11. Explain the differences between a web OS and a cloud OS.
12. How does the cloud computing paradigm help in effective application development?

References

1. Strassmann, P. A. How SOA fits into cloud computing. *SOA Symposium*, April 22, 2010.
2. Zhang, L.-J. and Q. Zhou. CCOA: Cloud computing open architecture. *IEEE International Conference on Web Services 2009 (ICWS 2009)*, 2009, pp. 607–616.
3. Gschwind, M. *Multicore Computing and the Cloud: Optimizing Systems with Virtualization*. IBM Corporation, 2009.
4. Sun, X.-H., Y. Chen, and S. Byna. Scalable computing in the multicore era. *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Programming*, 2008.
5. Sankaralingam, K. and R. H. Arpaci-Dusseau. Get the parallelism out of my loud. *Proceedings of the Second USENIX Conference on Hot Topics in Parallelism*, 2010.
6. Jamal, M. H. et al. Virtual machine scalability on multi-core processors based servers for cloud computing workloads. *IEEE International Conference on Networking, Architecture, and Storage, 2009 (NAS 2009)*, Hunan, China, July 9–11, 2009. IEEE, New York, 2009, pp. 90–97.
7. Venkatraman, A. Intel launches micro-server, network, storage technologies to power cloud datacenters. Available at: http://www.computerweekly.com/news/2240204767/Intel-launches-micro-server-network-storage-technologies-to-power-cloud-datacentres. Accessed December 10, 2013.
8. Introduction to storage technologies. Consulting Solutions, White Paper, Citrix XenDesktop.

9. Cloud storage for cloud computing. OpenGrid Forum, SNIA, Advancing Storage and Information Technology, White Paper, 2009.
10. Stryer, P. Understanding data centers and cloud computing. Global Knowledge Instructor, CCSI, CCNA.
11. Ingthorsson, O. Networking technologies in cloud computing. Available at: http://cloudcomputingtopics.com/2010/04/networking-technologies-in-cloud-computing/#comments. Accessed December 13, 2013.
12. Bitar, N., S. Gringeri, and T. J. Xia. Technologies and protocols for data center and cloud networking. *IEEE Communications Magazine* 51(9): 24–31, 2013.
13. Rosen, E. and Y. Rekhter. BGP/MPLS IP virtual private networks (VPNs). RFC 4364, 2006.
14. Sajassi et al. BGP MPLS based ethernet VPN. Work in progress, 2013.
15. O'Reilly, T. What is Web 2.0. O'Reilly Network, 2005. Accessed August 6, 2006.
16. VMWARE. Virtualization overview. Available at: www.vmware.com.
17. Techtarget. Definition of Web 2.0. Available at: http://whatis.techtarget.com/definition/Web-20-or-Web-2. Accessed December 1, 2013.
18. Reservoir Consortium. Resources and services virtualization without barriers. Scientific report. 2009.
19. Mulholland, A., J. Pyke, and P. Finger. *Enterprise Cloud Computing: A Strategy Guide for Business and Technology*, Meghan-Kiffer Press, Tampa, FL.
20. Keen, A., Web 1.0 + Web 2.0 = Web 3.0, Typepad.com. Available at: http://andrewkeen.typepad.com/the-great-seduction/2008/04/web-10-web20-w.html. Accessed November 21, 2013.
21. Viluda, P., Differences between Web 3.0 and Web 2.0 standards. Available at: http://www.cruzine.com/2011/02/14/web-3-web-2-standards/.
22. World Wide Web Consortium (W3C). W3C semantic web activity, 2011. Retrieved November 26, 2011. Accessed November 26, 2011.
23. Getting, B. Basic definitions: Web 1.0, Web. 2.0, Web 3.0. Available at: http://www.practicalecommerce.com/articles/464-Basic-Definitions-Web-1-0-Web-2-0-Web-3-0. Accessed December 1, 2013.
24. Spivack, N. Web 3.0: The third generation web is coming. Available at: http://lifeboat.com/ex/web.3.0. Accessed December 1, 2013.
25. Hoy, T. Web 3.0: Converging cloud computing and the web. Available at: http://www.ebizq.net/topics/cloud_computing/features/12477.html?page=3. Accessed November 23, 2013.
26. Shaw, T. Web 3.0 gives business smarter infrastructure. Available at: http://www.baselinemag.com/cloud-computing/Web-30--Gives-Business-Smarter-Infrastructure/. Accessed November 27, 2013.
27. Sommerville, I. *Software Engineering*, 8th edn. Pearson Education, 2006.
28. Guha, R. and D. Al-Dabass. Impact of Web 2.0 and cloud computing platform on software engineering. *2010 International Symposium on Electronic System Design (ISED)*, Bhubaneswar, India, December 20–22, 2010. IEEE, New York, 2010, pp. 213–218.
29. Pressman, R. *Software Engineering: A Practitioner's Approach*, 7th edn. McGraw-Hill Higher Education, New York, 2009.
30. Singh, A., M. Korupolu, and D. Mahapatra. Server-storage virtualization: Integration and load balancing in data centers. *International Conference for High Performance Computing, Networking, Storage and Analysis, 2008 (SC 2008)*, Austin, TX, November 15–21, 2008. IEEE/ACM Supercomputing (SC), 2008, pp. 1–12.

31. Velagapudi, M. SDLC for cloud computing—How is it different from the traditional SDLC? Available at: http://blog.bootstraptoday.com/2012/02/06/sdlc-for-cloud-computing-how-is-it-different-from-the-traditional-sdlc/.
32. Salesforce.com. Agile development meets Cloud computing for extraordinary results. Available at: www.salesforce.com. Accessed October 3, 2013.
33. Dumbre, A., S. S. Ghag, and S. P. Senthil. Practising Agile software development on the Windows Azure platform. Infosys Whitepaper, 2011.
34. Gulrajani, N. and D. Bowler. *Software Development in the Cloud–Cloud Management and ALM*.
35. Kannan, N. Ways the cloud enhances agile software development. Available at: http://www.cio.com/article/714210/6_Ways_the_Cloud_Enhances_Agile_Software_Development. Accessed December 5, 2013.
36. Mahmood, Z. and S. Saeed. *Software Engineering Frameworks for Cloud Computing Paradigm*. Springer-Verlag, London, U.K., 2013.
37. Liu, X. A programming model for the cloud platform. *International Journal of Advanced Science and Technology* 57: 75–81, 2013.
38. Jayaraj, A., J. John Geevarghese, K. Rajan, U. Kartha, and V. Samuel Varghese. *Programming Models for Clouds*.
39. McCormick, P. et al. Programming models. White Paper. Available at: https://asc.llnl.gov/exascale/exascale-pmWG.pdf.
40. Dean, J. and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM* 51(1): 107–113, 2008. Accessed December 8, 2013.
41. Iiard, M., M. Budiu, and Y. Yuan. Dryad: Distributed data-parallel programs from sequential building blocks. *Operating Systems Review* 41(3): 59–72, 2007.
42. Jin, C. and R. Buyya. Mapreduce programming model for. NET-based cloud computing. In: H. Sips, D. Epema, and H.-X. Lin (eds.), *Euro-Par 2009 Parallel Processing*. Springer, Berlin, Germany, 2009, pp. 417–428.
43. Miceli, C. et al. Programming abstractions for data intensive computing on clouds and grids. *9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, 2009.
44. Wang, P. et al. *Transformer: A New Paradigm for Building Data-Parallel Programming Models*. IEEE Computer Society, 2009.
45. Gannon, D. The computational data center—A science cloud. Indiana University, Bloomington, IN.
46. Soylu, A., P. De Causmaecker, and P. Desmet. Context and adaptivity in pervasive computing environments: Links with software engineering and ontological engineering. *Journal of Software* 4(9): 992–1013, 2009.
47. Gallagher, S. Forget "post-PC"—Pervasive computing and cloud will change the nature of IT. Available at: http://arstechnica.com/information-technology/2013/08/forget-post-pc-pervasive-computing-and-cloud-will-change-the-nature-of-it/2/. Accessed October 24, 2013.
48. Perspectives. Ubiquitous computing: Beyond mobility: Everywhere and every thing. *TCS Consulting Journal*. Available at: http://sites.tcs.com/insights/perspectives/enterprise-mobility-ubiquitous-computing-beyond-mobility#.UzPo8fmSzCe.
49. Namboodiri, V. Sustainable pervasive computing through mobile clouds. Available at: http://sensorlab.cs.dartmouth.edu/NSFPervasiveComputingAtScale/pdf/1569391485.pdf. Accessed October 28, 2013.

50. Steddum, J. A brief history of cloud computing. Available at: http://blog.softlayer.com/tag/mainframe.
51. Hurwitz, J. The role of the operating system in cloud environments. A Hurwitz White Paper, 2011.
52. Pianese, F. et al. Toward a cloud operating system. *Network Operations and Management Symposium Workshops (NOMS Wksps), 2010 IEEE/IFIP*. IEEE, 2010.
53. Quick, D. Cloud-based operating system in the works. Available at: http://www.gizmag.com/transos-cloud-based-operarting-system/24494/. Accessed December 14, 2013.
54. Dominiak, D. Standardizing a web-based application environment. Motorola White Paper. Available at: http://www.w3.org/2000/09/Papers/Motorola.html. Accessed December 1, 2013.
55. Proffitt, B. Building applications in the cloud: A tour of the tools. Available at: http://www.itworld.com/virtualization/189811/building-applications-cloud-tour-tools. Accessed November 24, 2013.
56. Siddiqui, Z. The impact of cloud computing on custom application development. Available at: http://www.trackvia.com/blog/technology/cloud_computing_and_custom_application_developments. Accessed December 6, 2013.
57. Linthicum, D. Why application development is better in the cloud. Available at: http://www.infoworld.com/d/cloud-computing/why-application-development-better-in-the-cloud-211239. Accessed December 10, 2013.
58. Jewell, T. Why cloud development environments are better than desktop development. Available at: http://readwrite.com/2013/04/16/why-cloud-development-environments-are-better-than-desktop-development#awesm=~ozHQROKsJUAOI2. Accessed December 4, 2013.
59. Buyya, R. and K. Sukumar. Platforms for building and deploying applications for cloud computing. arXiv preprint arXiv:1104.4379, 2011.
60. Le, T. Developers and cloud computing application programming interfaces (APIs). Available at: http://software.intel.com/en-us/blogs/2013/09/26/developers-and-cloud-computing-application-programming-interfaces-apis.

Further Reading

Dean, J. and S. Ghemawat. MapReduce: Simplified data processing on large clusters. *Proceedings of the Sixth Symposium on Operating System Design and Implementation*, 2004.