Unit II

Cloud Computing Architecture, Introduction, Cloud Reference Model, Architecture, Infrastructure / Hardware as a Service, Platform as a Service, Software as a Service, Types of Clouds, Public Clouds, Private Clouds, Hybrid Clouds, Community Clouds, Economics of the Cloud, Open Challenges, Cloud Definition, Cloud Interoperability and Standards Scalability and Fault ToleranceSecurity, Trust, and Privacy Organizational Aspects, Principles of Parallel and Distributed Computing- Elements of parallel computing, Elements of distributed computing, Technologies for distributed computing

Cloud Computing Architecture

1. It is possible to organize all the concrete realizations of cloud computing into a layered view covering the entire stack from hardware appliances to software systems.(Below figure)
2. Cloud resources are harnessed to offer "computing horsepower" required for providing services.
3. Often, this layer is implemented using a datacenter in which hundreds and thousands of nodes are stacked together.
4. Cloud infrastructure can be heterogeneous in nature because a variety of resources, such as clusters and even networked PCs, can be used to build it. Moreover, database systems and other storage services can also be part of the infrastructure.
5. The physical infrastructure is managed by the core middleware, the objectives of which are to provide an appropriate runtime environment for applications and to best utilize resources.

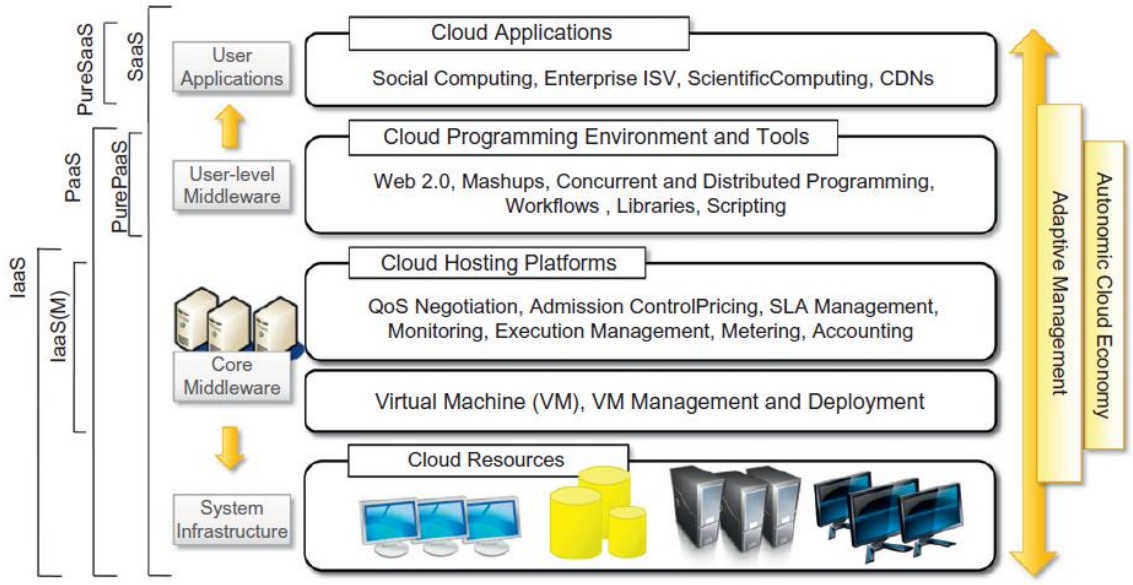


FIGURE 4.1

The cloud computing architecture.

6. At the bottom of the stack, virtualization technologies are used to guarantee runtime environment customization, application isolation, sandboxing, and quality of service.
7. Hardware virtualization is most commonly used at this level. Hypervisors manage the pool of resources and expose the distributed infrastructure as a collection of virtual machines. By using virtual machine technology it is possible to finely partition the

hardware resources such as CPU and memory and to virtualized specific devices, thus meeting the requirements of users and applications.

8. This solution is generally paired with storage and network virtualization strategies, which allow the infrastructure to be completely virtualized and controlled.
9. According to the specific service offered to end users, other virtualization techniques can be used; for example, programming-level virtualization helps in creating a portable runtime environment where applications can be run and controlled.

The combination of cloud hosting platforms and resources is generally classified as an Infrastructure-as-a-Service (IaaS) solution. We can organize the different examples of IaaS into two categories: Some of them provide both the management layer and the physical infrastructure; others provide only the management layer (IaaS (M)).
IaaS solutions are suitable for designing the system infrastructure but provide limited services to build applications. Such service is provided by cloud programming environment sand tools, which form a new layer for offering users a development platform for applications. The range of tools includes Web-based interfaces, command line tools, and frame works for concurrent and distributed programming. In this scenario, users develop their applications specifically for the cloud by using the API exposed at the user-level middleware. For this reason, this approach is also known as Platform-as-a-Service (PaaS) because the service offered to the user is a development platform rather than an infrastructure. PaaS solutions generally include the infrastructure as well, which is bundled as part of the service provided to users.

The top layer of the reference model depicted in contains services delivered at the application level.

The reference model also introduces the concept of everything as a Service (XaaS). This is one of the most important elements of cloud computing Cloud services from different providers can be combined to provide a completely integrated solution covering all the computing stack of a system. IaaS providers can offer the bare metal in terms of virtual machines where PaaS solutions are deployed.

==========

The 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)
Software-as-a-Service (SaaS)

These categories are related to each other as described in Figure 1.5, which provides an organic view of cloud computing. We refer to this diagram as the Cloud Computing Reference Model

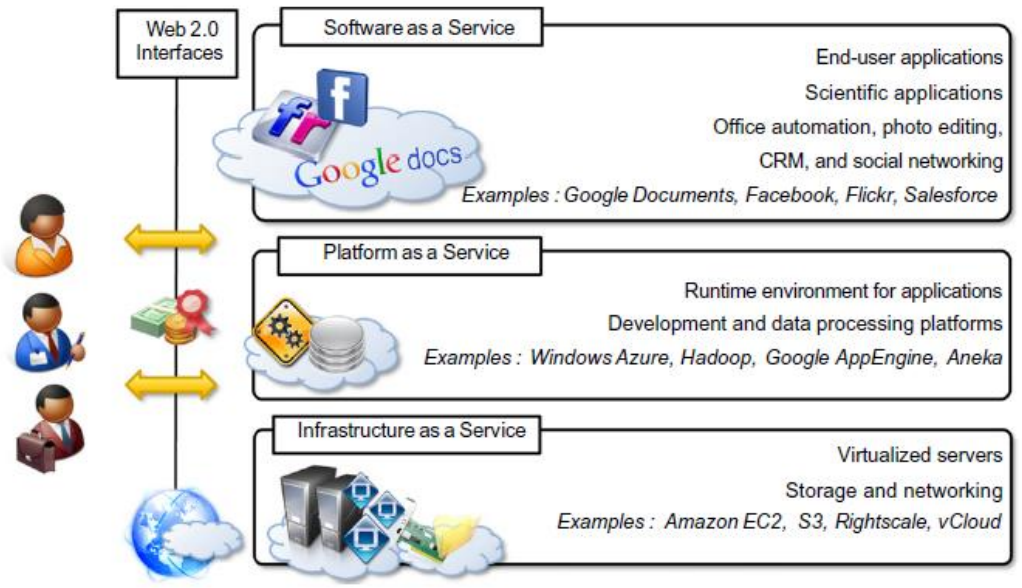


FIGURE 1.5

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.
- The former complements a virtual hardware offering that requires persistent storage.
- The latter is a more high-level abstraction for storing enti- ties rather than files.
- Virtual networking identifies the collection of services that manage the net- working 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. PaaS solutions provide scalable programming platforms for developing applications and are more appropriate when new systems have to be developed. SaaS solutions tar- get mostly end users who want to benefit from the elastic scalability of the cloud without doing any software development, installation, configuration, and maintenance.

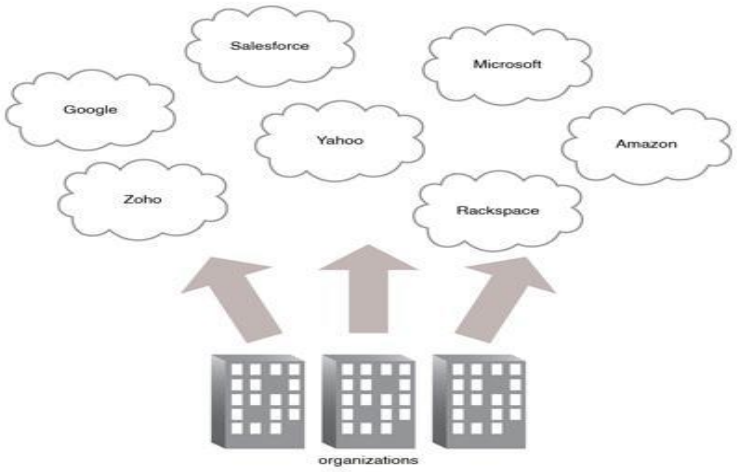
==========

CLOUD DEPLOYMENT MODELS

Enterprises can choose to deploy applications on Public, Private or Hybrid clouds. Cloud Integrators can play a vital part in determining the right cloud path for each organization.

Public Cloud Public clouds are owned and operated by third parties; they deliver superior economies of scale to customers, as the infrastructure costs are spread among a mix of users, giving each individual client an attractive low-cost, "Pay-as-you-go" model. All customers share the same infrastructure pool with limited configuration, security protections, and availability variances. These are managed and supported by the cloud provider. One of the advantages of a Public cloud is that they may be larger than an enterprises cloud, thus providing the ability to scale seamlessly, on demand.

Figure shows a partial view of the public cloud landscape, highlighting some of the primary vendors in the marketplace.



Private Cloud

Private clouds are built exclusively for a single enterprise. They aim to address concerns on data security and offer greater control, which is typically lacking in a public cloud. There are two variations to a private cloud:

- **On-premise Private Cloud:** On-premise private clouds, also known as internal clouds are hosted within one‟s own data center. This model provides a more standardized process and protection, but is limited in aspects of size and scalability. IT departments would also need to incur the capital and operational costs for the physical resources. This is best suited for applications which require complete control and configurability of the infrastructure and security.

- **Externally hosted Private Cloud:** This type of private cloud is hosted externally with a cloud provider, where the provider facilitates an exclusive cloud environment with full guarantee of privacy. This is best suited for enterprises that don‟t prefer a public cloud due to sharing of physical resources.

Hybrid Cloud

Hybrid Clouds combine both public and private cloud models. With a Hybrid Cloud, service providers can utilize 3rd party Cloud Providers in a full or partial manner thus increasing the flexibility of computing. The Hybrid cloud environment is capable of providing on-demand, externally provisioned scale. The ability to augment a private cloud with the resources of a public cloud can be used to manage any unexpected surges in workload.

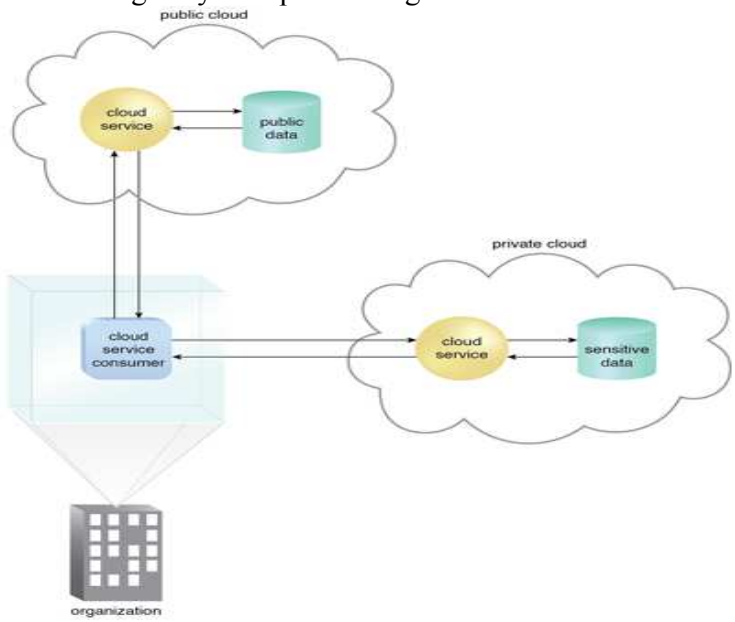


Figure - An organization using a hybrid cloud architecture that utilizes both a private and public cloud.

Community cloud

The cloud infrastructure in a community cloud is shared by several organizations which have shared concerns (e.g. security requirements, policy, and compliance considerations). It is generally managed by the organizations in the community or a third party and can be present either on-premises or off-premises1. Advantages of Community clouds are that they are secure than public clouds and sharing of resources among several organizations. Disadvantages are that it is less secure than private cloud and requires governing policies for administration.

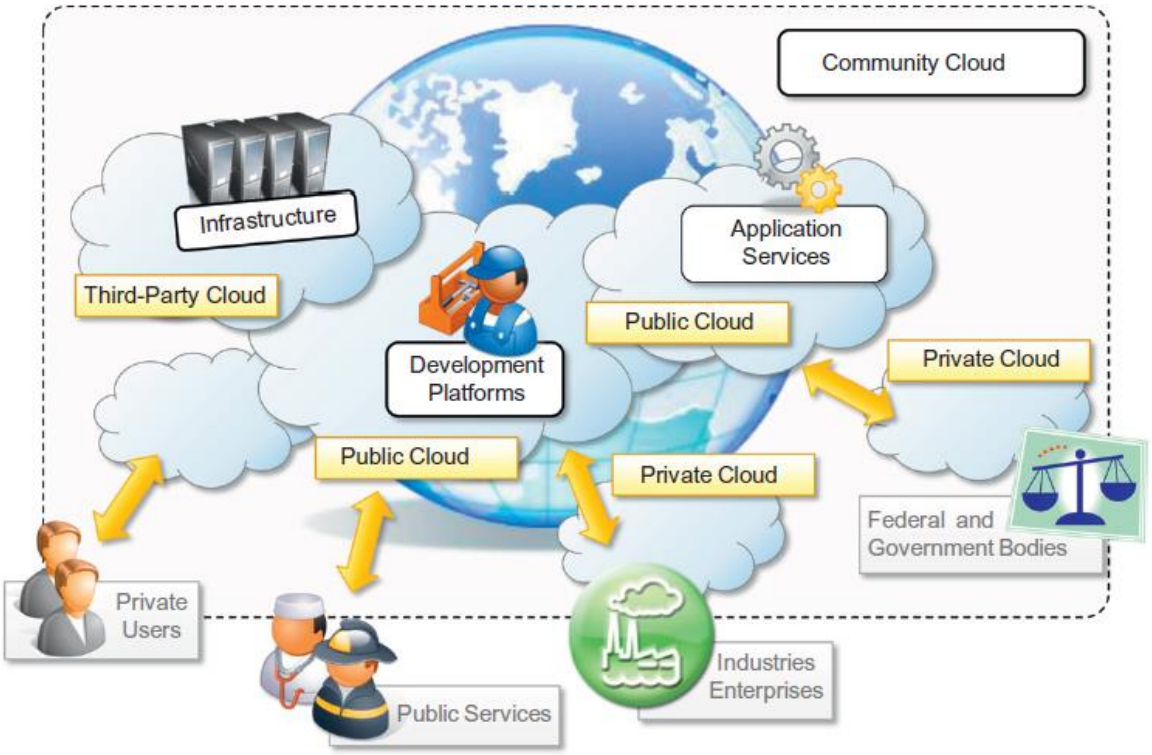


FIGURE 4.6

A community cloud.

Figure 4.6 provides a general view of the usage scenario of community clouds, together with reference architecture.

The users of a specific community cloud fall into a well-identified community, sharing the same concerns or needs; they can be government bodies, industries, or even simple users, but all of them focus on the same issues for their interaction with the cloud.

This is a different scenario than public clouds, which serve a multitude of users with different needs.

Community clouds are also different from private clouds, where the services are generally delivered within the institution that owns the cloud.

==========

CLOUD SERVICE MODELS

Software as a Service (SaaS). The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings.

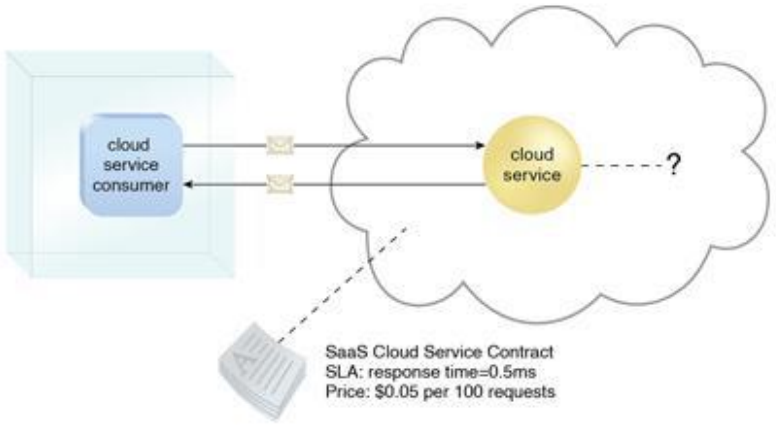


Figure The cloud service consumer is given access the cloud service contract, but not to any underlying IT resources or implementation details.

Platform as a Service (PaaS). The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider.The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment.

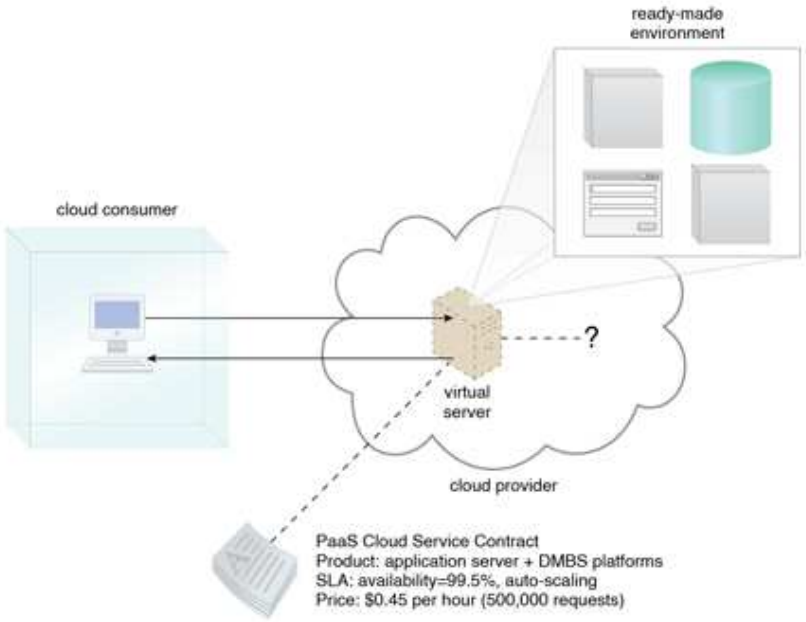


Figure - A cloud consumer is accessing a ready-made PaaS environment. The question mark indicates that the cloud consumer is intentionally shielded from the implementation details of the platform.

Infrastructure as a Service (IaaS). The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls).

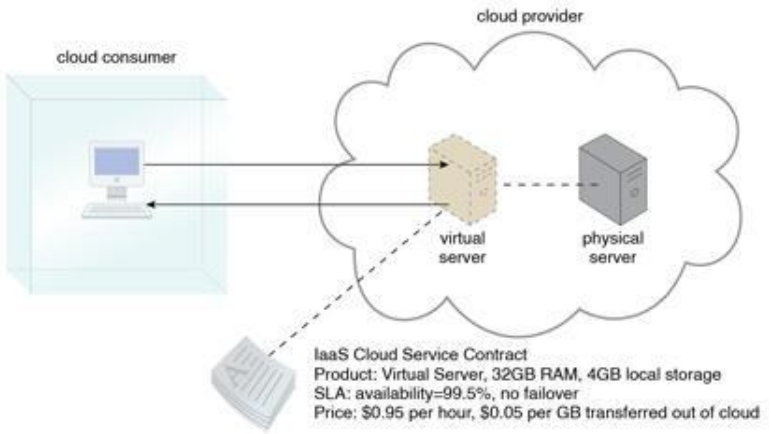
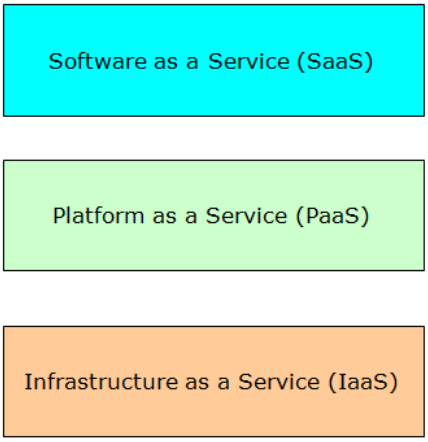


Figure - A cloud consumer is using a virtual server within an IaaS environment. Cloud consumers are provided with a range of contractual guarantees by the cloud provider, pertaining to characteristics such as capacity, performance, and availability.

- Infrastructure as a Service (IaaS)
  - CPU, Storage: Amazon.com, Nirvanix, GoGrid….
- Platform as a Service (PaaS)
  - Google App Engine, Microsoft Azure, Manjrasoft Aneka..
- Software as a Service (SaaS)
  - SalesForce.Com



Economics of the Cloud

The main drivers of cloud computing are economy of scale and simplicity of software delivery and its operation. In fact, the biggest benefit of this phenomenon is financial: the pay-as-you-go model offered by cloud providers. In particular, cloud computing allows:
1.Reducing the capital costs associated to the IT infrastructure
2.Eliminating the depreciation or lifetime costs associated with IT capital assets
3.Replacing software licensing with subscriptions
4.Cutting the maintenance and administrative costs of IT resources

A capital cost is the cost occurred in purchasing an asset that is useful in the production of goods or the rendering of services. Capital costs are one-time expenses that are generally paid up front and that will contribute over the long term to generate profit.IT resources constitute a capital cost for any kind of enterprise. It is good practice to try to keep capital costs low because

they introduce expenses that will generate profit over time; more than that, since they are associated with material things they are subject to depreciation over time, which in the end reduces the profit of the enterprise because such costs are directly subtracted from the enterprise revenues. One of the advantages introduced by the cloud computing model is that it shifts the capital costs previously allocated to the purchase of hardware and software into operational costs inducted by renting the infrastructure and paying subscriptions for the use of software. These costs can be better controlled according to the business needs and prosperity of the enterprise. Cloud computing also introduces reductions in administrative and maintenance costs. That is, there is no or limited need for having administrative staff take care of the management of the cloud infrastructure.

In terms of the pricing models introduced by cloud computing, we can distinguish three different strategies that are adopted by the providers:

1.Tiered pricing. In this model, cloud services are offered in several tiers, each of which offersafixed computing specification and SLA at a specific price per unit of time. This model is used by Amazon for pricing the EC2 service.

2.Per-unit pricing. This model is more suitable to cases where the principal source of revenue forthe cloud provider is determined in terms of units of specific services, such as data transfer and memory allocation. In this scenario customers can configure their systems more efficiently according to the application needs. This model is used, for example, by GoGrid, which makes customers pay according to RAM/hour units for the servers deployed in the GoGrid cloud.

3.Subscription-based pricing. This is the model used mostly by SaaS providers in which userspay a periodic subscription fee for use of the software or the specific component services that are integrated in their applications

Open Challenges

Cloud computing presents many challenges for industry and academia. There is a significant amount of work in academia focused on defining the challenges brought by this phenomenon.

In this section, we highlight the most important ones.

1 Cloud definition
2 Cloud interoperability and standards
3 Scalability and fault tolerance
4 Security, trust, and privacy
5 Organizational aspects

1 Cloud definition

There have been several attempts made to define cloud computing and to provide a classification of all the services and technologies identified as such.

NSIT 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.

Alternative taxonomies for cloud services. David Linthicum, founder of BlueMountains Labs, provides a more detailed classification, which comprehends 10 different classes and better suits the vision of cloud computing within the enterprise.

These characterizations and taxonomies reflect what is meant by cloud computing at the present time, but being in its infancy the phenomenon is constantly evolving, and the same will happen to the attempts to capture the real nature of cloud computing.

2 Cloud interoperability and standards

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. 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.

The first steps toward a standardization process have been made, and a few organizations, such as the Cloud Computing Interoperability Forum (CCIF), the Open Cloud Consortium, and the DMTF Cloud Standards Incubator, are leading the path. Another interesting initiative is the Open Cloud Manifesto, which embodies the point of view of various stakeholders on the benefits of open standards in the field.

The Open Virtualization Format (OVF)is an attempt to provide a common format for storing the information and metadata describing a virtual machine image. Even though the OVF provides a full specification for packaging and distributing virtual machine images in completely platform-independent fashion, it is supported by few vendors that use it to import static virtual machine images.

3 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. 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

4 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.

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.

The lack of control over 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.

5 Organizational aspects

More precisely, storage, compute power, network infrastructure, and applications are delivered as metered services over the Internet. This introduces a billing model that is new within typical

enterprise IT departments, which requires a certain level of cultural and organizational process maturity. In particular, the following questions have to be considered:
1.What is the new role of the IT department in an enterprise that completely or significantly relies on the cloud?
2.How will the compliance department perform its activity when there is a considerable lack of control over application workflows?
3.What are the implications (political, legal, etc.) for organizations that lose control over some aspects of their services?
4.What will be the perception of the end users of such services?

From an organizational point of view, the lack of control over the management of data and pro-cesses poses not only security threats but also new problems that previously did not exist.

==========

CHAPTER : Principles of Parallel and Distributed Computing

Parallel vs. Distributed Computing

The terms parallel computing and distributed computing are often used interchangeably, even though they mean slightly different things. The term parallel implies a tightly coupled system, whereas distributed refers to a wider class of system, including those that are tightly coupled

More precisely, the term parallel computing refers to a model in which the computation is divided among several processors sharing the same memory.

The term distributed computing encompasses any architecture or system that allows the computation to be broken down into units and executed concurrently on different computing elements, whether these are processors on different nodes, processors on the same computer, or cores within the same processor

Elements of parallel computing

Why parallel processing?

- The primary purpose of parallel processing is to enhance the computer processing capability and increase its throughput, i.e. the amount of processing that can be accomplished during a given interval of time.

What is parallel processing?

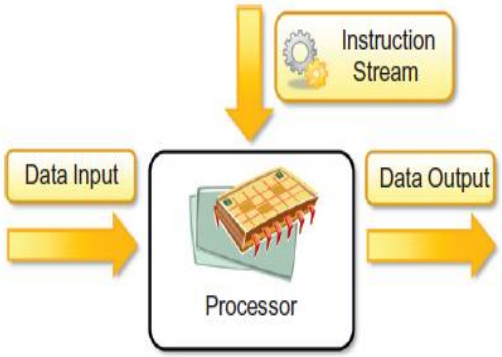
- Processing of multiple tasks simultaneously on multiple processors is called **parallel processing**
- The parallel program consists of multiple active processes (tasks) simultaneously solving a given problem. A given task is divided into multiple subtasks using a divide-and-conquer technique, and each subtask is processed on a different central processing unit (CPU).
- Programming on a multiprocessor system using the divide-and-conquer technique is called parallel programming.
- Parallel processing provides a cost-effective solution to this problem by increasing the number of CPUs in a computer and by adding an efficient communication system between them.
- The workload can then be shared between different processors. This setup results in higher computing power and performance than single-processor system offers.

Hardware architectures 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:

1. Single-instruction, single-data (SISD) systems
2. Single-instruction, multiple-data (SIMD) systems
3. Multiple-instruction, single-data (MISD) systems
4. Multiple-instruction, multiple-data (MIMD) systems

1. Single-instruction, single-data (SISD) systems



1. Uniprocessor machine capable of executing a single instruction, which operates on a single data stream
2. In SISD, machine instructions are processed sequentially; hence computers adopting this model are popularly called sequential computers.
3. Most conventional computers are built using the SISD model. All the instructions and data to be processed have to be stored in primary memory.
4. 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, statements such as

$$Ci = Ai * Bi$$

- Machines based on an SIMD model are well suited to scientific computing since they involve lots of vector and matrix operations.

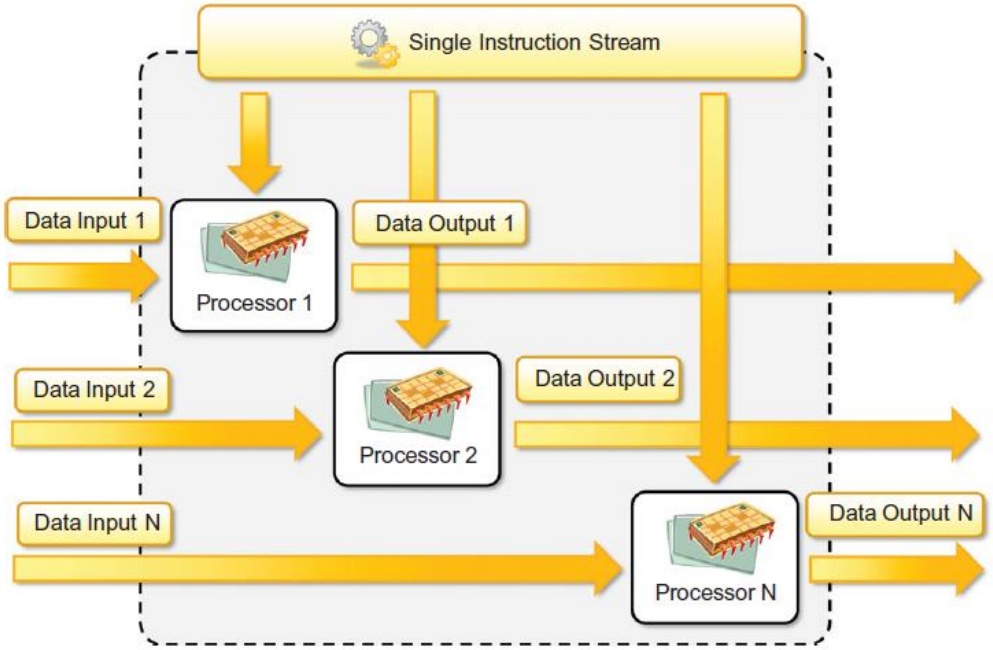


FIGURE 2.3

Single-instruction, multiple-data (SIMD) architecture.

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

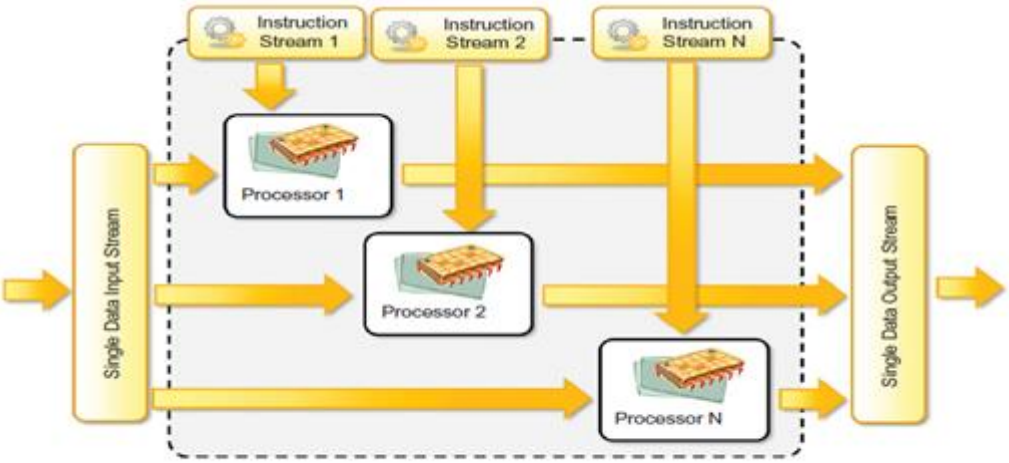


FIGURE 2.4
Multiple-instruction, single-data (MISD) architecture.

- 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.

Multiple-instruction, multiple-data (MIMD) systems

- An MIMD computing system is a multiprocessor machine capable of executing multiple instructions on multiple data sets
- 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.

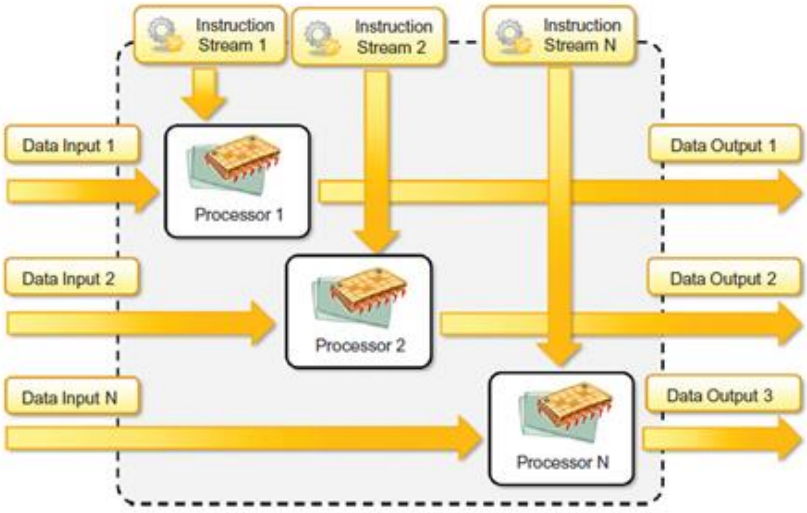


FIGURE 2.5
Multiple-instructions, multiple-data (MIMD) architecture.

MIMD machines are broadly categorized into shared-memory MIMD and distributed-memory MIMD

Approaches to parallel programming

A wide variety of parallel programming approaches are available. The most prominent among them are the following

- Data parallelism
- Process parallelism:
- Farmer-and-worker model:

In the case of data parallelism, the divide-and-conquer technique is used to split data into multiple sets, and each data set is processed on different PEs using the same instruction. This approach is highly suitable to processing on machines based on the SIMD model

In the case of process parallelism, a given operation has multiple (but distinct) activities that can be processed on multiple processors

In the case of the farmer-and-worker model, a job distribution approach is used:
one processor is configured as master and all other remaining PEs are designated as slaves; the master assigns jobs to slave PEs and, on completion, they inform the master, which in turn collects results

---

Elements of Distributed Computing

Distributed computing studies the models, architectures, and algorithms used for building and managing distributed systems. As a general definition of the term distributed system, we use the one

"A distributed system is a collection of independent computers that appears to its users as a single coherent system."

"A distributed system is one in which components located at networked computers communicate and coordinate their actions only by passing messages."

"A distributed system is a collection of computers that are separated and do not share a common memory or a clock. The processes executing on these computers communicates with one another by exchanging messages over communication channels. The messages are delivered after an arbitrary transmission delay. "

==========

Components of a distributed system

1. A distributed system is the result of the interaction of several components that traverse the entire computing stack from hardware to software.
2. It emerges from the collaboration of several elements that—by working together—give users the illusion of a single coherent system. Figure 2.10

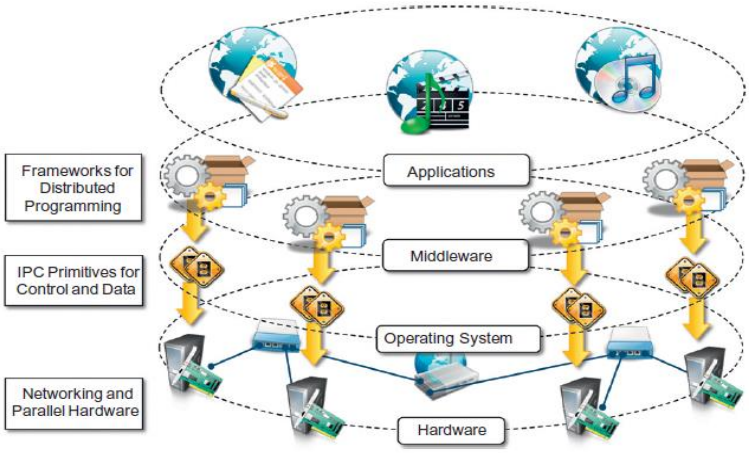


FIGURE 2.10
A layered view of a distributed system.

1. It is an overview of the different layers that are involved in providing the services of a distributed system.
2. At the very bottom layer, computer and network hardware constitute the physical infrastructure; these components are directly managed by the operating system, which provides the basic services for inter-process communication (IPC), process scheduling and management, and resource management in terms of file system and local devices. Taken together these two layers become the plat- form on top of which specialized software is deployed to turn a set of networked computers into a distributed system.
3. The middleware layer leverages such services to build a uniform environment for the development and deployment of distributed applications. This layer supports the programming paradigms for distributed systems.
4. By relying on the services offered by the operating system, the middleware develops its own protocols, data formats, and programming language or frameworks for the development of distributed applications. All of them constitute a uniform interface to distributed application developers that is completely independent from the underlying operating system and hides all the heterogeneities of the bottom layers.
5. The top of the distributed system stack is represented by the applications and services designed and developed to use the middleware. These can serve several purposes and often expose their features in the form of graphical user interfaces (GUIs) accessible locally or through the Internet via a Web browser. A very good example is constituted by Infrastructure-as-a-Service (IaaS) providers such as Amazon Web Services (AWS), which provide facilities for creating virtual

machines, organizing them together into a cluster, and deploying applications and systems on top.

Technologies for Distributed Computing

We introduce relevant technologies that provide concrete implementations of interaction models, which mostly rely on message-based communication. They are remote procedure call (RPC), distributed object frameworks, and service-oriented computing.

Remote procedure call

- RPC is the fundamental abstraction enabling the execution of procedures on client's request. RPC allows extending the concept of a procedure call beyond the boundaries of a process and a single memory address space.
- The called procedure and calling procedure may be on the same system or they may be on different systems in a network.
- Figure 2.14 illustrates the major components that enable an RPC system

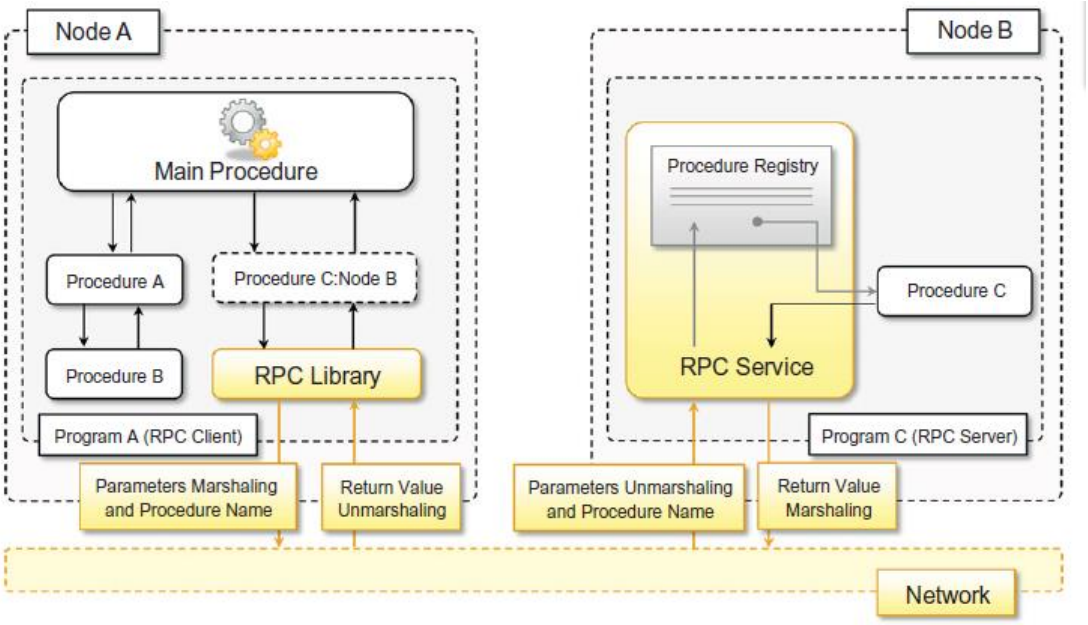


FIGURE 2.14

The RPC reference model.

- The system is based on a client/server model. The server process maintains a registry of all the available procedures that can be remotely invoked and listens for requests from clients that specify which procedure to invoke, together with the values of the parameters required by the procedure.
- An important aspect of RPC is marshaling, which identifies the process of converting parameter and return values into a form that is more suitable to be transported over a network through a sequence of bytes.
- The term unmarshaling refers to the opposite procedure. Marshaling and unmarshaling are performed by the RPC runtime infrastructure, and the client and server user code does not necessarily have to perform these tasks.

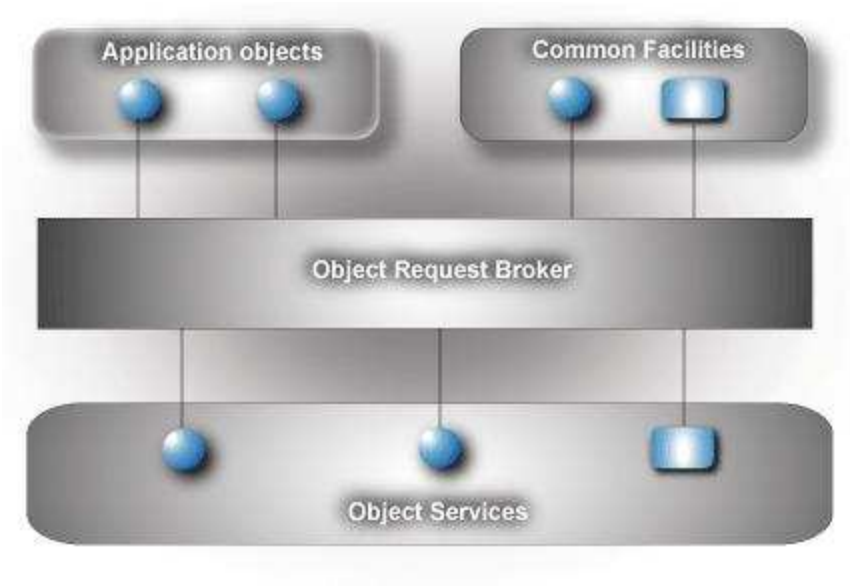
- The RPC runtime, on the other hand, is not only responsible for parameter packing and unpacking but also for handling the request-reply interaction that happens between the client and the server process in a completely transparent manner.
- Therefore, developing a system leveraging RPC for IPC consists of the following steps:
    - Design and implementation of the server procedures that will be exposed for remote invocation.
    - Registration of remote procedures with the RPC server on the node where they will be made available.
    - Design and implementation of the client code that invokes the remote procedure(s).
- Each RPC implementation generally provides client and server application programming inter- faces (APIs) that facilitate the use of this simple and powerful abstraction.
- Since the server and the client processes are in two separate address spaces, the use of parameters passed by references or pointers is not suitable in this scenario, because once unmarshaled these will refer to a memory location that is not accessible from within the server process.
- Second, in user-defined parameters and return value types, it is necessary to ensure that the RPC runtime is able to marshal them. This is generally possible, especially when user-defined types are composed of simple types, for which marshaling is naturally provided.
- Currently, the term RPC implementations encompass a variety of solutions including frameworks such distributed object programming (CORBA, DCOM, Java RMI, and .NET Remoting) and Web services that evolved from the original RPC concept.

CORBA

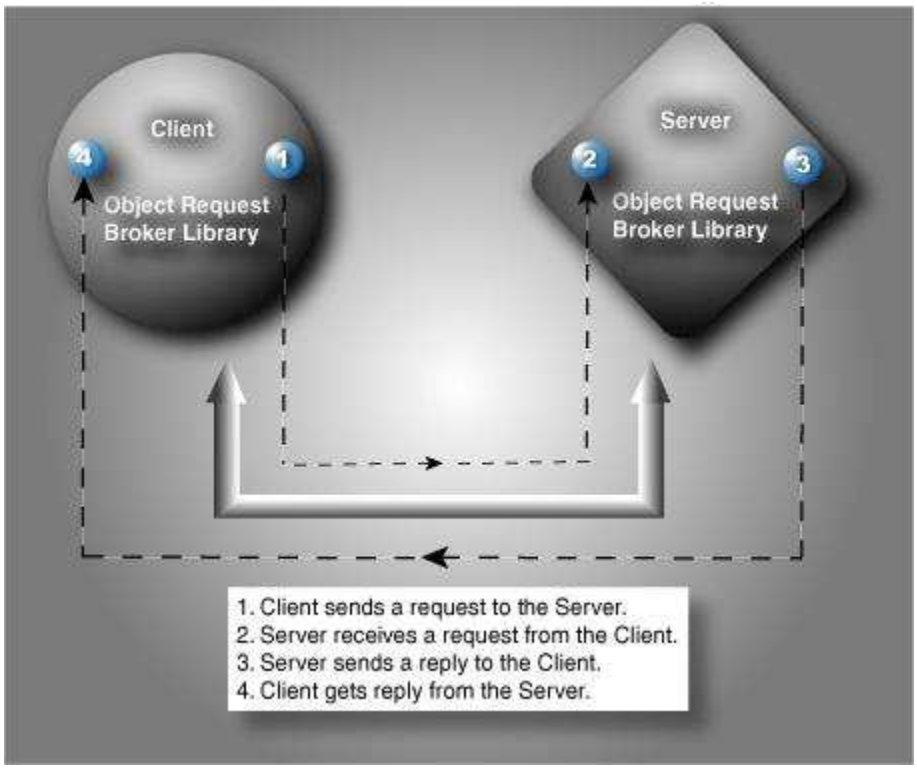
The Common Object Request Broker Architecture (CORBA) is a standard developed by the Object Management Group (OMG) to provide interoperability among distributed objects. CORBA is the world's leading middleware solution enabling the exchange of information, independent of hardware platforms, programming languages, and operating systems. CORBA is essentially a design specification for an Object Request Broker (ORB), where an ORB provides the mechanism required for distributed objects to communicate with one another, whether locally or on remote devices, written in different languages, or at different locations on a network.

The CORBA Interface Definition Language, or IDL, allows the development of language and location-independent interfaces to distributed objects. Using CORBA, application components can communicate with one another no matter where they are located, or who has designed them. CORBA provides the location transparency to be able to execute these applications.

CORBA is often described as a "software bus" because it is a software-based communications interface through which objects are located and accessed. The illustration below identifies the primary components seen within a CORBA implementation.



Data communication from client to server is accomplished through a well-defined object-oriented interface. The Object Request Broker (ORB) determines the location of the target object, sends a request to that object, and returns any response back to the caller. Through this object-oriented technology, developers can take advantage of features such as inheritance, encapsulation, polymorphism, and runtime dynamic binding. These features allow applications to be changed, modified and re-used with minimal changes to the parent interface. The illustration below identifies how a client sends a request to a server through the ORB:



DCOM

DCOM is a programming construct that allows a computer to run programs over the network on a different computer as if the program was running locally. DCOM is an acronym that stands for Distributed Component Object Model. DCOM is a proprietary Microsoft software component that allows COM objects to communicate with each other over the network.

Imp Questions

- With a neat diagram explain Cloud Computing Architecture
- With a neat diagram explain Cloud Reference Model
- Classify and explain the various types of clouds/deployment models
- Explain in detail different cloud service models.
- Write a note on Economics of the Cloud
- How does cloud computing help to reduce the time to market for applications and to cut down capital expenses?
- Explain in detail open challenges in cloud computing.
- What is the difference between parallel and distributed computing?
- Briefly explain parallel and distributed computing
- **Explain the Approaches to parallel programming**
- With a neat diagram explain **Hardware architectures for parallel processing**
- **With a neat diagram explain Components of a distributed system**
- **Brifly explain Technologies for Distributed Computing**