Third Semster MCA Degree Examination Jan/Feb 2023

Cloud Computing

20MCA342

Q1 a) What are the uses of creating data centre? Explain practical examples of cloud computing.

A data center is a facility that centralizes an organization’s IT operations and equipment for the purposes of storing, processing and disseminating data and applications. Because they house an organization's most critical and proprietary assets, data centers are vital to the continuity of daily operations. Consequently, security and reliability are among any organization’s top priorities.

Cloud computing is helping enterprises, governments, public and private institutions, and research organizations shape more effective and demand-driven computing systems. Access to, as well as integration of, cloud computing resources and systems is now as easy as performing a credit card transaction over the Internet. Practical examples of such systems exist across all market segments:

• Large enterprises can offload some of their activities to cloud-based systems. Recently, the New York Times has converted its digital library of past editions into a Web-friendly format. This required a considerable amount of computing power for a short period of time. By renting Amazon EC2 and S3 Cloud resources, the Times performed this task in 36 hours and relinquished these resources, with no additional costs.

• Small enterprises and start-ups can afford to translate their ideas into business results more quickly, without excessive up-front costs. Animoto is a company that creates videos out of images, music, and video fragments submitted by users. The process involves a considerable amount of storage and backend processing required for producing the video, which is finally made available to the user. Animoto does not own a single server and bases its computing infrastructure entirely on Amazon Web Services, which are sized on demand according to the overall workload to be processed. Such workload can vary a lot and require instant scalability.3 Up-front investment is clearly not an effective solution for many companies, and cloud computing systems become an appropriate alternative.

• System developers can concentrate on the business logic rather than dealing with the complexity of infrastructure management and scalability. Little Fluffy Toys is a company in London that has developed a widget providing users with information about nearby bicycle rental services. The company has managed to back the widget’s computing needs on Google AppEngine and be on the market in only one week.

• End users can have their documents accessible from everywhere and any device. Apple iCloud is a service that allows users to have their documents stored in the Cloud and access them from any device users connect to it. This makes it possible to take a picture while traveling with a smartphone, go back home and edit the same picture on your laptop, and have it show as updated on your tablet computer.

This process is completely transparent to the user, who does not have to set up cables and connect these devices with each other.

Q1 b) With a neat diagram, explain cloud computing reference model.

A fundamental characteristic of cloud computing is the capability to deliver, on demand, a variety of IT services that are quite diverse from each other. This variety creates different perceptions of what cloud computing is among users. Despite this lack of uniformity, it is possible to classify cloud computing services offerings into three major categories: Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS).

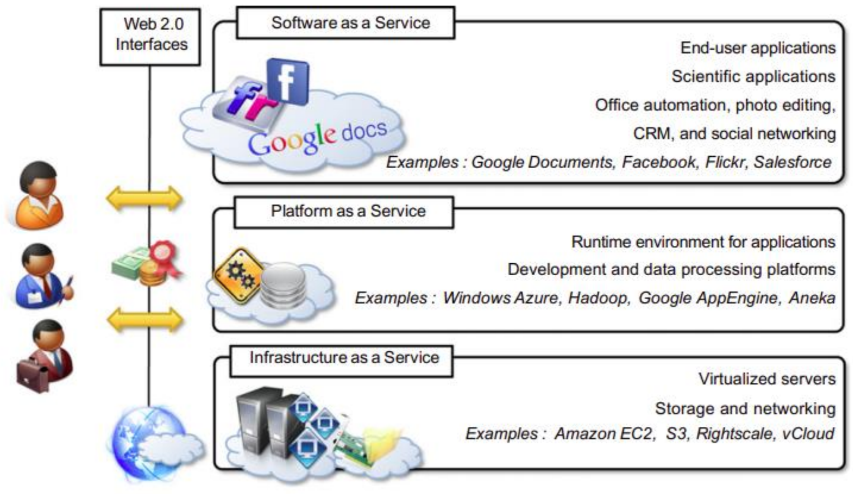

Web 2.0 Interfaces
Software as a Service
End-user applications
Scientific applications
Office automation, photo editing,
CRM, and social networking
Examples : Google Documents, Facebook, Flickr, Salesforce
Platform as a Service
Runtime environment for applications
Development and data processing platforms
Examples : Windows Azure, Hadoop, Google AppEngine, Aneka
Infrastructure as a Service
Virtualized servers
Storage and networking
Examples : Amazon EC2, S3, Rightscale, vCloud

FIGURE 1.5

The Cloud Computing Reference Model.

At the base of the stack, **Infrastructure-as-a-Service** solutions deliver infrastructure on demand in the form of virtual hardware, storage, and networking. Virtual hardware is utilized to provide compute on demand in the form of virtual machine instances. These are created at users' request on the provider's infrastructure, and users are given tools and interfaces to configure the software stack installed in the virtual machine. The pricing model is usually defined in terms of dollars per hour, where the hourly cost is influenced by the characteristics of the virtual hardware. Virtual storage is delivered in the form of raw disk space or object store.. Virtual networking identifies the collection of services that manage the networking among virtual instances and their connectivity to the Internet or private networks.

Platform-as-a-Service solutions are the next step in the stack. They deliver scalable and elastic runtime environments on demand and host the execution of applications. These services are backed by a core middleware platform that is responsible for creating the abstract environment where applications are deployed and executed. It is the responsibility of the service provider to provide scalability and to manage fault tolerance, while users are requested to focus on the logic of the application developed by leveraging the provider's APIs and libraries. This approach increases the level of abstraction at which cloud computing is leveraged but also constrains the user in a more controlled environment.

At the top of the stack, **Software-as-a-Service** solutions provide applications and services on demand. Most of the common functionalities of desktop applications—such as office automation, document management, photo editing, and customer relationship management (CRM) software—are replicated on the provider's infrastructure and made more scalable and accessible through a browser on demand. These applications are shared across multiple users whose interaction is isolated from the other users. The SaaS layer is also the area of social networking Websites, which leverage cloud-based infrastructures to sustain the load generated by their popularity.

Each layer provides a different service to users. IaaS solutions are sought by users who want to leverage cloud computing from building dynamically scalable computing systems requiring a specific software stack. IaaS services are therefore used to develop scalable Websites or for background processing. PaaS solutions provide scalable programming platforms for developing applications and are more appropriate when new systems have to be developed. SaaS solutions target mostly end users who want to benefit from the elastic scalability of the cloud without doing any software development, installation, configuration, and maintenance. This solution is appropriate when there are existing SaaS services that fit users needs (such as email, document management, CRM, etc.) and a minimum level of customization is needed.

Q2 a) Define cloud computing Explain the characteristics and benefits of cloud computing

The term cloud has historically been used in the telecommunications industry as an abstraction of the network in system diagrams. It then became the symbol of the most popular computer network: the Internet. This meaning also applies to cloud computing, which refers to an Internet-centric way of computing.

The Internet plays a fundamental role in cloud computing, since it represents either the medium or the platform through which many cloud computing services are delivered and made accessible. This aspect is also reflected in the definition given by Armbrust et al. [28]:

Cloud computing refers to both the applications delivered as services over the Internet and the hardware and system software in the datacenters that provide those services.

This definition describes cloud computing as a phenomenon touching on the entire stack: from the underlying hardware to the high-level software services and applications. It introduces the concept of everything as a service, mostly referred as XaaS, 2 where the different components of a system—IT infrastructure, development platforms, databases, and so on—can be delivered, measured, and consequently priced as a service

This notion of multiple parties using a shared cloud computing environment is highlighted in a definition proposed by the U.S. National Institute of Standards and Technology (NIST):

Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

According to Reese [29], we can define three criteria to discriminate whether a service is delivered in the cloud computing style:

- *The service is accessible via a Web browser (nonproprietary) or a Web services application programming interface (API).*
- *Zero capital expenditure is necessary to get started.*
- *You pay only for what you use as you use it.*

The utility-oriented nature of cloud computing is clearly expressed by Buyya et al. [30]:

A cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers.

Cloud computing has some interesting characteristics that bring benefits to both cloud service consumers (CSCs) and cloud service providers (CSPs). These characteristics are:

- No up-front commitments
- On-demand access
- Nice pricing
- Simplified application acceleration and scalability
- Efficient resource allocation
- Energy efficiency
- Seamless creation and use of third-party services

Q2 b) Explain how the two technologies grid computing and web 2.0 has contributed for cloud computing

Grid computing appeared in the early 1990s as an evolution of cluster computing. In an analogy to the power grid, grid computing proposed a new approach to access large computational power, huge storage facilities, and a variety of services. Users can “consume” resources in the same way as they use other utilities such as power, gas, and water. Grids initially developed as aggregations of geographically dispersed clusters by means of Internet connections. These clusters belonged to different organizations, and arrangements were made among them to share the computational power. Different from a “large cluster,” a computing grid was a dynamic aggregation of heterogeneous computing nodes, and its scale was nationwide or even worldwide. Several developments made possible the diffusion of computing grids: (a) clusters became quite common resources; (b) they were often underutilized; (c) new problems were requiring computational power that went beyond the capability of single clusters; and (d) the improvements in networking and the diffusion of the Internet made possible long-distance, high-bandwidth connectivity. All these elements led to the development of grids, which now serve a multitude of users across the world.

Cloud computing is often considered the successor of grid computing. In reality, it embodies aspects of all these three major technologies. Computing clouds are deployed in large datacenters hosted by a single organization that provides services to others. Clouds are characterized by the fact of having virtually infinite capacity, being tolerant to failures, and being always on, as in the case of mainframes. In many cases, the computing nodes that form the infrastructure of computing clouds are commodity machines, as in the case of clusters. The services made available by a cloud vendor are consumed on a pay-per-use basis, and clouds fully implement the utility vision introduced by grid computing.

Web 2.0

The Web is the primary interface through which cloud computing delivers its services. At present, the Web encompasses a set of technologies and services that facilitate interactive information sharing, collaboration, user-centered design, and application composition. This evolution has transformed the Web into a rich platform for application development and is known as Web 2.0. This term captures a new way in which developers architect applications and deliver services through the Internet and provides new experience for users of these applications and services. Web 2.0 brings interactivity and flexibility into Web pages, providing enhanced user experience by gaining Web-based access to all the functions that are normally found in desktop applications. These capabilities are obtained by integrating a collection of standards and technologies such as XML, Asynchronous JavaScript and XML (AJAX), Web Services, and others. These technologies allow us to build applications leveraging the contribution of users, who now become providers of content. Furthermore, the capillary diffusion of the Internet opens new opportunities and markets for the Web, the services of which can now be accessed from a variety of devices: mobile phones, car dashboards, TV sets, and others. These new scenarios require an increased dynamism for applications, which is another key element of this technology. Web 2.0

applications are extremely dynamic: they improve continuously, and new updates and features are integrated at a constant rate by following the usage trend of the community. There is no need to deploy new software releases on the installed base at the client side. Users can take advantage of the new software features simply by interacting with cloud applications. Lightweight deployment and programming models are very important for effective support of such dynamism. Loose coupling is another fundamental property. New applications can be "synthesized" simply by composing existing services and integrating them, thus providing added value. This way it becomes easier to follow the interests of users. Finally, Web 2.0 applications aim to leverage the "long tail" of Internet users by making themselves available to everyone in terms of either media accessibility or affordability

Q3 a) Explain different hardware architecture for parallel processing

The core elements of parallel processing are CPUs. Based on the number of instruction and data streams that can be processed simultaneously, computing systems are classified into the following four categories:

• Single-instruction, single-data (SISD) systems

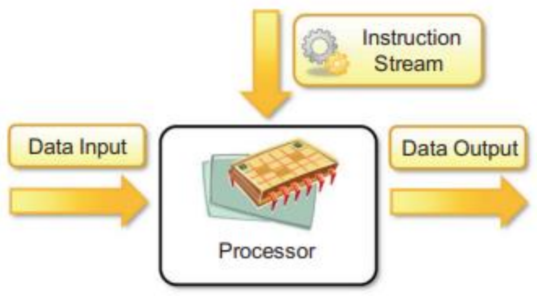
• Single-instruction, multiple-data (SIMD) systems

• Multiple-instruction, single-data (MISD) systems

• Multiple-instruction, multiple-data (MIMD) systems

Single-instruction, single-data (SISD) systems

An SISD computing system is a uniprocessor machine capable of executing a single instruction, which operates on a single data stream (see Figure 2.2). In SISD, machine instructions are processed sequentially; hence computers adopting this model are popularly called sequential computers. Most conventional computers are built using the SISD model. All the instructions and data to be processed have to be stored in primary memory. The speed of the processing element in the SISD model is limited by the rate at which the computer can transfer information internally. Dominant representative SISD systems are IBM PC, Macintosh, and workstations



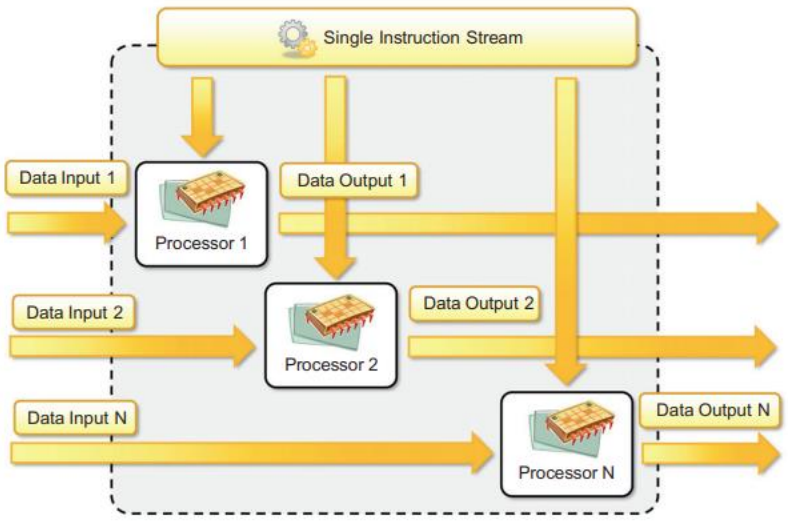
Single-instruction, multiple-data (SIMD) systems

An SIMD computing system is a multiprocessor machine capable of executing the same instruction on all the CPUs but operating on different data streams. Machines based on an SIMD model are well suited to scientific computing since they involve lots of vector and matrix operations. For instance, statements such as

Ci 5 Ai ———————————————— Bi

can be passed to all the processing elements (PEs); organized data elements of vectors A and B can be divided into multiple sets (N-sets for N PE systems); and each PE can process one data set. Dominant representative SIMD systems are Cray's vector processing machine and Thinking Machines' cm*.

$$y = \sin(x) + \cos(x) + \tan(x)$$



Multiple-instruction, single-data (MISD) systems

An MISD computing system is a multiprocessor machine capable of executing different instructions on different PEs but all of them operating on the same data set (see Figure 2.4). For instance, statements such as

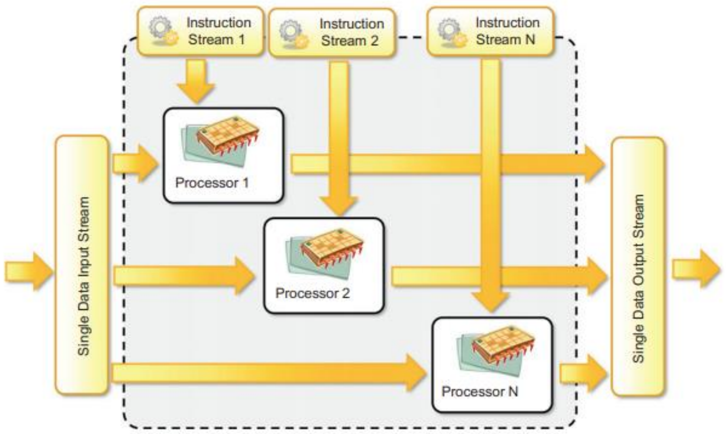


FIGURE 2.4
Multiple-instruction, single-data (MISD) architecture.

perform different operations on the same data set. Machines built using the MISD model are not useful in most of the applications; a few machines are built, but none of them are available commercially. They became more of an intellectual exercise than a practical configuration.

Multiple-instruction, multiple-data (MIMD) systems

An MIMD computing system is a multiprocessor machine capable of executing multiple instructions on multiple data sets (see Figure 2.5). Each PE in the MIMD model has separate instruction and data streams; hence machines built using this model are well suited to any kind of application. Unlike SIMD and MISD machines, PEs in MIMD machines work asynchronously. MIMD machines are broadly categorized into shared-memory MIMD and distributed-memory MIMD based on the way PEs are coupled to the main memory.

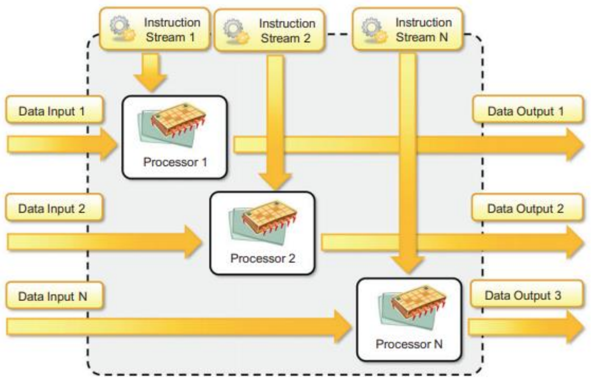


FIGURE 2.5
Multiple-instructions, multiple-data (MIMD) architecture.

Q3 b) Define components and connector. Explain data-centered architectural style

A component represents a unit of software that encapsulates a function or a feature of the system. Examples of components can be programs, objects, processes, pipes, and filters. A connector is a communication mechanism that allows cooperation and coordination among components. Differently from components, connectors are not encapsulated in a single entity, but they are implemented in a distributed manner over many system components.

Data centered architectures

These architectures identify the data as the fundamental element of the software system, and access to shared data is the core characteristic of the data-centered architectures. Therefore, especially within the context of distributed and parallel computing systems, integrity of data is the overall goal for such systems. The repository architectural style is the most relevant reference model in this category. It is characterized by two main components: the central data structure, which represents the current state of the system, and a collection of independent components, which operate on the central data. The ways in which the independent components interact with the central data structure can be very heterogeneous. In particular, repository-based architectures differentiate and specialize further into subcategories according to the choice of control discipline to apply for the shared data structure. Of particular interest are databases and blackboard systems. In the former group the dynamic of the system is controlled by the independent components, which, by issuing an operation on the central repository, trigger the selection of specific processes that operate on data. In blackboard systems, the central data structure is the main trigger for selecting the processes to execute.

The blackboard architectural style is characterized by three main components:

• Knowledge sources. These are the entities that update the knowledge base that is maintained in the blackboard.

• Blackboard. This represents the data structure that is shared among the knowledge sources and stores the knowledge base of the application.

• Control. The control is the collection of triggers and procedures that govern the interaction with the blackboard and update the status of the knowledge base.

Within this reference scenario, knowledge sources, which represent the intelligent agents sharing the blackboard, react opportunistically to changes in the knowledge base, almost in the same way that a group of specialists brainstorm in a room in front of a blackboard. Blackboard models have become popular and widely used for artificial intelligent applications in which the blackboard maintains the knowledge about a domain in the form of assertions and rules, which are entered by domain experts. These operate through a control shell that controls the problem-solving activity of the system. Particular and successful applications of this model can be found in the domains of speech recognition and signal processing.

Q4 a) Explain client server architectural style

Client/server This architecture is very popular in distributed computing and is suitable for a wide variety of applications. As depicted in Figure 2.12, the client/server model features two major components: a server and a client. These two components interact with each other through a network connection using a given protocol. The communication is unidirectional: The client issues a request to the server, and after processing the request the server returns a response. There could be multiple client components issuing requests to a server that is passively waiting for them. Hence, the important operations in the client-server paradigm are request, accept (client side), and listen and response (server side). The client/server model is suitable in many-to-one scenarios, where the information and the services of interest can be centralized and accessed through a single access point: the server. In general, multiple clients are interested in such services and the server must be appropriately designed to efficiently serve requests coming from different clients. This consideration has implications on both client design and server design. For the client design, we identify two major models:

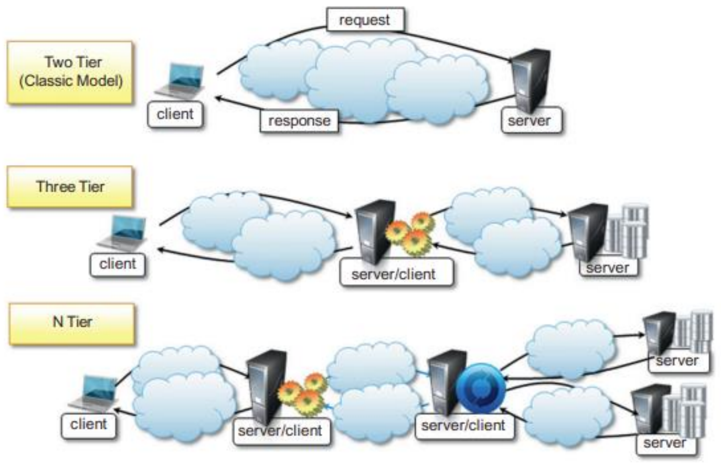


FIGURE 2.12
Client/server architectural styles.

• Thin-client model. In this model, the load of data processing and transformation is put on the server side, and the client has a light implementation that is mostly concerned with retrieving and returning the data it is being asked for, with no considerable further processing.

• Fat-client model. In this model, the client component is also responsible for processing and transforming the data before returning it to the user, whereas the server features a relatively light implementation that is mostly concerned with the management of access to the data.

The three major components in the client-server model: presentation, application logic, and data storage. In the thin-client model, the client embodies only the presentation component, while the server absorbs the other two. In the fat-client model, the client encapsulates presentation and most of the application logic, and the server is principally responsible for the data storage and maintenance

Presentation, application logic, and data maintenance can be seen as conceptual layers, which are more appropriately called tiers. The mapping between the conceptual layers and their physical implementation in modules and components allows differentiating among several types of architectures, which go under the name of multitiered architectures. Two major classes exist:

• Two-tier architecture. This architecture partitions the systems into two tiers, which are located one in the client component and the other on the server. The client is responsible for the presentation tier by providing a user interface; the server concentrates the application logic and the data store into a single tier. The server component is generally deployed on a powerful machine that is capable of processing user requests, accessing data, and executing the application logic to provide a client with a response. This architecture is suitable for systems of limited size and suffers from scalability issues. In particular, as the number of users increases the performance of the server might dramatically decrease. Another limitation is caused by the dimension of the data to maintain, manage, and access, which might be prohibitive for a single computation node or too large for serving the clients with satisfactory performance.

• Three-tier architecture/N-tier architecture. The three-tier architecture separates the presentation of data, the application logic, and the data storage into three tiers. This architecture is generalized into an N-tier model in case it is necessary to further divide the stages composing the application logic and storage tiers. This model is generally more scalable than the two-tier one because it is possible to distribute the tiers into several computing nodes, thus isolating the performance bottlenecks. At the same time, these systems are also more complex to understand and manage. A classic example of three-tier architecture is constituted by a medium-size Web application that relies on a relational database management system for storing its data. In this scenario, the client component is represented by a Web browser that embodies the presentation tier, whereas the application server encapsulates the business logic tier, and a database server machine (possibly replicated for high availability) maintains the data storage. Application servers that rely on third-party (or external) services to satisfy client requests are examples of N-tiered architectures.

The client/server architecture has been the dominant reference model for designing and deploying distributed systems, and several applications to this model can be found. The most relevant is perhaps the Web in its original conception. Nowadays, the client/server model is an important building block of more complex systems, which implement some of their features by identifying a server and a client process interacting through the network. This model is generally suitable in the case of a many-to-one scenario, where the interaction is unidirectional and started by the clients and suffers from scalability issues, and therefore it is not appropriate in very large systems

Q4 b) What is a service? Explain the characteristics that identify service.

A service encapsulates a software component that provides a set of coherent and related functionalities that can be reused and integrated into bigger and more complex applications. The term service is a general abstraction that encompasses several different implementations using different technologies and protocols. Don Box identifies four major characteristics that identify a service:

• Boundaries are explicit. A service-oriented application is generally composed of services that are spread across different domains, trust authorities, and execution environments. Generally, crossing such boundaries is costly; therefore, service invocation is explicit by design and often leverages message passing. With respect to distributed object programming, whereby remote method invocation is transparent, in a service-oriented computing environment the interaction with a service is explicit and the interface of a service is kept minimal to foster its reuse and simplify the interaction.

• Services are autonomous. Services are components that exist to offer functionality and are aggregated and coordinated to build more complex system. They are not designed to be part of a specific system, but they can be integrated in several software systems, even at the same time. With respect to object orientation, which assumes that the deployment of applications is atomic, service orientation considers this case an exception rather than the rule and puts the focus on the design of the service as an autonomous component. The notion of autonomy also affects the way services handle failures. Services operate in an unknown environment and interact with third-party applications. Therefore, minimal assumptions can be made concerning such environments: applications may fail without notice, messages can be malformed, and clients can be unauthorized. Service-oriented design addresses these issues by using transactions, durable queues, redundant deployment and failover, and administratively managed trust relationships among different domains.

• Services share schema and contracts, not class or interface definitions. Services are not expressed in terms of classes or interfaces, as happens in object-oriented systems, but they define themselves in terms of schemas and contracts. A service advertises a contract describing the structure of messages it can send and/or receive and additional constraint—if any—on their ordering. Because they are not expressed in terms of types and classes, services are more easily consumable in wider and heterogeneous environments. At the same time, a service orientation requires that contracts and schema remain stable over time, since it would be possible to propagate changes to all its possible clients. To address this issue, contracts and schema are defined in a way that allows services to evolve without breaking already deployed code. Technologies such as XML and SOAP provide the appropriate tools to support such features rather than class definition or an interface declaration.

• Services compatibility is determined based on policy. Service orientation separates structural compatibility from semantic compatibility. Structural compatibility is based on contracts and schema and can be validated or enforced by machine-based techniques. Semantic compatibility is expressed in the form of policies that define the capabilities and requirements for a service. Policies are organized in terms of expressions that must hold true to enable the normal operation of a service.

Q5 a) Explain the advantages of visualization in detail.

Virtualization allows the creation of a secure, customizable, and isolated execution environment for running applications, even if they are untrusted, without affecting other users' applications. The basis of this technology is the ability of a computer program—or a combination of software

and hardware—to emulate an executing environment separate from the one that hosts such programs.
Virtualization is a large umbrella of technologies and concepts that are meant to provide an abstract environment—whether virtual hardware or an operating system—to run applications.
Hardware virtualization plays a fundamental role in efficiently delivering Infrastructure-as-a-Service (IaaS) solutions for cloud computing
Virtualization technology provide a virtual environment for not only executing applications but also for storage, memory, and networking

- Increased performance and computing capacity

Now a days the average end-user desktop PC is powerful enough to fulfil almost all the needs of everyday computing and there is an extra capacity that is rarely used.
Individual PCs have resources enough to host a virtual machine manager and execute a virtual machine with by far acceptable performance
Likewise, the supercomputers can provide immense compute power that can accommodate the execution of hundreds or thousands of virtual machines

- Underutilized hardware and software resources

Computers today are so powerful that in most cases only a fraction of their capacity is used by an application or the system
To improve the efficiency of the IT infrastructure, to transparently provide such a service, it would be necessary to deploy a completely separate environment, which can be achieved through virtualization.

- Lack of space

The continuous need for additional capacity, whether storage or compute power, makes data centers grow quickly adding the size of data centers for every need by the IT enterprise would be infeasible
This condition, along with hardware underutilization, has led to the diffusion of a technique called server consolidation, for which virtualization technologies are fundamental

- Greening initiatives

Maintaining a data center operation not only involves keeping servers on, but a great deal of energy is also consumed in keeping them cool.

Infrastructures for cooling have a significant impact on the carbon footprint of a data center

Virtualization technologies can provide an efficient way of consolidating servers thereby reducing the power consumption and hence the carbon footprint

- Rise of administrative costs

Power consumption and cooling costs have now become higher than the cost of IT equipment

the higher the number of servers that have to be managed, the higher the administrative costs

Virtualization can help reduce the number of required servers for a given workload, thus reducing the cost of the administrative personnel

Q5 b) What is Hypervisor? With a neat diagram explain the types of hypervisor.

A fundamental element of hardware virtualization is the hypervisor, or virtual machine manager (VMM). It recreates a hardware environment in which guest operating systems are installed. There are two major types of hypervisor: Type I and Type II (see Figure 3.7).

• Type I hypervisors run directly on top of the hardware. Therefore, they take the place of the operating systems and interact directly with the ISA interface exposed by the underlying hardware, and they emulate this interface in order to allow the management of guest operating systems. This type of hypervisor is also called a native virtual machine since it runs natively on hardware

• Type II hypervisors require the support of an operating system to provide virtualization services. This means that they are programs managed by the operating system, which interact with it through the ABI and emulate the ISA of virtual hardware for guest operating systems. This type of hypervisor is also called a hosted virtual machine since it is hosted within an operating system.

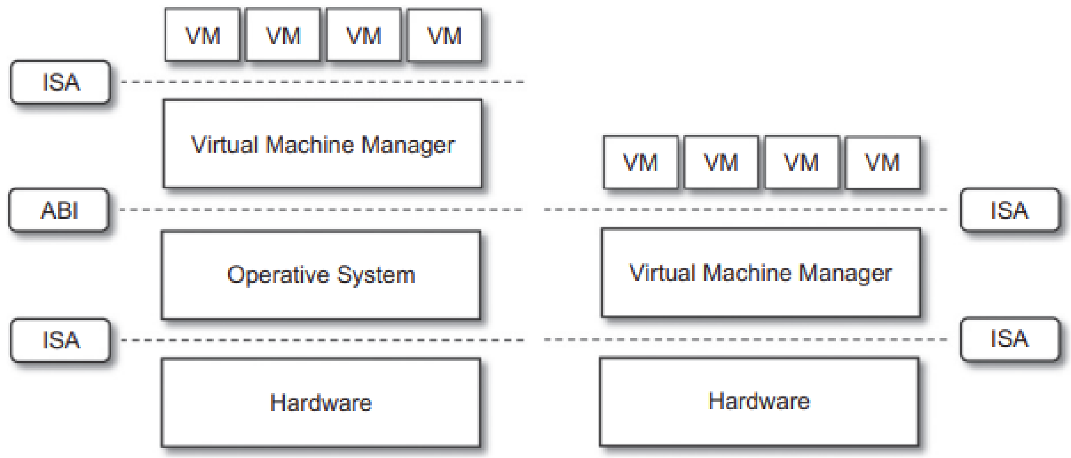


FIGURE 3.7

Hosted (left) and native (right) virtual machines. This figure provides a graphical representation of the two types of hypervisors.

Conceptually, a virtual machine manager is internally organized as described in Figure 3.8. Three main modules, dispatcher, allocator, and interpreter, coordinate their activity in order to emulate the underlying hardware. The dispatcher constitutes the entry point of the monitor and reroutes the instructions issued by the virtual machine instance to one of the two other modules. The allocator is responsible for deciding the system resources to be provided to the VM: whenever a virtual machine tries to execute an instruction that results in changing the machine resources associated with that VM, the allocator is invoked by the dispatcher. The interpreter module consists of interpreter routines. These are executed whenever a virtual machine executes a privileged instruction: a trap is triggered and the corresponding routine is executed. The design and architecture of a virtual machine manager, together with the underlying hardware design of the host machine, determine the full realization of hardware virtualization, where a guest operating system can be transparently executed on top of a VMM as though it were run on the underlying hardware. The criteria that need to be met by a virtual machine manager to efficiently support virtualization were established by Goldberg and Popek in 1974 [23]. Three properties have to be satisfied:

- Equivalence. A guest running under the control of a virtual machine manager should exhibit the same behavior as when it is executed directly on the physical host.
- Resource control. The virtual machine manager should be in complete control of virtualized resources.
- Efficiency. A statistically dominant fraction of the machine instructions should be executed without intervention from the virtual machine manager.

Q 6 a) Explain the different types of hardware virtualization

Hardware-assisted virtualization.

This term refers to a scenario in which the hardware provides architectural support for building a virtual machine manager able to run a guest operating system in complete isolation. This technique was originally introduced in the IBM System/370. At present, examples of hardware-assisted virtualization are the extensions to the x86-64 bit architecture introduced with Intel VT (formerly known as Vanderpool) and AMD V (formerly known as Pacifica). These extensions, which differ between the two vendors, are meant to reduce the performance penalties experienced by emulating x86 hardware with hypervisors. Before the introduction of hardware-assisted virtualization, software emulation of x86 hardware was significantly costly from the performance point of view. The reason for this is that by design the x86 architecture did not meet the formal requirements introduced by Popek and Goldberg, and early products were using binary translation to trap some sensitive instructions and provide an emulated version. Products such as VMware Virtual Platform, introduced in 1999 by VMware, which pioneered the field of x86 virtualization, were based on this technique. After 2006, Intel and AMD introduced processor extensions, and a wide range of virtualization solutions took advantage of them: Kernel-based Virtual Machine (KVM), VirtualBox, Xen, VMware, Hyper-V, Sun xVM, Parallels, and others.

Full virtualization.

Full virtualization refers to the ability to run a program, most likely an operating system, directly on top of a virtual machine and without any modification, as though it were run on the raw hardware. To make this possible, virtual machine managers are required to provide a complete emulation of the entire underlying hardware. The principal advantage of full virtualization is complete isolation, which leads to enhanced security, ease of emulation of different architectures, and coexistence of different systems on the same platform. Whereas it is a desired goal for many virtualization solutions, full virtualization poses important concerns related to performance and technical implementation. A key challenge is the interception of privileged instructions such as I/O instructions: Since they change the state of the resources exposed by the host, they have to be contained within the virtual machine manager. A simple solution to achieve full virtualization is to provide a virtual environment for all the instructions, thus posing some limits on performance. A successful and efficient implementation of full virtualization is obtained with a combination of hardware and software, not allowing potentially harmful instructions to be executed directly on the host. This is what is accomplished through hardware-assisted virtualization.

Paravirtualization.

This is a not-transparent virtualization solution that allows implementing thin virtual machine managers. Paravirtualization techniques expose a software interface to the virtual machine that is slightly modified from the host and, as a consequence, guests need to be modified. The aim of paravirtualization is to provide the capability to demand the execution of performance-critical operations directly on the host, thus preventing performance losses that would otherwise be experienced in managed execution. This allows a simpler implementation of virtual machine managers that have to simply transfer the execution of these operations, which were hard to virtualize, directly to the host.

Partial virtualization.

Partial virtualization provides a partial emulation of the underlying hardware, thus not allowing the complete execution of the guest operating system in complete isolation. Partial virtualization allows many applications to run transparently, but not all the features of the operating system can be supported, as happens with full virtualization. An example of partial virtualization is address space virtualization used in time-sharing systems; this allows multiple applications and users to run concurrently in a separate memory space, but they still share the same hardware resources (disk, processor, and network). Historically, partial virtualization has been an important milestone for achieving full virtualization, and it was implemented on the experimental IBM M44/44X. Address space virtualization is a common feature of contemporary operating systems.

Operating system-level virtualization

Operating system-level virtualization offers the opportunity to create different and separated execution environments for applications that are managed concurrently. Differently from hardware virtualization, there is no virtual machine manager or hypervisor, and the virtualization is done within a single operating system, where the OS kernel allows for multiple isolated user space instances. The kernel is also responsible for sharing the system resources among instances and for limiting the impact of instances on each other. A user space instance in general contains a proper view of the file system, which is completely isolated, and separate IP addresses, software configurations, and access to devices. Operating systems supporting this type of virtualization are general-purpose, timeshared operating systems with the capability to provide stronger namespace and resource isolation

Q6 b) With a neat diagram, explain Xen-Paravirtualization.

Xen is an open-source initiative implementing a virtualization platform based on paravirtualization. Initially developed by a group of researchers at the University of Cambridge in the United Kingdom, Xen now has a large open-source community backing it. Citrix also offers it as a commercial solution, XenSource. Xen-based technology is used for either desktop virtualization or server virtualization, and recently it has also been used to provide cloud computing solutions by means of Xen Cloud Platform (XCP). At the basis of all these solutions is the Xen Hypervisor, which constitutes the core technology of Xen. Recently Xen has been advanced to support full virtualization using hardware-assisted virtualization. Xen is the most popular implementation of paravirtualization, which, in contrast with full virtualization, allows high-performance execution of guest operating systems. This is made possible by eliminating the performance loss while executing instructions that require special management. This is done by modifying portions of the guest operating systems run by Xen with reference to the execution of such instructions. Therefore it is not a transparent solution for implementing virtualization. This is particularly true for x86, which is the most popular architecture on commodity machines and servers.

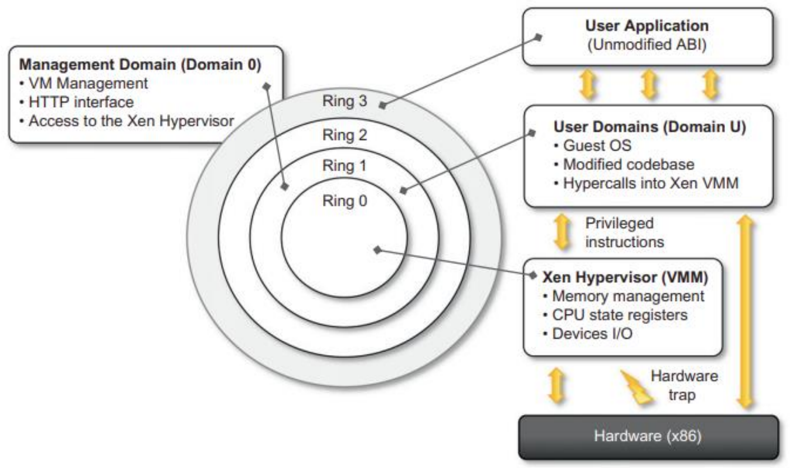


FIGURE 3.11

Xen architecture and guest OS management.

Figure 3.11 describes the architecture of Xen and its mapping onto a classic x86 privilege model. A Xen-based system is managed by the Xen hypervisor, which runs in the highest privileged mode and controls the access of guest operating system to the underlying hardware. Guest operating systems are executed within domains, which represent virtual machine instances. Moreover, specific control software, which has privileged access to the host and controls all the other guest operating systems, is executed in a special domain called Domain 0. This is the first one that is loaded once the virtual machine manager has completely booted, and it hosts a HyperText Transfer Protocol (HTTP) server that serves requests for virtual machine creation, configuration, and termination. This component constitutes the embryonic version of a distributed virtual machine manager, which is an essential component of cloud computing systems providing Infrastructure-as-a-Service (IaaS) solutions.

Popek and Goldberg provided a classification of the instruction set and proposed three theorems that define the properties that hardware instructions need to satisfy in order to efficiently support virtualization

THEOREM 3.1 For any conventional third-generation computer, a VMM may be constructed if the set of sensitive instructions for that computer is a subset of the set of privileged instructions.

THEOREM 3.2 A conventional third-generation computer is recursively virtualizable if:

• It is virtualizable and

• A VMM without any timing dependencies can be constructed for it.

THEOREM 3.3 A hybrid VMM may be constructed for any conventional third-generation machine in which the set of user-sensitive instructions is a subset of the set of privileged instructions.

Q7 a) With a neat diagram, explain infrastructure as a service reference implementation

Infrastructure- and Hardware-as-a-Service (IaaS/HaaS) solutions are the most popular and developed market segment of cloud computing. They deliver customizable infrastructure on demand. The available options within the IaaS offering umbrella range from single servers to entire infrastructures, including network devices, load balancers, and database and Web servers. The main technology used to deliver and implement these solutions is hardware virtualization: one or more virtual machines opportunely configured and interconnected define the distributed system on top of which applications are installed and deployed. Virtual machines also constitute the atomic components that are deployed and priced according to the specific features of the virtual hardware: memory, number of processors, and disk storage. IaaS/HaaS solutions bring all the benefits of hardware virtualization: workload partitioning, application isolation, sandboxing, and hardware tuning.

From the perspective of the service provider, IaaS/HaaS allows better exploiting the IT infrastructure and provides a more secure environment where executing third party applications. From the perspective of the customer it reduces the administration and maintenance cost as well as the capital costs allocated to purchase hardware. At the same time, users can take advantage of the full customization offered by virtualization to deploy their infrastructure in the cloud; in most cases virtual machines come with only the selected operating system installed and the system can be configured with all the required packages and applications. Other solutions provide prepackaged system images that already contain the software stack required for the most common uses: Web servers, database servers, or LAMP1 stacks. Besides the basic virtual machine management capabilities, additional services can be provided, generally including the following: SLA resource-based allocation, workload management, support for infrastructure design through advanced Web interfaces, and the ability to integrate third-party IaaS solutions. Figure 4.2 provides an overall view of the components forming an Infrastructure-as-a-Service solution. It is possible to distinguish three principal layers: the physical infrastructure, the software management infrastructure, and the user interface. At the top layer the user interface provides access to the services exposed by the software management infrastructure. Such an interface is generally based on Web 2.0 technologies: Web services, RESTful APIs, and mash-ups.

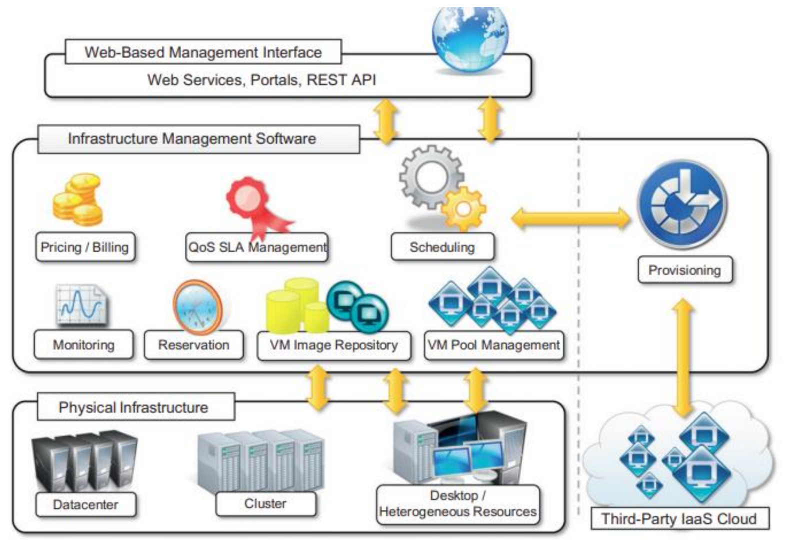


FIGURE 4.2

Infrastructure-as-a-Service reference implementation.

Q7 b) What is the need for private cloud? Discuss the advantages of using private cloud.

Private clouds are virtual distributed systems that rely on a private infrastructure and provide internal users with dynamic provisioning of computing resources. Instead of a pay-as-you-go model as in public clouds, there could be other schemes in place, taking into account the usage of the cloud and proportionally billing the different departments or sections of an enterprise. Private clouds have the advantage of keeping the core business operations in-house by relying on the existing IT infrastructure and reducing the burden of maintaining it once the cloud has been set up. In this scenario, security concerns are less critical, since sensitive information does not flow out of the private infrastructure. Moreover, existing IT resources can be better utilized because the private cloud can provide services to a different range of users. Another interesting opportunity that comes with private clouds is the possibility of testing applications and systems at a comparatively lower price rather than public clouds before deploying them on the public virtual infrastructure. A Forrester report [34] on the benefits of delivering in-house cloud computing solutions for enterprises highlighted some of the key advantages of using a private cloud computing infrastructure:

• Customer information protection. Despite assurances by the public cloud leaders about security, few provide satisfactory disclosure or have long enough histories with their cloud offerings to provide warranties about the specific level of security put in place on their systems. In-house security is easier to maintain and rely on.

• Infrastructure ensuring SLAs. Quality of service implies specific operations such as appropriate clustering and failover, data replication, system monitoring and maintenance, and disaster recovery, and other uptime services can be commensurate to the application needs. Although public cloud vendors provide some of these features, not all of them are available as needed.

• Compliance with standard procedures and operations. If organizations are subject to third-party compliance standards, specific procedures have to be put in place when deploying and executing applications. This could be not possible in the case of the virtual public infrastructure.

All these aspects make the use of cloud-based infrastructures in private premises an interesting option. From an architectural point of view, private clouds can be implemented on more heterogeneous hardware: They generally rely on the existing IT infrastructure already deployed on the private premises. This could be a datacenter, a cluster, an enterprise desktop grid, or a combination of them. The physical layer is complemented with infrastructure management software (i.e., IaaS (M); see Section 4.2.2) or a PaaS solution, according to the service delivered to the users of the cloud.

Q8 a) Explain the sectors where community clouds are used?

Candidate sectors for community clouds are as follows:

• Media industry. In the media industry, companies are looking for low-cost, agile, and simple solutions to improve the efficiency of content production. Most media productions involve an extended ecosystem of partners. In particular, the creation of digital content is the outcome of a collaborative process that includes movement of large data, massive compute-intensive rendering tasks, and complex workflow executions. Community clouds can provide a shared environment where services can facilitate business-to-business collaboration and offer the horsepower in terms of aggregate bandwidth, CPU, and storage required to efficiently support media production.

• Healthcare industry. In the healthcare industry, there are different scenarios in which community clouds could be of use. In particular, community clouds can provide a global platform on which to share information and knowledge without revealing sensitive data maintained within the private infrastructure. The naturally hybrid deployment model of community clouds can easily support the storing of patient-related data in a private cloud while using the shared infrastructure for noncritical services and automating processes within hospitals.

• Energy and other core industries. In these sectors, community clouds can bundle the comprehensive set of solutions that together vertically address management, deployment, and orchestration of services and operations. Since these industries involve different providers, vendors, and organizations, a community cloud can provide the right type of infrastructure to create an open and fair market.

• Public sector. Legal and political restrictions in the public sector can limit the adoption of public cloud offerings. Moreover, governmental processes involve several institutions and agencies and are aimed at providing strategic solutions at local, national, and international administrative levels. They involve business-to-administration, citizen-to-administration, and possibly business-to-business processes. Some examples include invoice approval, infrastructure planning, and public hearings. A community cloud can constitute the optimal venue to provide a distributed environment in which to create a communication platform for performing such operations.

• Scientific research. Science clouds are an interesting example of community clouds. In this case, the common interest driving different organizations sharing a large distributed infrastructure is scientific computing.

Q8 b) Explain the challenges in cloud computing?

Cloud definition

As discussed earlier, there have been several attempts made to define cloud computing and to provide a classification of all the services and technologies identified as such. One of the most comprehensive formalizations is noted in the NIST working definition of cloud computing. It characterizes cloud computing as on-demand self-service, broad network access, resource-pooling, rapid elasticity, and measured service; classifies services as SaaS, PaaS, and IaaS; and categorizes deployment models as public, private, community, and hybrid clouds. The view is in line with our discussion and shared by many IT practitioners and academics.

Cloud interoperability and standards

Cloud computing is a service-based model for delivering IT infrastructure and applications like utilities such as power, water, and electricity. To fully realize this goal, introducing standards and allowing interoperability between solutions offered by different vendors are objectives of fundamental importance. Vendor lock-in constitutes one of the major strategic barriers against the seamless adoption of cloud computing at all stages. In particular there is major fear on the part of enterprises in which IT constitutes the significant part of their revenues. Vendor lock-in can prevent a customer from switching to another competitor's solution, or when this is possible, it happens at considerable conversion cost and requires significant amounts of time. This can occur either because the customer wants to find a more suitable solution for customer needs or because the vendor is no longer able to provide the required service. The presence of standards that are actually implemented and adopted in the cloud computing community could give room for interoperability and then lessen the risks resulting from vendor lock-in.

Scalability and fault tolerance

The ability to scale on demand constitutes one of the most attractive features of cloud computing. Clouds allow scaling beyond the limits of the existing in-house IT resources, whether they are infrastructure (compute and storage) or applications services. To implement such a capability, the cloud middleware has to be designed with the principle of scalability along different dimensions in mind—for example, performance, size, and load. The cloud middleware manages a huge number of resource and users, which rely on the cloud to obtain the horsepower that they cannot obtain within the premises without bearing considerable administrative and maintenance costs. These costs are a reality for whomever develops, manages, and maintains the cloud middleware and offers the service to customers. In this scenario, the ability to tolerate failure becomes fundamental, sometimes even more important than providing an extremely efficient and optimized system. Hence, the challenge in this case is designing highly scalable and fault-tolerant systems that are easy to manage and at the same time provide competitive performance.

Security, trust, and privacy

Security, trust, and privacy issues are major obstacles for massive adoption of cloud computing. The traditional cryptographic technologies are used to prevent data tampering and access to sensitive information. The massive use of virtualization technologies exposes the existing system to new threats, which previously were not considered applicable. For example, it might be possible that applications hosted in the cloud can process sensitive information; such information can be stored within a cloud storage facility using the most advanced technology in cryptography to protect data and then be considered safe from any attempt to access it without the required permissions. Although these data are processed in memory, they must necessarily be decrypted by the legitimate application, but since the application is hosted in a managed virtual environment it becomes accessible to the virtual machine manager that by program is designed to access the memory pages of such an application. In this case, what is experienced is a lack of control over the environment in which the application is executed, which is made possible by leveraging the cloud. It then happens that a new way of using existing technologies creates new opportunities for additional threats to the security of applications. The lack of control over their own data and processes also poses severe problems for the trust we give to the cloud service provider and the level of privacy we want to have for our data.

Q9 a) What is open stack? Discuss the important components of open stack.

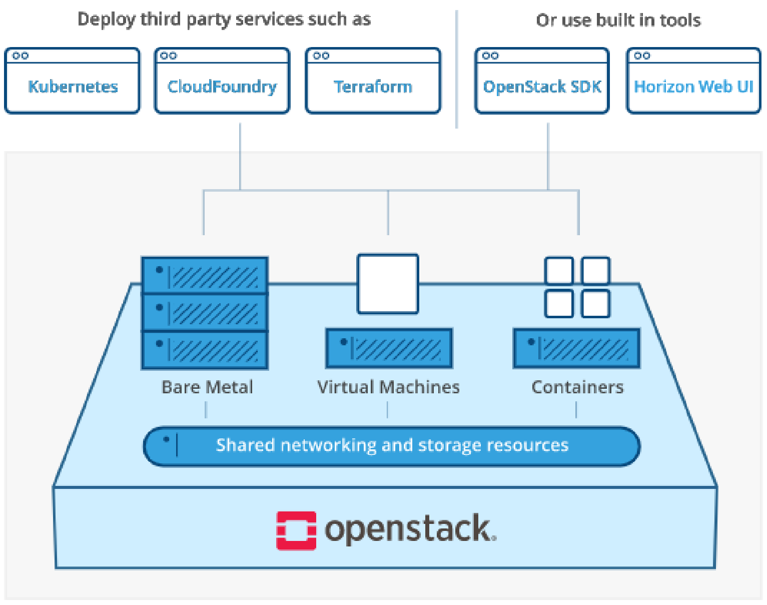
OpenStack is a free open standard cloud computing platform, mostly deployed as infrastructure-as-a-service (IaaS) in both public and private clouds where virtual servers and other resources are made available to users.

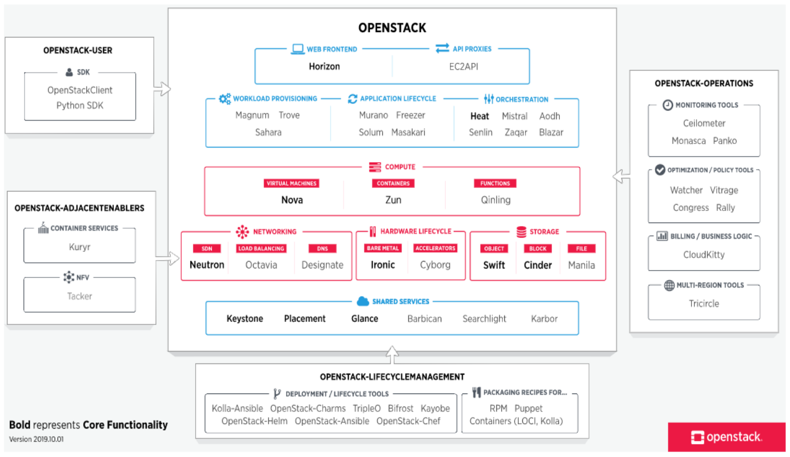
The software platform consists of interrelated components that control diverse, multi-vendor hardware pools of processing, storage and networking resources throughout a data center.

Users manage it either through a web-based dashboard, through command-line tools or through RESTful web services.

Developed by Open Infrastructure Foundation and community.

Developed in Python and released in 2010.





Q9 b) Explain briefly the storage devices of Amazon Web services.

Amazon Simple Storage Service (S3)

As the name suggests, S3 has been designed to provide a simple storage service that's accessible through a Representational State Transfer (REST) interface, which is quite similar to a distributed file system but which presents some important differences that allow the infrastructure to be highly efficient:

• The storage is organized in a two-level hierarchy. S3 organizes its storage space into buckets that cannot be further partitioned. This means that it is not possible to create directories or other kinds of physical groupings for objects stored in a bucket. Despite this fact, there are few limitations in naming objects, and this allows users to simulate directories and create logical groupings.

• Stored objects cannot be manipulated like standard files. S3 has been designed to essentially provide storage for objects that will not change over time. Therefore, it does not allow renaming, modifying, or relocating an object. Once an object has been added to a bucket, its content and position is immutable, and the only way to change it is to remove the object from the store and add it again.

• Content is not immediately available to users. The main design goal of S3 is to provide an eventually consistent data store. As a result, because it is a large distributed storage facility, changes are not immediately reflected. For instance, S3 uses replication to provide redundancy and efficiently serve objects across the globe; this practice introduces latencies when adding objects to the store—especially large ones—which are not available instantly across the entire globe.

• Requests will occasionally fail. Due to the large distributed infrastructure being managed, requests for object may occasionally fail. Under certain conditions, S3 can decide to drop a request by returning an internal server error. Therefore, it is expected to have a small failure rate during day-to-day operations, which is generally not identified as a persistent failure.

Access to S3 is provided with RESTful Web services. These express all the operations that can be performed on the storage in the form of HTTP requests (GET, PUT, DELETE, HEAD, and POST), which operate differently according to the element they address. As a rule of thumb PUT/ POST requests add new content to the store, GET/HEAD requests are used to retrieve content and information, and DELETE requests are used to remove elements or information attached to them.

Amazon elastic block store

The Amazon Elastic Block Store (EBS) allows AWS users to provide EC2 instances with persistent storage in the form of volumes that can be mounted at instance startup. They accommodate up to 1 TB of space and are accessed through a block device interface, thus allowing users to format them according to the needs of the instance they are connected to (raw storage, file system, or other).

The content of an EBS volume survives the instance life cycle and is persisted into S3. EBS volumes can be cloned, used as boot partitions, and constitute durable storage since they rely on S3 and it is possible to take incremental snapshots of their content. EBS volumes normally reside within the same availability zone of the EC2 instances that will use them to maximize the I/O performance. It is also possible to connect volumes located in different availability zones. Once mounted as volumes, their content is lazily loaded in the background and according to the request made by the operating system. This reduces the number of I/O requests that go to the network. Volume images cannot be shared among instances, but multiple (separate) active volumes can be created from them. In addition, it is possible to attach multiple volumes to a single instance or create a volume from a given snapshot and modify its size, if the formatted file system allows such an operation.

Amazon ElastiCache

ElastiCache is an implementation of an elastic in-memory cache based on a cluster of EC2 instances. It provides fast data access from other EC2 instances through a Memcached-compatible protocol so that existing applications based on such technology do not need to be modified and can transparently migrate to ElastiCache. ElastiCache is based on a cluster of EC2 instances running the caching software, which is made available through Web services. An ElastiCache cluster can be dynamically resized according to the demand of the client applications. Furthermore, automatic patch management and failure detection and recovery of cache nodes allow the cache cluster to keep running without administrative intervention from AWS users, who have only to elastically size the cluster when needed.

Structured storage solutions

Traditionally, RDBMS have been the common data back-end for a wide range of applications, even though recently more scalable and lightweight solutions have been proposed. Amazon provides applications with structured storage services in three different forms: preconfigured EC2 AMIs, Amazon Relational Data Storage (RDS), and Amazon SimpleDB.

Q10 a) Explain ay two scientific applications where cloud computing can be used

ECG monitoring

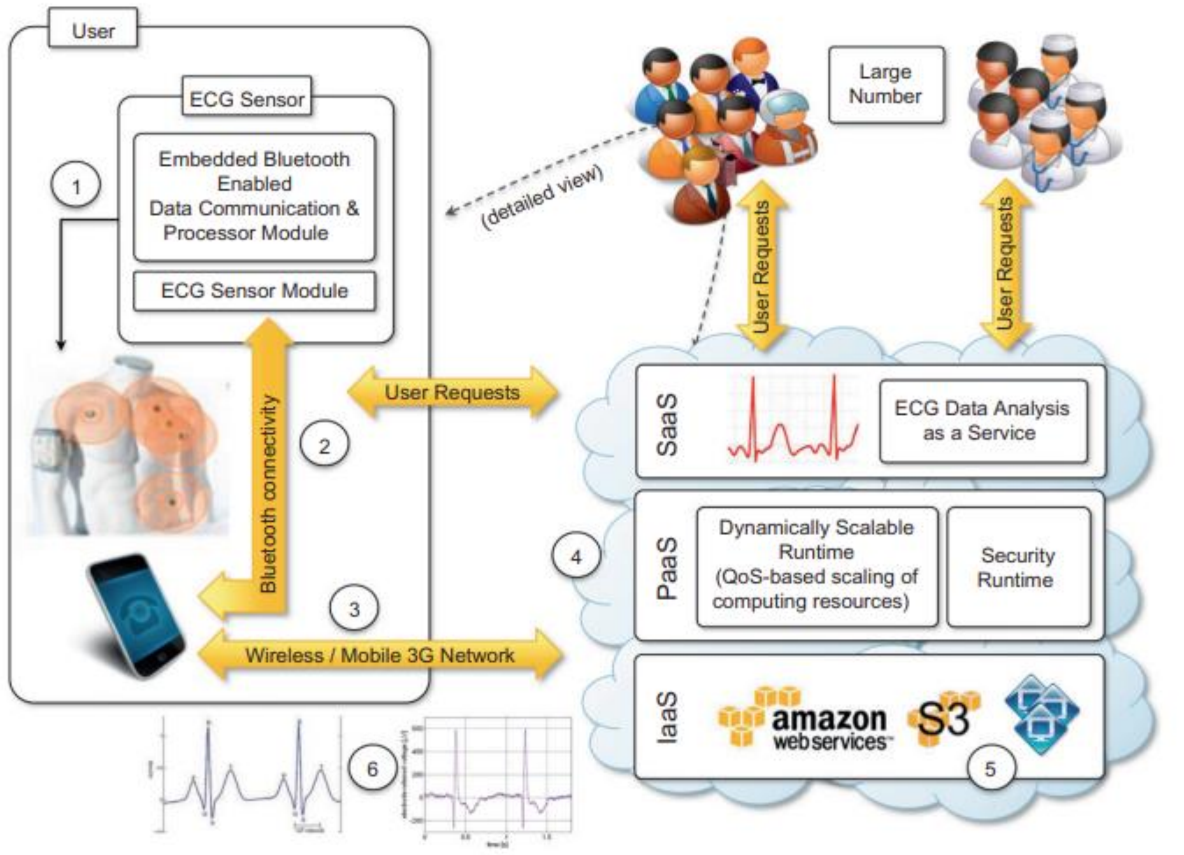

User
ECG Sensor
1
Embedded Bluetooth Enabled Data Communication & Processor Module
ECG Sensor Module
(detailed view)
Large Number
User Requests
User Requests
Bluetooth connectivity
2
User Requests
SaaS
ECG Data Analysis as a Service
4
PaaS
Dynamically Scalable Runtime (QoS-based scaling of computing resources)
Security Runtime
3
Wireless / Mobile 3G Network
IaaS
amazon webservices
S3
5
6

FIGURE 10.1

An online health monitoring system hosted in the cloud.

The capillary development of Internet connectivity and its accessibility from any device at any time has made cloud technologies an attractive option for developing health-monitoring systems. ECG data analysis and monitoring constitute a case that naturally fits into this scenario. ECG is the electrical manifestation of the contractile activity of the heart's myocardium. This activity produces a specific waveform that is repeated over time and that represents the heartbeat. The analysis of the shape of the ECG waveform is used to identify arrhythmias and is the most common way to detect heart disease. Cloud computing technologies allow the remote monitoring of a patient's heartbeat data, data analysis in minimal time, and the notification of first-aid personnel and doctors should these data reveal potentially dangerous conditions. This way a patient at risk can be constantly monitored without going to a hospital for ECG analysis. At the same time, doctors and first-aid personnel can instantly be notified of cases that require their attention.

An illustration of the infrastructure and model for supporting remote ECG monitoring is shown in Figure 10.1. Wearable computing devices equipped with ECG sensors constantly monitor the patient's heartbeat. Such information is transmitted to the patient's mobile device, which will eventually forward it to the cloud-hosted Web service for analysis. The Web service forms the front-end of a platform that is entirely hosted in the cloud and that leverages the three layers of the cloud computing stack: SaaS, PaaS, and IaaS. The Web service constitute the SaaS application that will store ECG data in the Amazon S3 service and issue a processing request to the scalable cloud platform. The runtime platform is composed of a dynamically sizable number of instances running the workflow engine and Aneka. The number of workflow engine instances is controlled according to the number of requests in the queue of each instance, while Aneka controls the number of EC2 instances used to execute the single tasks defined by the workflow engine for a single ECG processing job. Each of these jobs consists of a set of operations involving the extraction of the waveform from the heartbeat data and the comparison of the waveform with a reference waveform to detect anomalies. If anomalies are found, doctors and first-aid personnel can be notified to act on a specific patient.

Even though remote ECG monitoring does not necessarily require cloud technologies, cloud computing introduces opportunities that would be otherwise hardly achievable. The first advantage is the elasticity of the cloud infrastructure that can grow and shrink according to the requests served. As a result, doctors and hospitals do not have to invest in large computing infrastructures designed after capacity planning, thus making more effective use of budgets. The second advantage is ubiquity. Cloud computing technologies have now become easily accessible and promise to deliver systems with minimum or no downtime. Computing systems hosted in the cloud are accessible from any Internet device through simple interfaces (such as SOAP and REST-based Web services). This makes these systems not only ubiquitous, but they can also be easily integrated with other systems maintained on the hospital's premises.

Finally, cost savings constitute another reason for the use of cloud technology in healthcare. Cloud services are priced on a pay-per-use basis and with volume prices for large numbers of service requests.

These two models provide a set of flexible options that can be used to price the service, thus actually charging costs based on effective use rather than capital costs.

Jeeva Portal

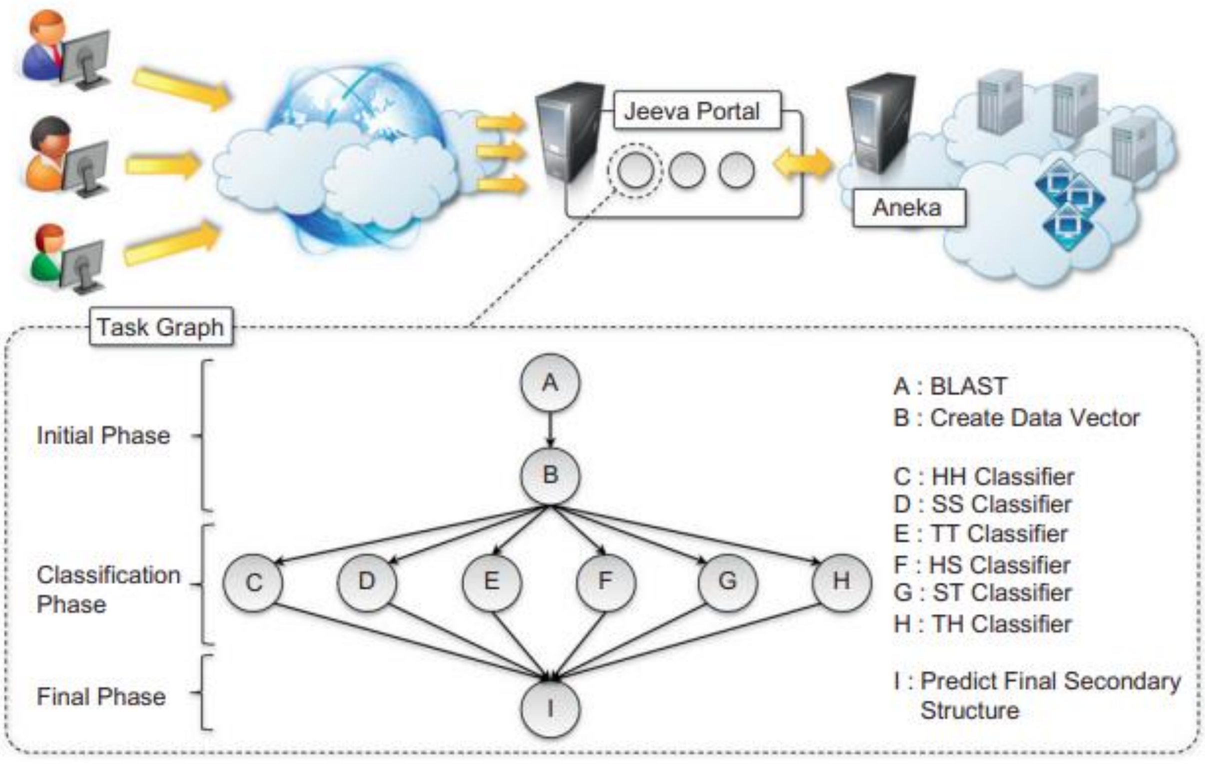


FIGURE 10.2

Architecture and overview of the Jeeva Portal.

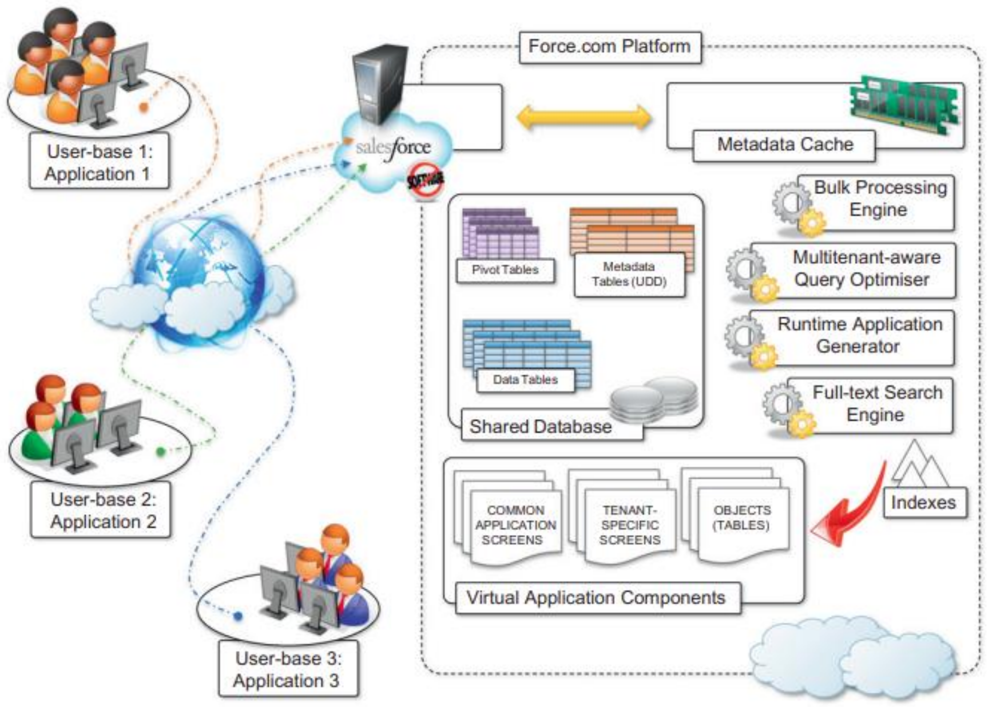
One project that investigates the use of cloud technologies for protein structure prediction is Jeeva—an integrated Web portal that enables scientists to offload the prediction task to a computing cloud based on Aneka (see Figure 10.2). The prediction task uses machine learning techniques (support vector machines) for determining the secondary structure of proteins. These techniques translate the problem into one of pattern recognition, where a sequence has to be classified into one of three possible classes (E, H, and C). A popular implementation based on support vector machines divides the pattern recognition problem into three phases: initialization, classification, and a final phase. Even though these three phases have to be executed in sequence, it is possible to take advantage of parallel execution in the classification phase, where multiple classifiers are executed concurrently.

This creates the opportunity to sensibly reduce the computational time of the prediction. The prediction algorithm is then translated into a task graph that is submitted to Aneka. Once the task is completed, the middleware makes the results available for visualization through the portal. The advantage of using cloud technologies (i.e., Aneka as scalable cloud middleware) versus conventional grid infrastructures is the capability to leverage a scalable computing infrastructure that can be grown and shrunk on demand. This concept is distinctive of cloud technologies and constitutes a strategic advantage when applications are offered and delivered as a service.

Q10 b) Explain in detail business and consumers application of cloud computing

<u>CRM and ERP</u>

salesforce and Force.com architecture



Force.com Platform
User-base 1: Application 1
User-base 2: Application 2
User-base 3: Application 3
salesforce
Metadata Cache
Bulk Processing Engine
Multitenant-aware Query Optimiser
Runtime Application Generator
Full-text Search Engine
Pivot Tables
Metadata Tables (UDD)
Data Tables
Shared Database
COMMON APPLICATION SCREENS
TENANT-SPECIFIC SCREENS
OBJECTS (TABLES)
Indexes
Virtual Application Components

FIGURE 10.5

Salesforce.com and Force.com architecture.

Salesforce.com is probably the most popular and developed CRM solution available today. As of today more than 100,000 customers have chosen Safesforce.com to implement their CRM solutions. The application provides customizable CRM solutions that can be integrated with additional features developed by third parties. Salesforce.com is based on the Force.com cloud development platform. This represents scalable and high-performance middleware executing all the operations of all Salesforce.com applications. The architecture of the Force.com platform is shown in Figure 10.5. Initially designed to support scalable CRM applications, the platform has evolved to support the entire life cycle of a wider range of cloud applications by implementing a flexible and scalable infrastructure. At the core of the platform resides its metadata architecture, which provides the system with flexibility and scalability. Rather than being built on top of specific components and tables, application core logic and business rules are saved as metadata into the Force.com store. Both application structure and application data are stored in the store.

Productivity

Dropbox and Icloud

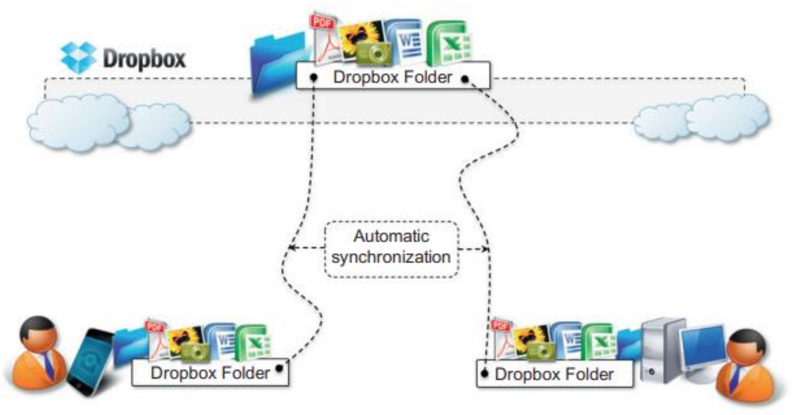


FIGURE 10.6

Dropbox usage scenario.

The most popular solution for online document storage is Dropbox, an online application that allows users to synchronize any file across any platform and any device in a seamless manner (see Figure 10.6). Dropbox provides users with a free amount of storage that is accessible through the abstraction of a folder. Users can either access their Dropbox folder through a browser or by downloading and installing a Dropbox client, which provides access to the online storage by means of a special folder. All the modifications into this folder are silently synched so that changes are notified to all the local instances of the Dropbox folder across all the devices.

Google Docs

Google Docs is a SaaS application that delivers the basic office automation capabilities with support for collaborative editing over the Web. The application is executed on top of the Google distributed computing infrastructure, which allows the system to dynamically scale according to the number of users using the service. Google Docs allows users to create and edit text documents, spreadsheets, presentations, forms, and drawings. It aims to replace desktop products such as Microsoft Office and OpenOffice and provide similar interface and functionality as a cloud service. It supports collaborative editing over the Web for most of the applications included in the suite. This eliminates tedious emailing and synchronization tasks when documents need to be edited by multiple users. By being stored in the Google infrastructure, these documents are always available from anywhere and from any device that is connected to the Internet.

Cloud Desktops EyeOS and XIOS/3

EyeOS1 is one of the most popular Web desktop solutions based on cloud technologies. It replicates the functionalities of a classic desktop environment and comes with pre-installed applications for the most common file and document management tasks (see Figure 10.7). Single users can access the EyeOS desktop environment from anywhere and through any Internet-connected device, whereas organizations can create a private EyeOS Cloud on their premises to virtualize the desktop environment of their employees and centralize their management.
Xcerion XML Internet OS/3 (XIOS/3) is another example of a Web desktop environment. The service is delivered as part of the CloudMe application, which is a solution for cloud document storage. The key differentiator of XIOS/3 is its strong leverage of XML, used to implement many of the tasks of the OS: rendering user interfaces, defining application business logics, structuring file system organization, and even application development.The architecture of the OS concentrates most of the functionalities on the client side while implementing server-based functionalities by means of XML Web services. The client side renders the user interface, orchestrates processes, and provides data-binding capabilities on XML data that is exchanged with Web services.

Social Networking

Facebook is probably the most evident and interesting environment in social networking. With more than 800 million users, it has become one of the largest Websites in the world. To sustain this incredible growth, it has been fundamental that Facebook be capable of continuously adding capacity and developing new scalable technologies and software systems while maintaining high performance to ensure a smooth user experience. Currently, the social network is backed by two data centers that have been built and optimized to reduce costs and impact on the environment. On top of this highly efficient infrastructure, built and designed out of inexpensive hardware, a completely customized stack of opportunely modified and refined open-source technologies constitutes the back-end of the largest social network.