CONTENTS

❑❑❑

Cloud Computing

University Prescribed Syllabus

Introduction to Cloud Computing, Characteristics and benefits of Cloud Computing, Basic concepts of Distributed Systems, Web 2.0, Service-Oriented Computing, Utility-Oriented Computing. Elements of Parallel Computing. Elements of Distributed Computing. Technologies for Distributed Computing. Cloud Computing Architecture. The cloud reference model. Infrastructure as a service. Platform as a service. Software as a service. Types of clouds.

1.1 Introduction to Cloud Computing

GQ. 1.1.1 Write short notes on Introduction of Cloud Computing.

- Computing is being transformed into a model with services.
- In such a model, users gain access to services based on their needs, regardless of where the services are hosted.
- Different computing paradigms, such as grid computing, have promised to deliver this utility computing vision.
- Cloud computing is the most recent emerging paradigm that promises to turn the vision of "computing utilities" into reality.
- Cloud computing is a technological advancement that mainly focuses on how we leverage existing services to design computing systems, develop applications, and build software.
- It is mainly focus on the concept of dynamic provisioning, which is applied to services as well as to capacity, storage, networking, and information technology (IT) infrastructure.
- The resources are made available via the Internet and offered on a pay-per-use basis from cloud computing vendors.
- Today, anyone with a credit card can subscribe to cloud services and grow and shrink the infrastructure that serves its application according to demand and implement servers, and only pay for using these resources have been used.
- In 1969, Leonard Kleinrock, one of the principal scientists of the original Advanced Research Projects Agency Network (ARPANET), seeded the Internet.
- This is based on a service-provisioning model anticipated a massive transformation of the entire computing industry in the 21st century, which would make computing services readily available on demand.
- In such a model, users use services based on their needs without hosting services.
- This model has been referred to as utility computing or, more recently (since 2007) cloud computing.
- The latter term often refers to infrastructure as a "cloud" from which businesses and users can access applications from anywhere in the world and as services on demand.
- Cloud computing transforms IT services into utilities.
- Such a delivery model is made possible by an effective combination of multiple technologies that have reached reasonable maturity levels.
- Web 2.0 technologies play a central role in building cloud computing, which is an attractive opportunity to build computing systems.
- Service orientation allows cloud computing to deliver its capabilities with familiar capabilities, while virtualization provides the degree of customization, control and flexibility required on cloud computing and the creation of enterprise systems.
- The NIST's definition of cloud computing is, "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"

1.1.1 The Vision of Cloud Computing

GQ. 1.1.2 Describe the vision introduced by cloud computing.

- The Cloud Computing provides the provision of virtual hardware, runtime environment and services to provide money to an individual.
- These all things can be utilized as long as they are required by the user, there is no requirement for upfront commitment.
- The entire collection of computing systems is transformed into a collection of utilities, which can be provisioned and assembled to deploy the system in hours rather than days, with the expense of maintenance.
- The long-term vision of cloud computing is that IT services are traded as utilities in the open market without technical and legal constraints.
- In the near future we can imagine that it will be possible to find a solution that matches our needs by registering our request in the global digital marketplace doing business with cloud computing services.
- The existence of such a market will enable automation of the search process and its integration into its existing software systems.
- The existence of a global platform for trading cloud services will also help service providers to increase their revenue.
- The cloud provider can also become a consumer of a competitive service to fulfill its promises to customers.

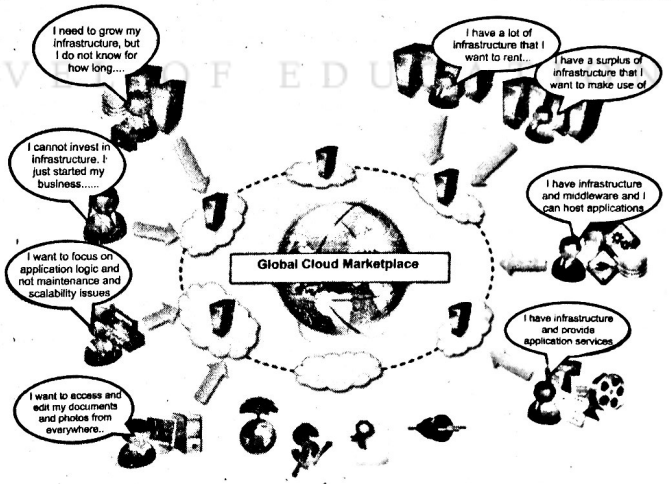

Fig. 1.1.1 : The Cloud Computing Vision

1.1.2 Defining Cloud

GQ. 1.1.3 *Define the term Cloud.*

- Cloud computing has become a popular discussion; It has been widely used to refer to various technologies, services and concepts.
- It is often associated with virtualized infrastructure or hard-ware on demand, utility computing, IT outsourcing, platforms and software as a service, and many other things that are now the focus of the IT industry.
- The term cloud has historically been used in the telecommunications industry as an abstraction of networks in system diagrams.
- It seems like the symbol of the most popular computer network i.e. the Internet.
- The meaning also applies to cloud computing, which refers to the Internet-centric way of computing.
- The Internet plays a fundamental role in cloud computing, as it represents either the medium or the platform through which many cloud computing services are delivered and accessible.
- Cloud computing refers to as applications delivered as services in the Internet and hardware well as system software in datacenters which providing those services.
- This definition describes cloud computing as a touching phenomenon across the stack: from built-in hardware to high-end software services and applications.
- It introduces the concept of everything as a service, mostly referred to as XaaS, where the various components of a system - IT infrastructure, development platform, database, and so on.
- Can be delivered, measured, and finally priced as a service.
- Cloud computing provides ubiquitous, on-demand, convenient network access to a shared pool of configurable resources, like networks, servers, storage, applications, and services.
- It supports rapid provisioning with minimal management effort and can be issued or service provider negotiation.
- Another important aspect of cloud computing is its usability oriented approach.
- Unlike any other trend in distributed computing, cloud computing mainly based on delivering services with a given pricing model, in most of the cases a "pay-per-use" strategy.
- This makes it possible to access online storage, rent virtual hardware or use development platforms, and pay for their effective use, with no or minimal cost.
- All these operations can be performed by entering credit card details and accessing the exposed services through a web browser.
- Cloud computing is based on three criteria :
  - The services are accessible via internet.
  - To get started with cloud computing zero capital expenditure is necessary.
  - Pay as per usages.

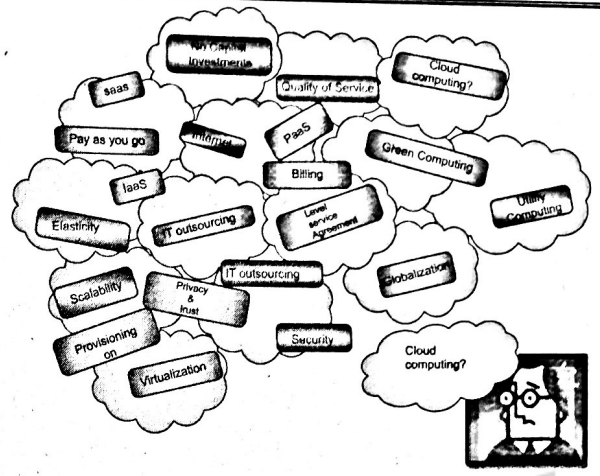

Fig. 1.1.2 : Cloud computing technologies, concepts, and ideas

1.1.3 Cloud Computing Models

Cloud Computing models are characterised in following two types :

1.1.3.1 Cloud Computing Deployment Model

UQ. 1.1.4 *Explain the concept of public, private and hybrid cloud.* **(MU - April 19, 5 Marks)**

GQ. 1.1.5 *What is Cloud Deployment Model?*

Cloud deployment models shows how cloud services are made available to users. The three deployment models associated with cloud computing are as follows :

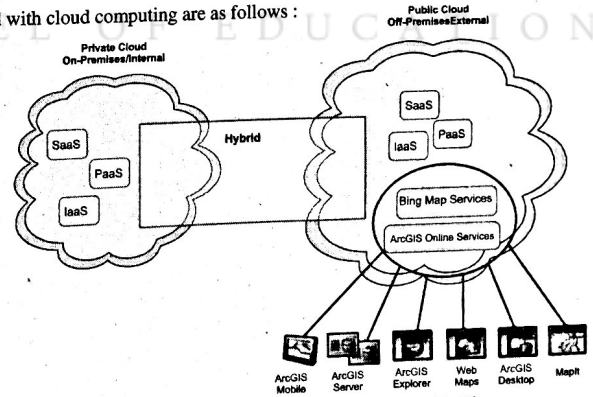

Fig. 1.1.3: Cloud Computing Deployment Model

1. Public Cloud

- As the name public cloud suggests, this deployment model supports all users who want to use a computing resource, such as hardware (OS, CPU, memory, storage) on a subscription, or Software (application server, database) base.
- The most common uses of public clouds are for application development and testing, non-mission-critical tasks like file-sharing and e-mail service.

2. Private Cloud

- As per its name, a private cloud is typically infrastructure commonly utilized by a single organization.
- This infrastructure can be managed or maintained by the organization itself to provide support to different user groups, or it can be managed by a service provider who looks after it either on-site or off-site.
- As compare to public cloud private clouds are more expensive because capital expenditures are involved in acquiring and maintaining them.
- However, private clouds are better at addressing the security and privacy concerns of organizations today.

3. Hybrid cloud

- In hybrid cloud, an organization uses a mutual private and public cloud infrastructure.
- Many organizations use this model when they need to rapidly grow their IT infrastructure, such as when leveraging public clouds to complement the available capacity within a private cloud.
- Example is that, if an online retailer needs more computing resources to run its web application during the holiday season, it can obtain those resources through the public cloud.

1.1.3.2 Cloud Computing Reference Model

GQ. 1.1.6 Write short note on Cloud Computing Reference Model.

Cloud service models tells how cloud services are made available to the clients. Most fundamental service models include a combination of infrastructure as a service (IaaS), platform as a service(PaaS), and software as a service(SaaS).

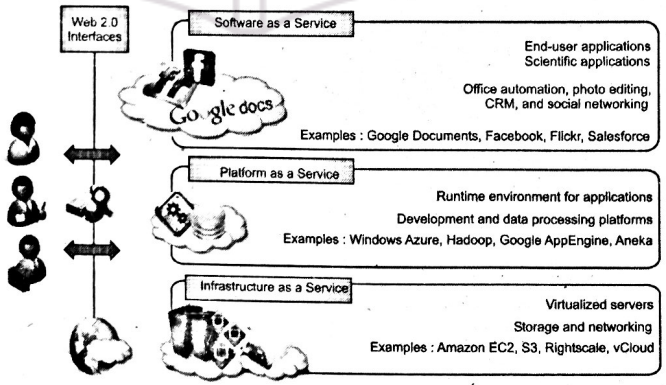

Fig. 1.1.4 : Cloud Computing Reference Model

1. Infrastructure as a Service (IaaS)

- The Infrastructure as a Service (IaaS) model provides infrastructure components to customers. Components may include virtual machines, storage, networks, firewalls, load balancers, etc.
- With IaaS, clients have access to the lowest-level software in the stack - that is, on the management dashboard of the operating system or firewall or load balancer on virtual machines.
- Amazon Web Services is one of the largest IaaS providers.

2. Platform as a Service (PaaS)

- The Platform as a Service (PaaS) model delivers a pre-built application platform to the customer; Customers should not spend time building the underlying infrastructure for their applications.
- On the backend, PaaS, automatically measures and provisions the required infrastructure based on application requirements.
- Basically, PaaS provide an API that includes a set of functions for programmatic platform management and solution development.
- Google AppEngine is a popular PaaS provider, and offers some PaaS solutions in addition to the Amazon Web Services IaaS offerings.

3. Software as a Service (SaaS)

- The Software as a Service (SaaS) provides ready online software solutions.
- The SaaS software provider has complete control and management on application software.
- SaaS application examples include online mail, project-management systems, CRMs, and social media platforms.
- The basic difference between SaaS and PaaS is that PaaS typically represents a platform for application development, while SaaS provides online applications that are already developed.

1.1.4 Characteristics of Cloud Computing

UQ. 1.1.7 Explain the characteristics of Cloud computing as per NIST. **(MU - April 19, 5 Marks)**

GQ. 1.1.8 Explain the characteristics of Cloud Computing.

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

1. Resources Pooling

- This means that the cloud provider pulled computing resources to provide services to multiple customers with the help of a multi-tenant model.
- Different physical and virtual resources are assigned and reassigned which depends on customer demand.
- The customer typically has no control or information on the location of the resources provided, but is able to specify the location at a high level of abstraction.

2. On-Demand Self-Service

- This is one of the important and valuable characteristics of cloud computing because the user can continuously monitor server uptime, capabilities and allocated network storage.
- With this feature, the user can also monitor computing capabilities.

3. Easy Maintenance

- Servers are easily maintained and the downtime is very low and even in some cases, there is no downtime.
- Cloud computing slowly improves with each update.
- Updates are more compatible with devices and perform faster than older ones with bugs that have been fixed.

4. Large Network Access

- The user can access data from the cloud or upload data to the cloud from anywhere with the help of a device and an internet connection.
- These capabilities are available throughout the network and are accessed with the help of the Internet.

5. Availability

- The capabilities of the cloud can be modified as per usage and can be greatly enhanced.
- It analyzes storage usage and allows the user to purchase additional cloud storage when needed for a much smaller amount.

6. Automatic System

- Cloud computing automatically analyzes the required data and supports the ability to measure at some level of services.
- We can monitor, control and report usage. This will provide transparency for the host as well as the customer.

7. Economical

- This is a one-time investment because the company (host) has to purchase storage and a small portion of it can be provided to many companies that save the host monthly or annual costs.
- Only the amount spent is on basic maintenance and some other expenses which are very less.

8. Security

- Cloud Security, is one of the best features of cloud computing.
- It creates a snapshot of the data stored so that the data may not get lost even if one of the servers gets damaged.
- The data is stored within the storage devices, which cannot be hacked and utilized by any other person.
- The storage service is quick and reliable.

9. Pay as you go

- In cloud computing, a user only has to pay for the service or the space they use.
- There are no hidden or additional fees that have to be paid.
- The service is economical and most of the time some space is allocated for free.

10. Measured Service

- Cloud computing resources are used for monitoring and the company uses it for recording.
- This resource utilization is analyzed by supporting charge-per-use capabilities.
- This means that resource usage that can be either virtual server instances that are running in the cloud is being measured and reported by the service provider.
- The model you go to is variable based on the actual consumption of the payment manufacturing organization.

1.1.5 Advantages of Cloud Computing

UQ. 1.1.9 Explain the benefits for a start up to implement cloud infrastructure. **(MU - April 19, 5 Marks)**

GQ. 1.1.10 What are the advantages and disadvantages of cloud computing?

1. Easy Implementation

- Cloud hosting allows the business to maintain similar applications and business processes to deal with backend technology.
- Easily manageable by the Internet, cloud infrastructure can be easily and quickly accessed by enterprises.

2. Accessiblity

- Access your data anywhere, anytime.
- An Internet cloud infrastructure maximizes enterprise productivity and efficiency and ensures that your application is always accessible.
- This allows for easy collaboration and sharing between users across multiple locations.

3. No hardware required

- Since everything will be hosted in the cloud, a physical storage center is not required.
- However, a backup can be worth looking into in the event of a disaster that can curb your company's productivity.

4. Cost per Head

The cost of new required technology of cloud should keep minimum with cloud hosting services, provides businesses to utilize additional time and resources to improve the company's infrastructure.

5. Flexibility for Development

- The cloud is easily scalable so companies can add or subtract resources depending on their needs. With the growth of company, their resources, needs will also increases.

- As companies grow, their system will grow with them. Cloud resources are quite flexible to easily scale-up and scale-down as per need.

6. Efficient Recovery

- Cloud computing delivers faster and more accurate retrieval of applications and data.
- With little downtime, it is the most efficient recovery plan.

1.1.6 Disadvantages of Cloud Computing

GQ. 1.1.11 What are the advantages and disadvantages of cloud computing?

1. No longer in control

- When transferring services on to the cloud, you are handing over your data and information.
- Even though companies that have in-house IT engineers, they will not be able to handle issues on their own.
- However, the cloud company should have a 24/7 live help desk that can quickly fix any problem.

2. Not all features can be found

- Not all cloud services are the same.
- Some cloud providers offer limited editions and enable only the most popular features, so you may not get every feature or customization.
- Before signing up, make sure you know what your cloud service provider offers.

3. Doesn't mean you should do away with servers

- You may have fewer servers to handle, which means that your IT staff is less to handle, but that doesn't mean you can let all your servers and employees go.
- Although data centers and cloud infrastructure are costly, redundancy is critical for backup and recovery.

4. No redundancy

- A cloud server is not redundant and also not backed up. Since technology can fail here and there, avoid burning by purchasing redundancy plans.
- Although it is an additional cost, in most cases it will be worth it.

5. Bandwidth Issues

For ideal performance, customers have to plan accordingly and do not pack large amounts of servers and storage devices into a small set of data centers.

1.1.7 Challenges in front of Cloud Computing

GQ. 1.1.12 What are the challenges in front of Cloud Computing?

1. Security and Privacy of Cloud

- The data store in the cloud must be secure and provide complete privacy.
- Customers trust the cloud provider so much.
- This means that cloud providers must take necessary security measures to secure customers' data.
- Security is also the responsibility of the customer as they must provide a strong password, do not share the password with anyone, and change the password regularly which we did.
- If the data is outside the firewall then there may be some problems that can be eliminated by the cloud provider.
- Hacking and malware are also one of the major problems as it can affect many customers.
- Hacking can cause data loss; encrypted file system and many other problems.

2. Interoperability and Portability

- The customer should be provided with migration services in and out of the cloud.
- There should be no bond period as it can create obstacles for customers.
- The cloud should have the ability to provide facilities on campus.
- One of the cloud challenges is remote access that can be eliminated by the cloud provider so that the customer can get security from the cloud anywhere.

3. Reliable and Flexible

- Reliability and flexibility are also one of the important challenges for cloud customers and can be eliminated in such a way that the data provided to the cloud should not leak and the host must provide reliability to the customers.
- To eliminate this challenge, services provided by third parties should be monitored and supervised on performance, robustness, and business dependency.

4. Cost

- Cloud computing is inexpensive but modifying the cloud to customer demand can sometimes be expensive.
- Furthermore, it can create obstacles in small-scale organization as the cloud is modifying because their demand can sometimes cost more.
- In addition, transferring data from the cloud to the premises can sometimes be costly.

5. Downtime

- Downtime is a common challenge of cloud computing because no cloud provider guarantees a platform that is free of downtime.
- An internet connection also plays an important role because if a company has an unfaithful internet connection then there can be a problem as they may face downtime.

6. Lack of resources

- Lack of resources and expertise is also one of the major challenges facing the cloud industry and many companies are hoping to overcome this challenge by hiring more workers who are more experienced.
- These workers will not only help companies overcome challenges but also train existing employees to benefit the company.
- Today many IT workers are working to promote cloud computing expertise and the CEO of the company is finding it difficult because the workers are not very skilled.
- This assumes that workers with knowledge of the latest developments and related technologies will become more valuable in business.

7. Management of Multi-Cloud Environment :

- Nowdays companies do not use a single cloud, instead they are using multiple clouds.
- An average company is using 4.8 different public and private clouds due to which their management is disrupted.
- When a company uses multi-cloud, the IT team faces a lot of complexities.
- This cloud challenge can be eliminated by training employees, using appropriate tools and conducting research.

1.2 Basic Concepts of Distributed System

UQ. 1.2.1 Write short note on Distributed computing. (MU - April 19, 5 Marks)

GQ. 1.2.2 What is a distributed system? What are the components that characterize it?

- A computing concept namely Distributed computing is that, multiple computers working on a single task.
- In distributed computing, one task is split into several components or several sub-task, and every sub-task is solved by totally different computers.
- As long as the computers are connected via network, they can communicate with each other to solve their problem.
- If done properly, the computers perform like a single unit or entity.
- One vital characteristic of a distributed system is that processes are not executed on a single processor but rather divide it into a number of processors.
- This needs that interprocess communication mechanisms are introduced to manage interaction between processes executing on completely different machines.
- Faults within the network result in isolation of the computers that are connected thereto, however this doesn't mean that these computers should stop working.
- If truth be told the programs running on them might not be able to find whether or not the network has failing.
- Similarly, the failure of a system, or an unexpected application crash, isn't forthwith detected by alternative elements with which the failing component communicates.

- Every part in a distributed system will fail independently, allow others still running properly.
- The ultimate goal of distributed computing is to maximize performance by connecting users and IT resources in a very efficient, clear and reliable manner.
- It additionally ensures fault tolerance and permits resource accessibility within the event even if that one in every of the part fails

1.2.1 Distributed System Architecture

- The distributed system is a collection of independent computers that appear as a coherent system for its users.
- It consists of multiple computers but they do not share memory.
- Every computer have their own memory and runs its own operating system.
- Computers can communicate with each other via a communication network.

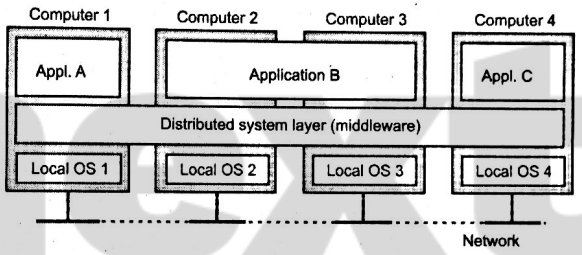

Fig. 1.2.1 : Distributed System Architecture

1.2.2 Characteristics of Distributed System

GQ. 1.2.3 Provide a brief characterization of a distributed system.

- **Fault-Tolerant :** It can overcome component failures without doing the wrong thing.
- **Highly Available :** It can restore operations, allowing some components to resume to provide services even if they fail.
- **Recoverable :** Failed components can restart themselves and reconnect the system, after repairing the cause of the failure.
- **Consistent :** The system can coordinate actions by multiple components, often in the presence of concurrency and failure. It provides the capability of a distributed system to act like a non-distributed system.
- **Scalable :** It can also work correctly, as some aspects of the system are smaller in size.
- **Predefined Performance :** Ability to deliver desired accountability in a timely manner.
- **Secure :** The system authenticates access to data and services

1.2.3 Advantages of Distributed System

- **Data Sharing :** There is a provision for a user on one site to access data residing on other sites.
- **Autonomy :** The system is autonomous due to data sharing through data distribution, with each site being able to maintain a degree of control over the data stored as local autonomy.
- **Availability :** If a site fails in a distributed system, the remaining sites may be able to continue operations.

1.2.4 Disadvantages of Distributed System

- **Software Development Cost :** A distributed database system is more difficult to implement due to the high cost of installation.
- **Greater Potential for Insects :** Since sites operate in parallel, the accuracy of the algorithm is difficult to maintain and difficult to recover from failures.
- **Increased processing overhead :** Better inter-coordination and overhead is required for information exchange and additional computations.

1.3 Web 2.0

GQ. 1.3.1 What is the role of Web 2.0?

- Web is the primary interface (medium) through which cloud computing provides or provides its services to all.
- The meaning of the word web has evolved since the time of the monument.
- Currently encompasses a set of web technologies and services that facilitate interactive sharing, collaboration, user-centric design and application structure.
- So basically Web 2.0 is the current state of online technology as it compares to the early days, it involves more and more user engagement and collaboration and enhanced communication channels.
- "Web 2.0 is the business revolution in the computer industry caused by the move to the internet as a platform, and an attempt to understand the rules for success on that new platform." - Tim O' Reilly.
- It is an improved version of the first world wide web specifically characterized by the transformation of static to dynamic or user-generated content and the development of social media.
- The background concept Web 2.0 refers to rich web oriented architecture, web applications and social web.
- It refers to changes in the way web pages are designed and used by users, without any technical specification.
- The name Web 2.0 is utilized for the second generation of www, i.e. the World Wide Web, where the Web has moved to static HTML pages for a more responsive and interactive and dynamic web experience.
- Web 2.0 focuses on the ability of users or people to share and share information online through social media, web based communities, blogging, and the Internet off course.
- But now the web is a major component for social media, with new web applications such as AZAX and more modern browsers that provide opportunities for people to express their views.

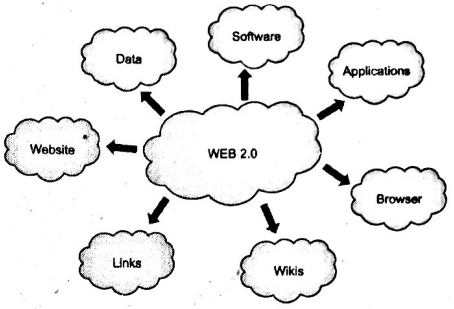

Fig. 1.3.1 : Web 2.0

1.3.1 Features and Benefits of Web 2.0

- We can access all services available on the Internet anytime or from anywhere.
- Today's web services are user friendly or have a better user interface.
- Multiple types or different types of media are available on the web.
- It can create a dynamic learning communities.
- Now anyone can have facility of real time discussion.
- Everyone is now an author and an editor, now everything or every change is traceable.
- Learners can be actively involved in knowledge creation.
- Search result is efficient and quick.
- Updating any information is easy.
- Web 2.0 is quick, collaborative, reliable, etc.

1.3.2 Service Running on Web 2.0

- Facebook
- Flicker
- Linked in
- YouTube
- Wikipedia
- BlogSpot
- Word press
- Twitter
- Pinterest
- Google Docs
- Instagram and many more.

1.4 Service-Oriented Computing

UQ. 1.4.1 What is SOA? (MU - April 19, 1 Mark)

- Service orientation is the main reference model for cloud computing systems.
- This approach adopts the services concept as the main building blocks of application and system development.
- Service-oriented computing (SoC) supports the development of rapid, low-cost, flexible, interopera- ble and developable applications and systems.
- Service-oriented computing presents and expands two important concepts, which are also fun-damaging to cloud computing: quality of service (QoS) and software-as-a-service (SaaS).
- Service-oriented computing is based on a service-oriented architecture (SOA).
- A service-oriented architecture (SOA) is a construction format in computer software design in which application parts or tolls provide services to other parts by using a standard communications protocol, basically via internet.
- The standards of service-orientation are not based on any vendor, product, tools or technology.
- SOA simply provides it easier for software components over different networks to work with one and another.
- Web services which are based on the SOA architecture inclined to make web service extremely independent.
- The web services themselves can interchange data with one and other.
- Because of the fundament principles on which they are made, they are not depending on any sort of human interaction and also don't need any code alteration.
- It guarantees that the web services on a network can communicate with each and other flawlessly.

1.5 Utility-Oriented Computing

UQ. 1.5.1 What is utility computing? (MU - April 19, 5 Marks)

- Utility computing, or computer utility, is nothing but a service provisioning model in which a service provider makes computing resources and infrastructure management available to the customer as needed, and charges them for specific usage rather than a flat rate.
- The term utility refers to utility services such as electricity, telephone, water and gas that are provided by a utility company.
- Similar to the electricity or telephone if the customer receives the utility computing, the computing power on a shared computer network, its consumption is measured and billed on that basis.
- The term utility refers to utility services such as electricity, telephone, water, and gas which are provided by a utility company.
- As like electricity or telephone if the customer receives utility computing, computing power on a shared computer network, its consumption is measured and billed on that basis.
- Utility computing is very similar to virtualization, so that the total amount of web storage space, along with the computing power that users receive, is much larger than a single time-sharing computer.

- Multiple backend web servers are often used to make this type of web service possible.
- Dedicated web servers can be used in cluster forms that are specifically created and then leased to end users.
- The method of using a single 'computation' on multiple web servers is called distributed computing.

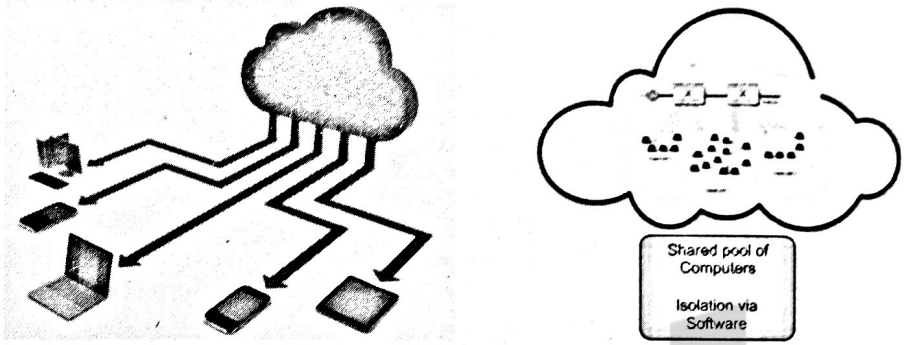

Fig. 1.5.1 : Utility Computing

1.5.1 Properties of Utility Computing

1. Scalability

- Utility computing should be ensured that adequate IT resources are available in all circumstances.
- Increasing demand for a service but it does not harm its quality (eg, response time).

2. Demand pricing

- Until now, companies have had to buy their own hardware and software when they need computing power.
- This IT infrastructure must be paid before the rule, regardless of how quickly the company uses them.
- Technology vendors obtain this link, for example, the fact that the lease rate for their servers depends on how many CPUs the customer has enabled.
- If it can be measured as the computing power to actually claim individual classes in a company, then the IT costs in the internal system may be directly attributable to individual departments.
- Other forms of connection are possible with the use of IT costs.

3. Standardized Utility Computing Services

- The utility computing service provider provides its customers with a list of standardized services.
- These may include agreements on various service level agreements (quality and price of IT).
- The client has no influence on underlying technologies such as the server platform.

4. Utility Computing and Virtualization

- Virtualization technologies can be used to share web and other resources in a shared pool of machines.
- This will divide the network into logical resources instead of available physical resources.
- An application is not stored on a specific pre-defined server or any store, but a free server runtime or memory from the pool.

5. Automation

- The repeating management tasks such as installing a new server or installing updates can be automated.
- In addition, the allocation of resources to services automatically and the management of IT services should be optimized, considering service level agreements and operating costs of IT resources.

1.5.2 Different Types of Utility Computing

- Utility computing is categories into two types: Internal utility and External utility.
- Internal utility means that the computer network is shared only within a company.
- Used by many different computer companies to pool together a particular service provider called an external utility.
- Additionally, various hybrid forms are possible in this type of utility computing.

1.5.3 Advantages of Utility Computing

1. Cost

- The cost of IT can be reduce due to Utility computing, given that existing resources can be used more effectively.
- In addition, the cost is transparent and can be assigned directly to different departments of a company.
- There will be fewer people required for operational activities in IT departments.

2. Flexibility

- Companies gain more flexibility, as their IT resources adapt to fluctuating demand more quickly and easily.
- Overall, the entire IT infrastructure is easier to manage, as it will no longer be built for every application, which is an advantage for specific IT infrastructure.

1.6 Elements of Parallel Computing

1.6.1 What is Parallel Computing?

- In the simplest sense, parallel computing is the use of multiple computational resources simultaneously to solve a computational problem :
  - A problem breaks into discrete parts that can be solved concurrently.
  - Every part is then broken down to a series of instructions.
  - The instructions for each part are simultaneously executed on different processors.

  - An overall control or coordination mechanism is employed.
- The entire real world operates in a dynamic nature i.e. many things happen at a certain time but concurrently in different places. This data is huge for large-scale management.
- Real-world data requires more dynamic simulation and modelling, and to achieve the same, parallel computing is important.
- Parallel computing used to provide concurrency and also saves time and money.
- Complex, large datasets and their management can be conducted using an approach of only and only parallel computing.
- Ensures effective use of resources. Hardware is guaranteed to be used effectively, whereas in serial computation only a part of the hardware is used and the rest becomes useless.
- Furthermore, it is impractical to implement a real-time system using serial computing.

1.6.2 Hardware Architectures for Parallel Processing

UQ. 1.6.1 Explain the hardware architecture of Parallel Computing. **(MU - April 19, 5 Marks)**

GQ. 1.6.2 List the major categories of parallel computing systems.

Based on the number of instructions and data streams being processed simultaneously, computing systems are classified into four major categories :

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

- The SISD computing system is a uniprocessor machine capable of executing a single instruction, operating on a single data stream.
- In SISD, machine instructions are processed sequentially, and computers adopting this model are popularly known as sequential computers.
- Most traditional computers have SISD architecture.
- All 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 (dependent) to the rate at which the computer can transfer information internally.
- Key representatives of SISD Systems IBM PC, Workstation.

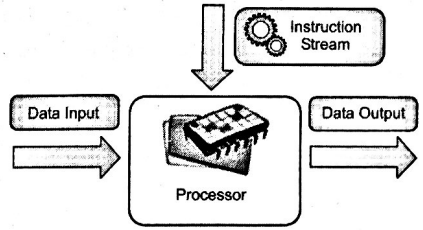

Fig. 1.6.1 : Single-instruction, single-data (SISD) systems

2. Single-instruction, multiple-data (SIMD) systems

- A SIMD system is a multiprocessor machine capable of executing the same instructions on all CPUs but operating on different data streams.
- Machines based on a SIMD model are well suited for scientific computing because they involve a lot of vector and matrix operations.
- So that information can be transmitted to all processing elements (PEs), the organized data elements of vectors can be divided into several sets (N-sets for N-PE Systems) and each PE can process one data set.
- The key representative of SIMD system is Cray's vector processing machine.

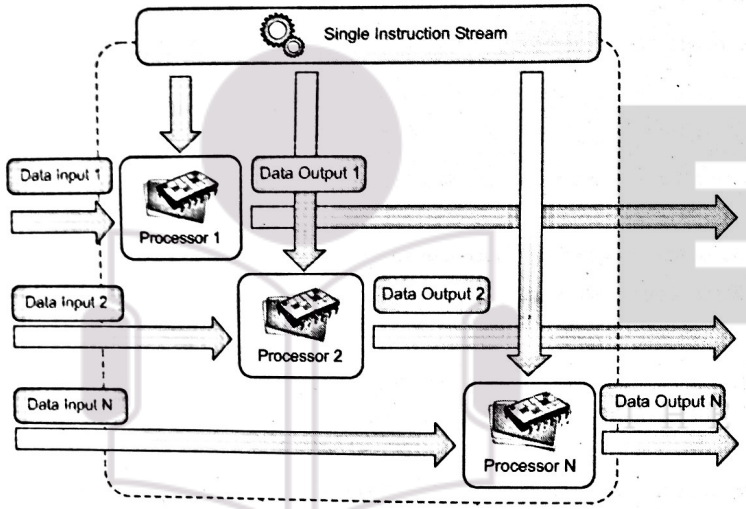

Fig. 1.6.2 : Single-instruction, multiple-data (SIMD) systems

3. Multiple-instruction, single-data (MISD) systems

- The MISD computing system is a multiprocessor machine capable of executing various instructions on different PEs but all of them working on the same dataset.

  Example Z = sin (x) + cos (x) + tan (x)

- The system performs different operations on the same data set.
- Machines built using the MISD model are not useful in most applications, some machines are built, but none of them are commercially available.

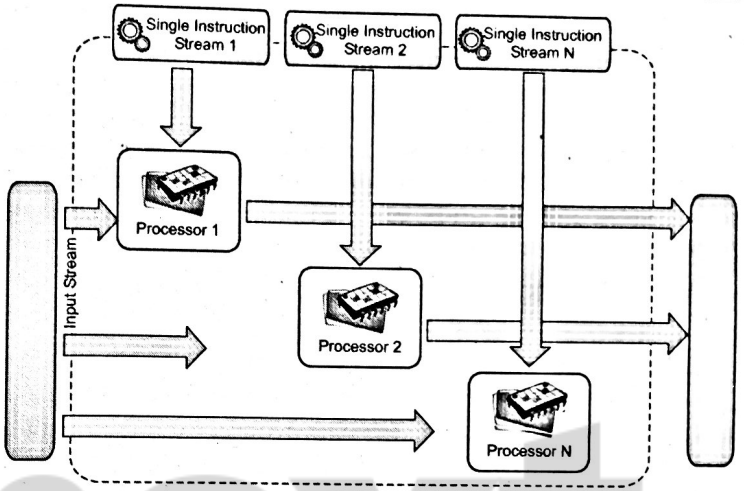

Fig. 1.6.3 : Multiple-instruction, single-data (MISD) system

4. Multiple-instruction, multiple-data (MIMD) systems

- The MIMD system is a multiprocessor machine capable of executing multiple instructions on multiple data sets.
- Each PE of the MIMD model has different instructions and data streams.
- Therefore machines manufactured using this model are capable of any type of application.
- Unlike SIMD and MISD machines, PEs operate asynchronously in MIMD machines.

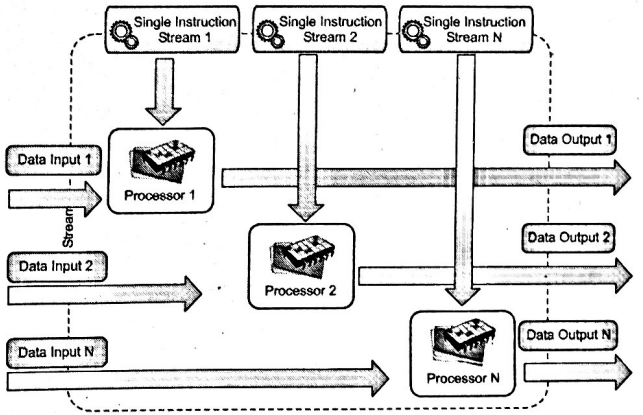

Fig. 1.6.4 : Multiple-instruction, multiple-data (MIMD) systems

- The MIMD machines are basically classified into shared-memory MIMDs and distributed-memory MIMDs based on how PEs are coupled to main memory.
- In the shared memory MIMD model (tightly coupled multiprocessor system), all PEs are connected to a single global memory and have access to all of them.
- In this model the communication between PEs is through shared memory, the modification of data stored by one PE in global memory is visible for all other PEs.
- Key representatives of shared memory MIMD systems are silicon graphics machines and SMP (Symmetric Multi-Processing) of Sun / IBM.

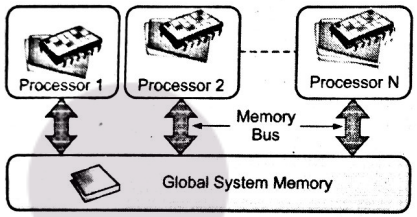

Fig. 1.6.5 : Shared memory MIMD model

- In distributed memory, all PEs in MIMD machines (loosely coupled multiprocessor systems) have a local memory.
- In this model, communication between PEs takes place through interconnected networks (inter-process communication channels, or IPCs).
- Networks connecting PE can be configured as trees, mesh, or as needed.

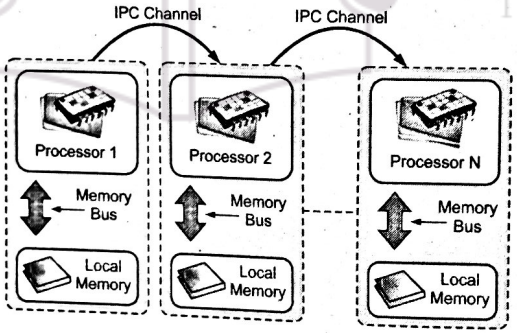

Fig. 1.6.6 : Distributed memory MIMD model

- The shared-memory MIMD architecture is more easy to program, but is less tolerant of failures and harder with respect to the distributed memory MIMD model.
- Failure in a shared-memory MIMD affects the entire system, while this is not the case for a distributed model, in which each PE can be easily separated.

- In addition, the shared memory MIMD architecture is less likely to scale because the addition of more PEs leads to memory conflicts.
- This is a situation that does not occur in the case of distributed memory, in which each PE has its own memory.
- As per the conclusion of practical outcomes and user requirement, the distributed memory MIMD architecture is superior to other existing models.

1.6.3 Types of Parallelism

GQ. 1.6.2. Describe the different levels of parallelism found in a computing system.

1. Bit-level parallelism

- This is a form of parallel computing that is based on the increasing size of the processor.
- It is used to reduce the number of instructions the system must execute to work on large-sized data.
- Example: Consider a scenario where an 8-bit processor must calculate the sum of two 16-bit integers.
- It must first sum 8 lower-order bits, then add 8 higher-order bits, thus requiring two instructions to perform the operation.
- A 16-bit processor can perform operations with just one instruction.

2. Instruction-level parallelism

- A processor can only address less than one instruction for each clock cycle step.
- These instructions can be re-ordered and grouped which are subsequently executed concurrently without affecting the program's outcome.
- This is called instruction-level similarity.

3. Task Parallelism

- It employs the decomposition of a task into a sub-task and then allocates each sub-task for execution.
- Processors perform concurrent tasks concurrently.

1.6.4 Applications of Parallel Computing

- Data bases and Data mining.
- Real time simulation of systems.
- Science and Engineering.
- Advanced graphics, augmented reality and virtual reality.

1.6.5 Advantages of Parallel Computing over Serial Computing

- This saves time and money because multiple resources working together will reduce time and cut potential costs.
- It can be technically impractical to solve major problems on serial computing.

- It can take the advantage of non-local resources when local resources are finite.
- Serial computing basically wastes potential computing power, thus making parallel computing a better work of hardware.

1.6.6 Limitations of Parallel Computing

- It used for communication and synchronization between many sub-tasks and processes that are difficult to achieve.
- Algorithms must be managed in such a way that they can be handled in a parallel system.
- The algorithm or program must have low coupling and high cohesion. But it is difficult to implement such programs.
- More technically proficient and expert programmers can code a similarity based program well.

1.7 Elements of Distributing Computing

UQ. 1.7.1 *Write short note on Distributed computing.* (MU - April 19, 5 Marks)

GQ. 1.7.2 *What is a distributed system? What are the components that characterize it?*

- As per the general definition of distributed system, we use the one proposed by Tennebaum : A distributed system is a collection of independent computers that appears to its users as a single coherent system.
- Communication is another basic aspect of distributed computing.
- Since distributed systems are made up of more than one computer that collaborates together, it is necessary to exchange some type of data and information between them, which is usually via the network.
- A distributed system is one in which components located on a networked computer communicate and coordinate their actions by passing only messages.
- As specified in this definition, distributed system components communicate with the passing of some type of message. It is a term that includes several communication models.

1.7.1 Components of a Distributed System

GQ. 1.7.3 *What is a distributed system? What are the components that characterize it?*

- The distributed system is the result of interactions of multiple components that traverse the entire computing stack from hardware to software.
- It emerges in collaboration with multiple elements that, by working together, give users the illusion of a single coherent system.

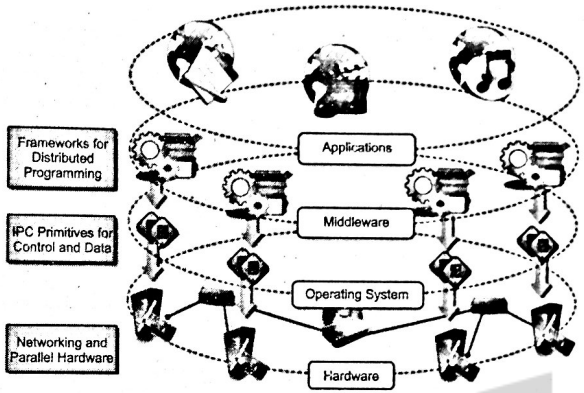

Fig. 1.7.1 : Layered View of Distributed System

Layers or Components involved in Distributed System

1. **Hardware**
   - At a very low level, computer and network hardware constitute physical infrastructure.
   - These components are directly managed by the operating system, which provides basic services for interprocess communication (IPC), process determination and management, and resource management in the context of file systems and local devices.
   - When taken together these two layers become a plat-form, above which special software is deployed to turn on a set of networked computers in a distributed system.
2. **Operating System**
   - The use of even more known standards at the operating system level and hardware and network levels allows easy exploitation of heterogeneous components and their organization into a consistent and uniform system.
   - For example, network connectivity between different devices is controlled by standards, which allows them to interact natively.
   - At the operating system level, IPC services are applied to standardized communication protocols such as Transmission Control Protocol / Internet Protocol (TCP/IP), User Datagram Protocol (UDP), or others.
   - Note that the hardware and operating system layers form a bare-bone infrastructure of one or more datatcenters, where racks of servers are deployed and connected together via high-speed connectivity.
   - This infrastructure is managed by the operating system, which provides basic capabilities of machine and network management.

3. Middleware

- The Middleware layer leverages such services to create a uniform environment for the development and deployment of distributed services.
- This layer supports programming paradigms for distributed systems. By relying on the services offered by the operating system, Middleware develops its own protocols, data formats, and programming language or framework for developing distributed applications.
- All of them have formed a uniform interface for distributed developers that is completely independent from the underlying operating system and hides all the oddities of the bottom layers.
- Core logic is then implemented in middle-ware that manages the virtualization layer, which is deployed on the physical infrastructure to maximize its use and provide a customizable runtime environment for applications.
- Middleware provides different features to application developers according to the type of services sold to customers.
- These features, offered through the Web 2.0-compliant interfaces, range from virtual infrastructure building and deployment to application development and runtime environments.

4. Applications

- The top of the distributed system stack is represented by applications and services designed and developed to use middleware.
- These can serve multiple purposes and often highlight their features as a graphical user interface (GUI) accessible via the Internet, locally or through a web browser.
- For example, in the case of cloud computing systems, the use of web technologies is strongly preferred, not only to interface distributed applications with the end-user, but to provide platform services aimed at building distributed systems.

1.7.2 Architectural styles for Distributed Computing

- Architectural styles are mainly used to determine the terminology of components and connectors that are used as examples of style together with a set of constraints on how they can be combined.
- Design patterns help to create a common sense within the community of software engineers and developers on how to formulate the relationships of components within an application and understand the internal organization of software applications.
- Architectural styles are similar to the overall architecture of software systems.
- Architectural style for distributed systems is helpful in understanding the different roles of components in the system and how they are distributed across multiple machines.
- Architectural styles into two major classes :
  - Software architectural styles
  - System architectural styles

- The first class relates to the logical organization of the software; the second class includes all those styles that describe the physical organization of distributed software systems in terms of their major components.
- A component represents a unit of software that encapsulates a function or a feature of the system.
- The 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.

1.7.2.1 Software Architectural Styles

GQ. 1.7.4 Write short note on Software architectural styles.

- Software architectural styles are based on the logical arrangement of software components. They are helpful because they provide an intuitive view of the whole system, despite its physical deployment.
- They also identify the main abstractions that are used to shape the components of the system and the expected interaction patterns between them.
- These models constitute the foundations on top of which distributed systems are designed from a logical point of view.

Category	Most Common Architectural Styles
Data-centered	Repository
	Blackboard
Data flow	Pipe and filter
	Batch sequential
Virtual machine	Rule-based system
	Interpreter
Call and return	Main program and subroutine call/top-down systems
	Object-oriented systems
	Layered systems
Independent components	Communicating processes
	Event systems

1.7.2.2 Data Centered Architecture

- These architectures identify data as a core element of software systems, and access to shared data is a core feature of data-centric architectures.
- Therefore, particularly in the case of distributed and parallel computing systems, data integrity is the overall goal for such systems.

1. Repository

- 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 the collection of independent components that operate on the central data.
- The manner in which independent components interact with the central data structure can be very heterogeneous.
- In particular, repository-based architectures differentiate and move into subcategories related to the choice of control discipline to apply to a shared data structure.
- Of particular interest are databases and blackboard systems.
- In the former group the dynamic of the system is controlled by independent components that, by issuing an operation at the central repository, trigger the selection of specific processes that work on the data.
- In Blackboard systems, the central data structure is the core trigger for selecting processes to execute.

2. Blackboard

- The blackboard architecture style has categories into three main components :
  - **Knowledge sources :** These are the units that update the knowledge base built into the blackboard.
  - **Blackboard :** It represents a data structure that is shared between knowledge sources and stores the application's knowledge base.
  - **Control :** Control is a collection of triggers and processes that control interactions with the blackboard and update the state of the Knowledge Base.
- Within this context scenario, sources of knowledge, which represent intelligent agents sharing the blackboard, react opportunistically to changes in the knowledge base, in much the same way as a group of experts in front of the blackboard.
- Blackboard models are popular and widely used for artificial intelligence applications, in which Blackboard maintains knowledge about a domain as assertion and rules that are entered by domain experts.
- These operate via a control shell that controls system problem solving activity.
- Special and successful applications of this model can be found in the domains of speech recognition and signal processing.

1.7.2.3 Data flow Architecture

- In the case of a data-flow architecture, the computation is control by the availability of data.
- With respect to data-centric styles, in which access to data is a core feature, data-flow styles explicitly include patterns of data flow, as their design is determined by the serial motion of the data from component to component, Which is the type of communication between them.
- Styles within this category differ in one of the following ways: how control is handled, the degree of consistency between components, and the topology that describes the flow of data.

1. Batch sequential

- Batch sequential style is characterized by executing one sequence of individual programs one by one.
- These programs are provided as input for the next program, the output generated by the last program after its completion that is most likely in the form of a file.
- It was so popular design in the mainframe era of computing and still finds application today.
- Example is, many distributed applications for scientific computing are specified by jobs expressed as a sequence of programs that, for example, pre-filter, analyze, and post-process data.
- Creating these steps using a batch-sequential style is very common.

2. Pipe and filter

- The pipe and filter style is a variation of the previous style to express the activity of a software system as a sequence of data changes.
- Each component of the processing chain is called a filter, and the relationship between one filter and the next is represented by a data stream.
- With respect to the batch sequential style, the data is processed sequentially and each filter processes the data as it becomes available on the input stream.
- Whenever one filter produces a consumable amount of data, the next filter can begin its processing.
- Filters typically do not have a state, identifying neither previous nor next filters, and they are associated with in-memory data structures such as first-in / first-out (FIFO) buffers or other structures.
- This type indexing is called pipelining and introduces concurrency in the execution of the filter.
- A classic example of this architecture is the microprocessor pipeline, under which multiple instructions are executed at the same time by completing a separate phase of each of them.
- We can find out the steps of instructions as filters, while data streams are represented by registries that are shared within the processor.
- Data-flow architectures are optimal when the system being designed embodies a multistage process, which can be clearly identified in a collection of individual components that need to be orchestrated simultaneously.
- Within this case, components have well-defined interfaces showing input and output ports, and connectors are represented by datastreams between these ports.

1.7.2.4 Virtual Machine Architectures

- Virtual machine classes of architecture styles are characterized by the presence of anabstract execution environment (usually referred to as a virtual machine) that emulates features that are not available in hardware or software.
- Applications and systems are implemented on top of this layer and become portable in various hardware and software environments, unless the implementation is done with a virtual machine interface.
- The following is the typical interaction flow for a system implementing this pattern : The program (or application) defines its operation and state in an abstract format, which is interpreted by the virtual machine engine.

- This interpretation of a program constitutes its execution. In this case it is quite common that the engine maintains an internal representation of the state of the program.
- Very popular examples of this category are rule-based systems, interpreters, and command-language processors.

1. Rule-based system

- This architecture is characterized as representing an abstract execution environment in the form of an implementation engine.
- Programs are expressed as rules or predicted to be correct.
- Input data for applications is usually shown by a set of assertions or facts that the inference engine uses to activate rules or implement predicates, thus converting the data.
- The output can either be the product of rule activation or claims to be true for the given input data.
- The set or predicate of rules identifies the knowledge base that can make assumptions about the system.
- This approach is quite strange, as it allows the system or domain to be expressed in terms of its behaviour rather than components.
- Rule-based systems are very popular in the field of artificial intelligence.
- Practical applications can be found in the area of process control, where rule-based systems are utilized to monitor the state of physical devices, which are collected from sensory data and processed by PLCs1 and by implementing alarms when particular condition on the sensory data apply.
- Another interesting use of rule based systems can be seen in the networking domain: Network Intrusion Detection Systems (NIDS) often rely on a certain rules to identify unusual behaviours associated with potential intrusion into computing systems.

2. Interpreter

- The main feature of the interpreter style is the presence of an engine, which is used to interpret a pseudo-program expressed in a form which is acceptable for the interpreter.
- The interpretation of the pseudo-program constitutes the execution of the program itself.
- The systems modelled with respect to this style exhibit four core components: the interpretation engine, which executes the main activity of this style, an internal memory that consists of the pseudo-code to be interpreted, a representation of the current state of the engine, and also a representation of the current state of the program being executed.
- This model is very useful in designing virtual machines for high-level programming (Java, C#) and scripting languages (Awk, PERL, and so on).
- Within this case, the virtual machine closes the gap between the end user abstractions and the software/hardware environment in which such abstractions are executed.
- Virtual machine architectural styles are characterized by an indirection layer between applications and the hosting environment.
- This design has the lots of advantages of decoupling applications from the underlying hardware and software environment, but at the same time it has some disadvantages, such as a performance bottlenecked.

- Other issues may be related to the fact that, by providing a virtual environment, particular features of the underlying system may not be accessible.

1.7.2.5 Call and Return Architectures

- This category finds all systems that are organised into components usually connected together by method calls.
- The process of systems modelled in this manner is characterized by a sequence of method calls whose overall execution and composition finds the execution of one or more operations.
- The internal organization of components and their connections may be different.
- But, it is possible to find out three major subcategories, which differentiate by the manner the system is structured and how methods are invoked: top-down style, object-oriented style, and layered style.

1. Top-down style

- This architectural style is fairly representative of systems developed with imperative programming, which leads to a divide-and-conquer approach to problem solving.
- Systems developed based on this style are composed of a large core program that performs its tasks by implementing subprograms or processes.
- The components of this style are process and subprogram, and the connection method is call or invoke.
- The calling program passes information in the form of parameters and receives data from the return value or parameter.
- Method calls can extend beyond the limits of a single process by taking advantage of technologies for remote process calls, such as remote procedure calls (RPC) and all its descendants.
- The overall structure of program execution at any given time is represented in the form of a tree, the root of which organizes the main function of the main program.
- This architectural style is very simple from a design point of view but still, it is difficult to maintain and manage large systems.

2. Object-Oriented Style

- It is a wide range of architectural style systems designed and implemented by leveraging the essence of Object-Oriented Programming (OOP).
- Systems are defined in terms of classes and applied in terms of objects.
- Classes specify the types of components by defining data that represent their state and the operations to be performed on these data.
- One of the important benefits of the top-down style is that there is a coupling between the data and the operations used to manipulate them.
- Object instances become liable for hiding their internal state representation and protecting its integrity while providing operations to other components.
- This results in a better decomposition process and more management-capable systems.

- The two main disadvantages of this style are : every object needs to know the identity of an object if it wants to operate on it, and shared objects need to be carefully designed to ensure the stability or consistency of its state .

3. Layered Style

- Layered system styles provide the design and implementation of software systems in terms of layers, providing a different level of abstraction of the system.
- Each layer typically operates with at most two layers: one that provides a lower abstraction level and one that provides a higher abstraction layer.
- Particular protocols and interfaces specific how adjacent layers interact. It is technically possible to model such systems as a stack of layers, one for each level abstraction.
- So, components are layers and connectors are interfaces and protocols used between adjacent layers.
- A user or client usually interacts with the layer at the highest abstraction, which uses the services of the lower layer, to perform its activity.
- This process can be repeated until the lowest layer is reached.
- Also it is possible to have the opposite behaviour: events and callbacks from the lower layers can trigger the higher layer's activity and propagate information through the stack.
- Layered style has advantages, as is the case for object-oriented style, it supports a modular design of the system and allows us to decompose the system according to different levels of abstractions by combining together all the operations related to a particular level.
- Layers can be replaced till they conform to the expected protocols and interfaces, thus making the system flexible.
- The main disadvantage is formed by the lack of extensibility, as it is not possible to change the protocol and add layers without interfaces between layers.
- This also complicates the addition of operations.
- Examples of layered architectures are the modern operating system kernel and the International Standards Organization / Open System Interconnection (ISO / OSI) or TCP / IP stack.

➲ 1.7.2.6 Independent Components

- The independent components have their own life cycles interact with each other to perform their activities.
- There are two main categories within this class, communication processes and event systems, that differentiate the way interactions are managed between components.

1. Communicative Processes

- In this, components are shown by independent processes that take advantage of IPC features for coordination management.
- It is an abstraction that is quite suitable for distributed modelling, being distributed via a network of computing nodes, necessarily composed of multiple concurrent processes.
- Each process provides other processes with services and can take advantage of services exposed by other processes.

- The conceptual organization of these processes and the way they communicate varies according to the specific model, i.e. peer-to-peer or client / server.
- Connectors are identified from the IPC facilities used by these processes to connect.

2. Event System

- In event system, the components of the system are loosely coupled and connected.
- In addition to showing operations for data and state manipulation, every component are also publishes (or declares) a collection of events with which other components may be registered.
- In general, other components provide a callback that will be executed when the event is activated.
- At the time of a component's activity, a particular runtime state can activate one of the exposed event, so triggering the execution of callbacks registered with it.
- Event activation can occur with relevant information that can be used to handle the event in a callback.
- This information can be transferred as an argument to callbacks or by utilizing some shared repository between components.
- Event-based systems have become quite popular, and support for their executing is provided at the API level or programming language level.
- The fundamental advantage of such an architectural style is that it promotes the development of open systems: new modules can be added and easily integrated into the system as long as they have compatible interfaces for registering events.
- This architectural style have some of the limitations shown for top-down and object-oriented styles.
- First, the method invocation format is implicit, and the relationship between the caller and the callee and is not hard-coded.
- This allows a lot of flexibility since adding or removing a handler for events without changing the source code of the applications.
- Second, the event source does not need to aware of the identity of the event handler to implement the callback.

- The main disadvantage of this a style is that it surrenders control over system computation.
- When the component triggers an event, it is not aware of how many event handlers will be invited and also whether there are any registered handlers.
- This information is only available at runtime and from a static design point of view, it becomes more complicated to identify connections between components and to argue about the accuracy of interactions.

➲ 1.7.2.7 System Architectural Styles

GQ. 1.7.5 Write short note on System architectural styles.

- System architectural styles cover the physical organization of components and processes on a distributed infrastructure.
- They used to provide a set of reference models for the deployment of such systems and engineers have a common terminology in not only describing the physical layout of the system, but also identifying the key advantages and drawbacks of a given deployment and whether it is applies to specific class of applications.

1.7.2.8 Client/Server

GQ. 1.7.6 *Explain Client/Server Architecture.*

- Client-Server Architecture is a part of distributed system where the task of client and server is separated.
- Clients send request for the services or resources and Server means the provider of services or resources.
- The server host several applications at its end for providing or sharing resources to its clients whenever they requested for it.
- Client and server can be installed on the same computer or may be at the remote location and access via a network.
- Client Server architecture is centralized resource management system where Server has all the resources.
- So the server should be highly secured and scalable to provide proper response to the all clients of the system.
- Client/Server Architecture is nothing but Service Oriented Architecture, means client can be remained busy with all its other work, in mean time server can provide the response.

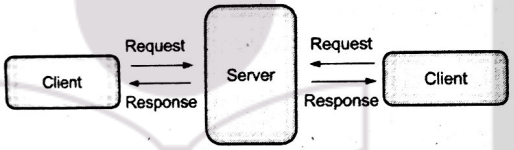

Fig. 1.7.2 : The Client-Server Architecture

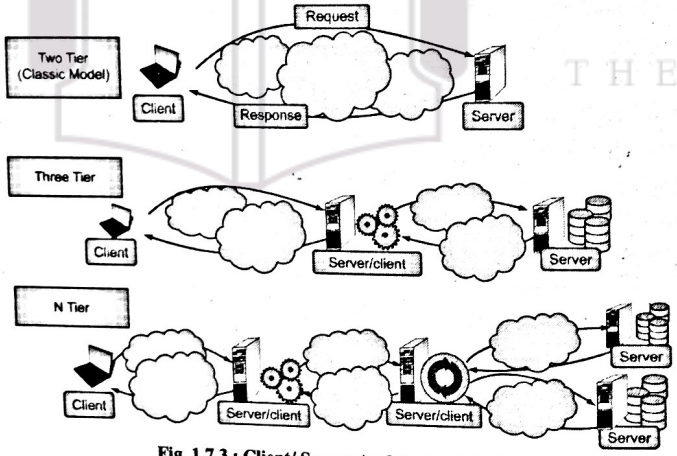

Fig. 1.7.3 : Client/ Server Architectural Styles

- **Thin-client model :** In Thin-client model, the load of data processing, transformation and every computation tasks are put on the server side, and the client has a light implementation, mostly related to receiving and returning data without bothering about further processing.

- **Fat-client model :** In Flat-client model, the client component is also responsible for processing, computation and exchanging data before sending data to the server, while the server has a relatively lightweight implementation that is mostly concerned with managing data usage.
- Three key components in the client-server modelare :Presentation, Application Logic, and DdataStorage.
- Presentation, application logic and data maintenance can be considered as conceptual layers, more properly known as tiers.
- Mapping of conceptual layers and their physical implementations in modules and components allows differentiating between various types of architectures, known as Multitier Architectures.

1.7.2.9 Two Major Types of Tier Architecture Classes

1. Two-Tier Architecture

- This architecture divides the system into two levels, one located in the client component and the other on the server.
- The client is liable for the presentation tier by providing a user interface. The server focuses the application logic and data store into a single tier.
- The server component is typically deployed on a powerful machine, which is capable of executing application logic to process user requests, access data, and provide a client with feedback.
- This architecture is best suited for systems of limited size and suffers from scalability problems.
- In particular, server performance may decrease dramatically as the number of users increases.
- Another limitation is due to maintaining, managing and using dimensions of data, which can be prohibitive for a single compute node or too large to serve customers with satisfactory performance.

2. Three-tier architecture / N-tier architecture

- The three-tiered architecture is used to separates data presentation, application logic and data storage into three levels.
- This architecture is generalized to the N-tier model, when it is necessary to further divide the stages that compose application logic and storage tiers.
- This model is basically more scalable than two-tier because it is possible to distribute tiers across multiple computing nodes, thus isolating performance constraints.
- At the same time, these systems are also more complex to understand and manage.
- A classic example of a three-tiered architecture is formed by a medium-sized web application that relies on a relational database management system to store its data.
- In this scenario, the client component is served by a web browser, which tiers the presentation, while the application server encrypts the business logic tier, and a database server machine (replicated for high availability) holds data storage.
- Application servers which rely on third-party (or external) services to fulfil the client requests are examples of N-tier architectures.

1.7.2.10 Features of Client Server Architecture

- Divides Processing between different machines
- Non –critical transaction and data are processed on client system.
- Critical transactions are processed and final data is stored on server machines.
- It improves client computer for data input and presentation purpose.
- It enables and improves server for data processing & storage.
- It provides scaling horizontally where multiple dedicated servers, works like distributed manner, each has its own capabilities and processing power.
- It scales vertically where multiple processors can be added to serve requests from increasing number of clients.
- It minimizes data redundancy.

1.7.2.11 Advantages of Client Server Computing

- All the needed data are concentrated in one place i.e. server. Therefore it is easy to protect data and provide authorization and authentication.
- The server must not be physically located near the clients. Data can still be accessed efficiently.
- It is easy to change, upgrade, or move nodes in the client server model because all nodes are independent and only request data from the server.
- Not all nodes i.e. clients and servers can be built on the same platform, yet they can easily provides data transfer.

1.7.2.12 Disadvantages of Client Server Computing :

- If all clients simultaneously request data from the server, it may be overloaded. This can cause network congestion.
- If the server fails due to any reason, none of the client's requests can be fulfilled. This results in client server network failure.
- The installation and maintenance of client server model is quite expensive.

1.7.2.13 Peer-To-Peer

- The peer-to-peer model introduces a symmetric architecture in which all components, known as peer, play the similar role and consolidate both client and server capabilities of the client / server model.
- In more perfect words, each peer acts as a server when it processes requests from other peers and as a client when it requests from other peers.
- With relation to the client / server model that divides the responsibilities of IPC between server and client, the peer-to-peer model attributes the same responsibilities to each component.
- So, this model is quite relevant to highly decentralized architecture, which can scale better with dimension of number of peers.
- The main disadvantage of this approach is, the management of the implementation of the algorithm is more complicated than the client / server model.

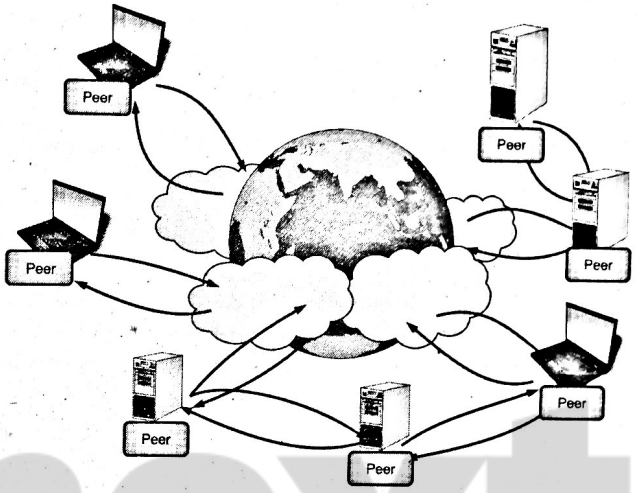

Fig. 1.7.4 : Peer-to-peer architectural style

- The most relevant example of peer-to-peer systems is formed by file-sharing application such as BitTorrent, Guttella and Kazaa.
- Apart from the differences between these networks in coordinating nodes and sharing files and information on their locations, all of these provide a user client that is at the same time a server that provides files to other peers and a client can download files from other peers.
- To locate an incredibly large number of peers, various architectures have been designed that are slightly different from peer-to-peer models.
- For example, not all peers in Kazaa have the same role, and some of them are used to group access information of a group of peers.

1.7.2.14 Characteristics of Peer to Peer Computing

- Peer-to-peer networks are typically built by a group of a dozen or fewer computers. These computers store their data using personal security but along with share data with all other nodes.
- Nodes in peer-to-peer networks both use and provide resources. Therefore, if nodes grow, the ability to share resources from peer to peer networks is also increases. This is different from client server networks where the server is overwhelmed as nodes grow.
- Since nodes in peer-to-peer networks works as both clients and servers, it is quite complicated to provide adequate security for nodes. This can results in denial of service attacks.
- Most of the modern operating systems like Windows and Mac OS have software to implement peer-to-peer networks.

1.7.2.15 Advantages of Peer to Peer Computing

- Each and every computer in a peer to peer network manages itself. Therefore, it is quite simple to set up and maintain a network.
- In a client server network, the only server handles all client requests. This provision is not needed in peer-to-peer computing and saves server costs.
- Peer to peer networks are easier to scale and add more nodes. This only increases the data sharing capacity of the system.
- No node in a peer-to-peer network is dependent on others for its functioning.

1.7.2.16 Disadvantages of Peer to Peer Computing

- Backing up the data is difficult because it is stored in various computer systems and there is no provision of central server.
- Providing overall security in a peer-to-peer network is difficult because each system is independent and has its own data.

1.7.3 Models for Interprocess Communication

GQ. 1.7.7 Discuss about Models for inter-process communications.

- Distributed systems are made up of a collection of concurrent processes interacting with each other through a network connection.
- So, IPC is a fundamental aspect of distributed systems design and implementation.
- IPC is either used for the exchange of data and information or to coordinate the activity of processes.
- IPC is that which binds together the various components of a distributed system, thus making them function as a single system.
- There are many different models in which processes can interact with each other.
- These maps in different abstracts for IPC.
- The most relevant that we can mention is shared memory, remote procedure call (RPC), and message passing.
- At a lower level, IPC is realized through basictools of network programming.
- Sockets are the most well-known IPC primitive for implementing communication channels between distributed processes.
- They provides interaction patterns, which mimic client / server abstraction at lower levels and are based on the request-reply communication model.
- Sockets facilitates the transferring core capability with an order of bytes, which at higher levels is converted into a more meaningful representation (like process parameters or return values or messages).
- Such a powerful abstraction allows engineers to focus on logic-coordinated distributed components and information that they exchange rather than networking information.
- Both these elements identify the model for IPC.

- Inter process communication (IPC) is a system which permits processes to communicate or coordinate with one another and provide synchronization between their activities.
- Two different machines with different operating system can communicate with each other via Interprocess communication.

1.7.3.1 Message-based Communication

- The abstraction of the message has played vital role in the development of models and technologies enabling distributed computing.
- The Culorisstated a distributed system as "one in which components located at networked computers communicate and coordinate their actions only by passing messages."
- Many distributed programming paradigms ultimately use message-based communication, regardless of the abstract presented to developers, for programming the interactions of distributed components.
- It is very important to understand that the concept of a message is a fundamental abstraction of IPC, and can be used explicitly or implicitly.

1. Message passing

- In the distributed systems two applications can communicate with each other by means of exchanging messages.
- Messaging provides Asynchronous, high-speed, program-to-program communication with reliable or trusted delivery.
- Two operations involved in message communication : send and receive, specified in form of source, destinations and messages.
  - **Serialization or Marshalling :** is the mechanism of divide message into small piece, so than it can be transferred easily over a network in the form in which it can be reconstructed perfectly at receiver side.
  - **Deserialization or Unmarshalling :** is the mechanism of combining gathered pieces of message in the form of bytes at receiver side to generate original message same as sender's message.

2. Distributed Objects

- It is an implementation of the RPC model for object-oriented methodology and utilizes this feature for remote invocation of methods exposed by objects.
- Each process registers a group of remotely accessible interfaces.
- Client processes can request a pointer to these interfaces and invoke methods available through them.
- The underlying runtime infrastructure is to convert the call of the local method into a request for a remote process and gather the result of execution.
- Communication between the caller and the remote process is done through the message. In relation to RPC models that are stateless by design, distributed object models presents object state management and lifetime complexity.
- Remotely executed methods operate in the context of an instance, which can be created for the sole execution of the method, exists for limited intervals of time, or is independent of the existence of requests.

- Examples of distributed object infrastructure are Component Object Model (COM, DCOM and COM1), Common Object Request Broker Architecture (CORBA), .NET remoting) and Java Remote Method Invocation (RMI).

3. Distributed Agents and Active Objects

- Despite the presence of requests, programming paradigms based on agents and active objects are included by the definition of the presence of instances, whether they are agents of objects.
- That means, objects have their own control thread, which allows them to perform their own activity.
- These models often make explicit use of messages to trigger the execution of methods, and messages have a more complex semantics attached to them.

4. Web Services

- The web service technology provides implementation of the RPC methodology over HTTP, thus allowing the interaction of components to be developed with different technologies.
- A web service is shown as a remote object hosted on a web server, and the method invocation is converted into an HTTP request, with specific protocols such as Simple Object Access Protocol (SOAP) or Representative State Transfer (REST) is packaged systematically.

1.7.3.2 Point-to-point Message Model

- The point-to-point messaging mechanism allows clients to transfer and receive messages synchronously as well as asynchronously.via queues.
- The point-to-point messaging scheme gives reliable communication for multi-staged applications.
- This technique was originally a pull-based or polling-based, where messages store in queue and receives it from queue, instead of automatically pushing message directly to receiver.

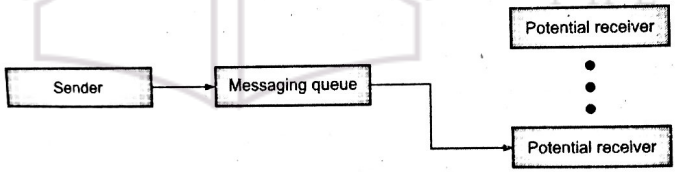

Fig. 1.7.5 : Point-to-Point Messaging

- I finds in two major sub-categories: direct communication and queue-based communication. In the former, the message is sent directly to the receiver and processed at the time of reception.
- In the latter, the receiver maintains a message queue in which the received messages are kept for subsequent processing.
- The point-to-point messaging architecture is useful for implementing systems that are mostly based on one-to-one or many-to-one communication.

1.7.3.3 Publish-and-Subscribe Message Model

- This mechanism is a highly scalable way of messaging.
- If a publisher application publishes messages on particular topic, any number of subscribing applications can subscribe to this topic and receive the messages published by the publishing application.
  - Publishers publish messages on particular topics.
  - A message server maintains track of all the messages, and all its currently active and durable subscribers. The message server gives a security for the messaging system by providing authentication and authorization.
  - Whenever messages are published on a particular topic, they are transferred to all of its subscribers.
- A message will be provided to all subscribers who have registered for the related event. There are two major strategies for sending this event to customers:
  - **Push strategy :** In this strategy, it is the responsibility of the publisher to notify all subscriber. For example, with a method invocation.
  - **Pull strategy :** In this strategy, the publisher only provides messages for a specific event, and it is the responsibility of subscriber to check whether there are messages on recorded events.
- The published-and-subscribe paradigm is very suited for implementing systems based on one-to-many communication models and makes it easy, the implementation of indirect communication pat-turn.
- In fact, it is not necessary for the publisher to know the identity of the subscribers in order to create the communication.

1.7.3.4 Request-reply Message Model

- On many events applications require that request/reply messaging transaction be performed.
- It can happen in two ways, means: synchronous request/reply messaging and asynchronous request/reply messaging.
- Synchronous request/reply messaging is mostly needed when trying to interact with a Web service client that blocks and waits for a synchronous response to get from receiver.
- In the asynchronous mode of request/reply messaging, the sender expects the reply to arrive at a later time and continue its work with interruption.
- From figure it is clear that in a simple request /reply asynchronous messaging scenario delivery channels are not bidirectional.
- To perform a request/reply transaction the sender and receiver should use two channels : one for the request and one for the reply.

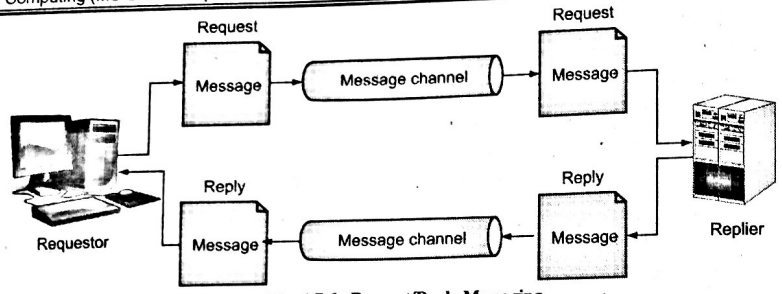

Fig. 1.7.6 : Request/Reply Messaging

- The point-to-point message methodology is mostly based on request-reply interactions, particularly in the case of direct communication.
- Publish-and-subscribe models are somewhat based on request-replies because they rely on information.

1.8 Technology of Distributing Computing

Distributed Computing Technologies that provide detailed implementations of interaction models, which mostly rely on message-based communication. These technologies are remote procedure call (RPC), distributed object frameworks, and service-oriented computing.

1.8.1 Remote Procedure Call (RPC)

UQ. 1.8.1 *Explain RPC in detail.* (MU - April 19, 5 Marks)

GQ. 1.8.2 *What is RPC?*

- A remote procedure call is one of the Interprocess Communication mechanisms that are utilized for client-server applications. It is also called as a function call or a subroutine call.
- RPC translates a client request or call and sends to server.
- After that server receive the request and provide appropriate response to the client.
- All this happens in synchronous way means, client is blocked till server is processing the request or call and it can resume its execution when server is finished with its processing of call.
- A client has a request message or call that the RPC translates and sends to the server. This request may be a procedure or a function call to a remote server.
- When the server receives the request, it sends the required response back to the client.
- At the time server is processing, the client need to blocked and only resumed execution after the server is finished.

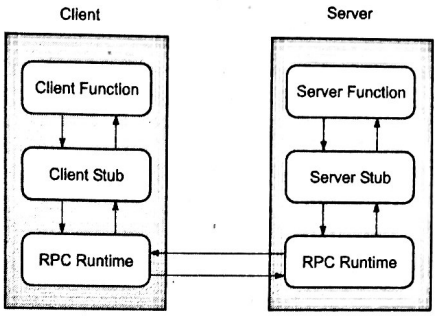

Fig. 1.8.1 : Remote Procedure Call

1.8.1.1 The Steps Involving in RPC

- The client invokes the Client stub.
- The client stub generates the system call to transfer the message to the server along with appropriate parameters.
- Client operating system provides support for transfer of message between client and server.
- Server operating system passes message to server stub.
- Server stub removes the parameters from message.
- Then, the server stub invokes server procedure to process it and generate the responses.
- The server OS transfer response to the client.
- Then client OS receive the response and passes response to client stub.
- After that client stub process the response.

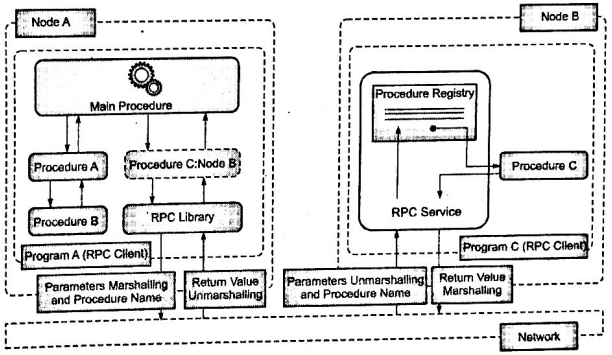

Fig. 1.8.2 : The RPC reference model

- A key aspect of RPC is marshalling, which finds out the process of returning a parameter and also returns values in a form that is more suitable to be carried via a network through a sequence of bytes.
- The word unmarshalling means the opposite process.
- The marshalling and unmarshalling are carried out by the RPC runtime infrastructure, and client and server user code is not required to perform these tasks.
- On the other hand, the RPC runtime is responsible not only for parameter packing and unpacking, but also for handling request-reply interactions that occur between the client and server process in a completely transparent manner.
- So, developing a system that leverages RPC for IPC involves the following steps :
  - Design and implementation of server processes exposed to remote calls.
  - Registration of remote processes with the RPC server on the node where they will be made available.
  - Design and implementation of client code invoking remote procedures.
- 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.

1.8.1.2 Advantages of RPC

- RPC provides support for both process oriented and thread oriented models.
- In this, the message passing technique is hidden from the user.
- It minimizes effort of re-writing and re-developing the code.
- It can be utilize in both local and distributed environment
- It enhances the performance by omitting many protocol layers.

1.8.1.3 Disadvantages of RPC

- It's not a standard technique, which can be implemented in many ways.
- In terms of hardware architecture, this is not flexible technology, just based on interaction
- Scheduling cost increases because of context switching.

1.8.2 Distributed Object Frameworks

GQ. 1.8.3 What is Distributed Object Framework?

- Distributed object frameworks enhance object-oriented programming systems, allowing objects to be distributed across a heterogeneous network and provide features so that they can function identically, as they were in the same address space.
- Distributed object frameworks take advantage of the basic mechanisms introduced with RPC and extend this to enable remote invocation of object methods and tracking in terms of objects provided through network connections.

- Regarding the RPC model, the infrastructure manages instances that are shown through wellaware interfaces rather than procedures. So, the following is the normal interaction pattern :
  - The server process maintains a registry of active objects, which that are provided to other processes. As per the specific implementation, active objects can be published by utilizing interface definitions or class definitions.
  - The client process, using a given address scheme, receives a reference to the active remote object. This reference is represented by pointer for an instance that is a shared type of interface and class definition.
  - The client process used to invokes methods on the active object by calling it through the previously received reference. According to RPC the parameter and the return value can be marshalled.

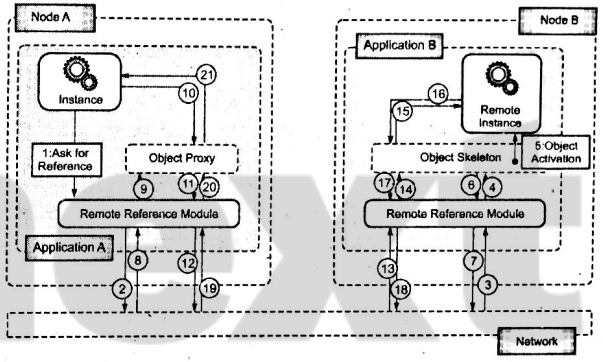

Fig. 1.8.3 : The Distributed Object Programming Model

- It provides the illusion of interaction with a local instance when implementing remote methods.
- This is done by a methodology known as a proxy skeleton.
- Proxies and skeletons always comes in a pair: the server process maintains the skeleton component to execute methods implemented remotely and the client maintains the proxy component, providing its hosting environment.
- Allows invoicing methods remotely through the interface.
- Transparency of RMI is achieved using one of the basic properties of object-oriented programming: inheritance and subclass.
- Both proxy and active remote objects shows the same interface, defining the set of methods that can be called remotely.
- On the client side, a runtime object is generated ina subclass of the type published by the server.
- This object translates the local method invocation in the RPC call to the as per method on the remote active object.
- On the server side, while an RPC request is received, it is unpacked and the method call is sent with the client to whom the request is issued.

- Whenever the execution of the method is completed on the server, the return values are packed and dispatched back to the client, and the local method calls the proxy return.
- Distributed object frameworks propose objects as first-class entities for IPC.
- They are principle gateways for carry out remote methods, but also can be passed as parameters and return values.
- This is an interesting problem, because object instances are complex instances that encapsulate a state and can be referenced by other components.
- This passing an object as a parameter or return value involves duplication of the instance on another execution context.
- This operation leads to two different objects whose state develops independently.
- Duplication becomes necessary because process boundaries have to be crossed which is needed for instance.
- This is a key aspect in designing distributed object systems, as it can lead to inconsistencies.
- An alternative to this regular procedure, called the marshalling by value, is the marshalling by reference.
- In this second case, object instance is not a duplicate and a proxy is created either on the server side (for parameters) or on the client side (for return values).
- The marshalling by reference is a more complex mechanism and typically involves a greater burden on the runtime infrastructure as remote references have to be tracked.
- Being more complicated and resource demanding, marshalling by reference should only be used when the duplication of parameters and return values results in unpredictable and inconsistent behaviour of the system.

1.8.2.1 Object Activation and Lifetime Management

- The first element must be considered is the activation of the object, which is nothing but the creation of the remote object.
- Different strategies can be utilized to manage object activation, from which we can distinguish two primary classes: server-based activation and client-based activation.
- In server-based activation, an active object is instantiated in the server process and registered as an instance that can be defined beyond process boundaries.
- In this case, an active object has its own life and sometimes invocation of a remote method results in methods of executes.
- In client-based activation, the active object is not actually present on the server side.
- It is created when a request comes from a customer to call the method.
- This scenario is basically more suitable when the active object is considered stateless and must exist for the sole purpose of implementing methods from remote clients.
- Consider example, if the remote object is only a gateway to access and modify other components hosted within the server process, then the more efficient pattern is the client-based activation.

- The second element is considered as the lifetime of distant objects.
- In the scenario of server-based activation, the lifetime of an object is typically user-controlled, because the activation of a remote object is explicit and controlled by the user.
- In a client-based activation, the creation of a remote object is implicit, so its lifetime is controlled by some policy of the runtime infrastructure.
- The different policies can be considered. The simplest means the creation of a new instance for each method invoked.
- This solution is actually demanding in the context of object instances and is typically integrated with some lease management method that allows objects to be reused for later method invocations, if they are occur in a given time interval (lease).
- Another approach may consider having only one instance at a time, and then the lifetime of the object is controlled by the number and frequency of method calls.
- These are properties that are now supported to some extent in almost all frameworks for distributed programming, as they are necessary to understand the behaviour of distributed systems.
- Object Activation and Lifetime Management aspects are becoming fundamental in designing components that are accessible from other processes and that maintain states.
- How many objects representing the same component have been created and for example, how long they are necessary to record inconsistency because of erroneous updates to the instance internal data.

1.8.2.2 Examples of Distributed Object Frameworks

- Common object request broker architecture (CORBA)
- Distributed component object model (DCOM/COM1)
- Service-oriented computing
- Java remote method invocation (RMI)
- .NET remoting.

1.8.3 Service Orientation and Cloud Computing

GQ. 1.8.4 Write short note on Service-Oriented Computing.

- Service-oriented computing establishes distributed systems in the context of services, which represent the main abstraction for building systems.
- Service orientation defines applications and software systems as an aggregation of services, which are coordinated within a service-oriented architecture (SOA).
- Yet there is no designed technology for developing service-oriented software systems, web services are the *de facto* standard for SOA.
- Web services, the basic component enabling cloud computing systems, leverage the Internet as the primary interaction channel between users and the system.

1.8.3.1 What is a Service?

GQ. 1.8.5 What is Service and key features of it?

- A service encapsulates a software component which gives several coherent and related functionalities that can be reused and integrated into larger and complex applications.
- The word service is a general abstraction that involves many different implementations by utilizing different technologies and protocols. Don Box describes four key features regarding a service :

1. Boundaries are explicit

- A service-oriented application is typically made up of services that span various domains, trust s, and execution environments.
- As crossing such boundaries is costly, service invocation is explicit by design and often takes advantage of message passing.
- In relation to distributed object programming, because of which remote method invocation is transparent, interaction with a service in a service-oriented computing environment is explicit and a service's interface is kept to a minimum to promote its reuse and simplify interaction is.

2. Services are autonomous

- Services are components which are exist to provide functionality and are aggregated and coordinated to create more complex systems.
- They are not created to be part of a particular system, but they can also be integrated into multiple software systems at the same time.
- As per the object orientation, which consider that the deployment of applications is atomic, service orientation seems the matter to be an exception rather than a rule and focuses on the design of the service as an autonomous component.
- The notion of autonomy also affects the way services fail.
- The services works in an unknown environment and interact with third-party applications.
- So, minimal assumptions can be made regarding such environments : Without notification an applications may fail, messages may be distorted, and client may become unauthorized.
- The service-oriented design addresses these issues using transactions, redundant deployment, durable queues and failure, and trust relationships can be managed administratively between different domains.

3. Services share schema and contracts, not class or interface definitions

- Services are not expressed in form of a class or interface, as in object-oriented systems, but they define themselves in terms of schemas and contracts.
- It advertises a contract describing the structure of the messages.
- It can transferor receive on their order and additional constraints, if any.
- Because they are not defined in the form of types and classes, services can be consumed more easily in a wider and heterogeneous environment.

- At the same time, a service orientation requires that contracts and schemas remain constant over time, as it is possible for changes to be propagated to all of its potential customers.
- To solve this problem, contracts and schemas are defined in a form that allows services to develop without breaking previously deployed code. Technologies like XML and SOAP provide appropriate tools to support such property rather than class definition or interface declaration.

4. Services compatibility is determined based on policy

- Service orientation distinguishes structural compatibility from semantic compatibility.
- The Structural compatibility is depend on contracts and schemas and can be validated or applied by machine-based techniques.
- Semantic compatibility is defined as policies that specify capabilities and requirements for a service.
- Policies are established in terms of expressions that must be true to enable the normal operation of a service.

1.8.3.2 Service-Oriented Architecture (SOA)

- Service orientation is the main reference model for cloud computing systems.
- This approach adopts the services concept as the main building blocks of application and system development.
- Service-oriented computing (SoC) supports the development of rapid, low-cost, flexible, interoperable and developable applications and systems.
- Service-oriented computing presents and expands two important concepts, which are also fun-damaging to cloud computing: quality of service (QoS) and software-as-a-service (SaaS).
- Service-oriented computing is based on a service-oriented architecture (SOA).
- A service-oriented architecture (SOA) is a construction format in computer software design in which application parts or tolls provide services to other parts by using a standard communications protocol, basically via internet.
- The standards of service-orientation are not based on any vendor, product, tools or technology.
- SOA simply provides it easier for software components over different networks to work with one and another.
- Web services which are based on the SOA architecture inclined to make web service extremely independent.
- The web services themselves can interchange data with one and other.
- Because of the fundament principles on which they are made, they are not depending on any sort of human interaction and also don't need any code alteration.
- It guarantees that the web services on a network can communicate with each and other flawlessly.

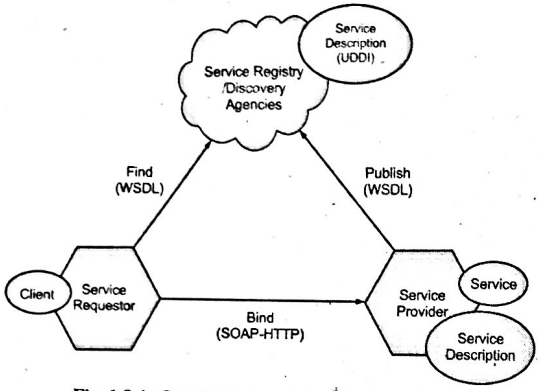

Fig. 1.8.4 : Service Oriented Architecture (SOA)

1.8.3.3 Features of SOA

GQ. 1.8.6 Describe the main characteristics of a service orientation.

- **Standardized Service Contract :** Services bound to a service description. A service should have some kind of description which specifies what the service is about. This makes it simpler for client applications to know what the service going to do.
- **Loose Coupling :** Means the less dependency on one and other. This is one of the fundamental characteristics of web services which simply say that there should be as less dependency as possible between the web services and the client invoking that web service. Assume that the case of tight coupling, in which, if the service functionality changes at any point in time, it should interrupt the client application and force it do some sort of changes.
- **Service Abstraction :** Services encapsulates the logic to hide it from the outside world. The service must not uncover how it implements or executes its functionality; it should simply tell the client application that what it does and not, and how it does it.
- **Service Reusability :** For the purpose of maximizing reusability, logic is partitioned into different services. For any development company re-usability is an important topic because definitely one wouldn't want to spend plenty time and effort making the same code repeatedly across multiple applications which require them. So, once the code for a web service is build it must have the ability to work with different types of applications.
- **Service Autonomy :** Services must have control over the logic they consists of. The service recognize everything on what functionality it offers and hence must also have complete control over the code it contains.
- **Service Statelessness :** Basically, services must be stateless. Means services should not carry information from one state to another. This would need to be done from either the client application. For example, an order placed on a shopping site. Now you can have a web service which provides you the price of a particular item. But if the items are added to a shopping cart by clicking on 'add to cart' button and then

web page navigates to the page where you do the payment, now the duty of transferring the price to the payment page should not be done by the web service. Instead, it should be done by the web application.

- **Service Discoverability :** Services can be discovered usually in a service registry. We know concept of the UDDI, which performs a registry which can hold information about the web service.
- **Service Composability :** Services breakdowns huge problems into little modules or problems. So one should not combine all functionality of an application into a single.
- **Service Interoperability :** Services should use standards or rules that permit different subscribers to use the service. In that means service or code created in any language can communicate with other services, The standards as XML and communication over HTTP are used to ensure it conforms to this principle.

1.8.3.4 Components of SOA

- **The Service :** A service is a software module deployed or installed on network provided by the service provider. It exists to be invoked by or to interact with client application known as a service requestor. In short, the client application may request a service for the execution of their application.
- **The Service Description :** The service description consists of the detail information about the interface and implementation of the service. This involves its data types, binding information, operations and network location. It could also consist of categorization and other metadata to provide discovery and utilization by client or requestors. This description is nothing but the set of XML documents. The service description may be published and provided to a request or directly or to a discovery agency.

1.8.3.5 Roles of SOA

- **Service Provider :** From a business point of view, Service Provider is the owner of the service. From an architectural point of view, this is the device or platform that hosts and provides the access to the service. It has also been known as a Service Execution Environment or a Service Container. In the Client-Server model it acts as a server that serves the request means provide the service.
- **Service Requestor :** From a business point of view, Service Requestor is the business or client that requires certain function to be fulfilled. From an architectural point of view, this is the application that is wants to invoke or initialize the interaction with a service. The requestor role can be played by a person accessing website and a program without a user interface, e.g. another web service. In the Client-Server model it acts as a client that requests the service.
- **Discovery Agency or Service Registry :** This is the registry or agency, means a searchable set of service descriptions where service providers publish or stores their service descriptions. The service discovery agency can be in the both form i.e. centralized or distributed. Discovery agencies can works both as provider for client and requestor for the main service provider. But basically service requestors may find services and get binding information from the service registry during development for static binding, or during execution for dynamic binding. For statically bound service requestors, the service discovery agent has an optional role in the architecture, as a service provider can directly send the description to service requestors.

- Service registry is work as **'Service Aggregator'** means it works as both the requestor, who request the service from main provider and provider, who provide the service main client. It also work as broker means it actually provides service taken from provides, so it known as **Service Broker**.

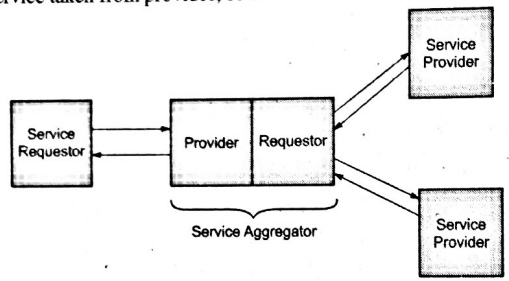

Fig. 1.8.5 : Service Aggregator

1.8.3.6 Operations of SOA

If any application wants to take advantage of Web services, three behaviors should take place: publication of service descriptions, finding or retrieval of service descriptions, and invoking or binding of services based on the information present in service description. These operations are as follows :

- **Publish :** In order to get accessed by requestor, services supposed to publish its description such that the requestor can easily find it. The location with the service is published by provider or publisher is the Service Registry or Discovery Agencies.
- **Find :** In the find operation, the service requestor retrieves or finds a service description directly or queries the service registry if the specific sort service is required. Service requestor can perform find operation in two different lifecycle phases for the service requestor i.e. at design time in order to retrieve the service's interface description for program development called as static binding and at runtime in order to retrieve the service's binding and location description for invocation called as dynamic binding.
- **Bind :**Basically, a service must to be invoked. In the bind operation the service requestor initiates or invokes an interaction with the service at runtime by using the binding information in the service description to locate, contact, and invoke the service. Examples of the interaction include: single message one way, a multimessage conversation, broadcast from requester to many services, or a business process. These types of communication or interactions can be synchronous or asynchronous.

1.8.3.7 Advantages of SOA

- **Service reusability :** In SOA, applications are made from existing or previously created services. Thus, services can be reused to make many other applications.
- **Easy maintenance :** As services are not dependent on client application, it can be updated and modified easily without affecting other services and client application.
- **Platform independent :** In SOA, services are not depends on client application or its platform. So it permits making a complex application by combining several services picked from different sources, independent of the platform.

- **Availability:** SOA features are easily available or accessible to anyone on request.
- **Reliability :** It is easy to debug small services rather than huge codes. So SOA applications are more reliable.
- **Scalability :** As per requirement services can run on several servers within an environment, this enhances the scalability

1.8.3.8 Disadvantages of SOA

- **High overhead :** Input parameters of services can be validated at the time when services interact, this decreases performance as it increases load and response time.
- **High investment :** To implement the SOA, huge initial investment is required.
- **Complex service management :** When services interact they need to interchange messages. The number of messages may go in millions. It becomes a complicated task to handle a large number of messages.

1.8.4 Web Service

GQ. 1.8.7 What is Web Services?

1.8.4.1 What is Web Services ?

- The term Web Services is software module which is available via internet.
- It's a self-describing, self-contained web application component used for client server communication.
- In short web service is nothing but a client computer request a service from server computer and server provides that service to the client.
- It's an outside application which gets coupled with other application whenever they need that service to fulfill their requirement such as to solve some problems, to get some extra resources and information.
- So web service execute transaction on behalf of user or user application.
- So web service is any service that :
  - Is available over network.
  - Is a piece of software for communication between two devices
  - Is a XML based service
  - Is a protocol for exchanging information between devices.

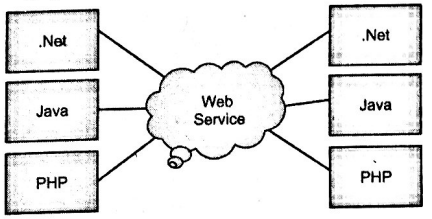

Fig. 1.8.6 : Basic Working of Web Services.

- From Fig. 1.8.6, it is clear that, Java, .Net or PHP applications can communicate with other applications using web service via internet.
- For example, Java application can interact with Java, .Net and PHP applications and also extract information or data from that language.

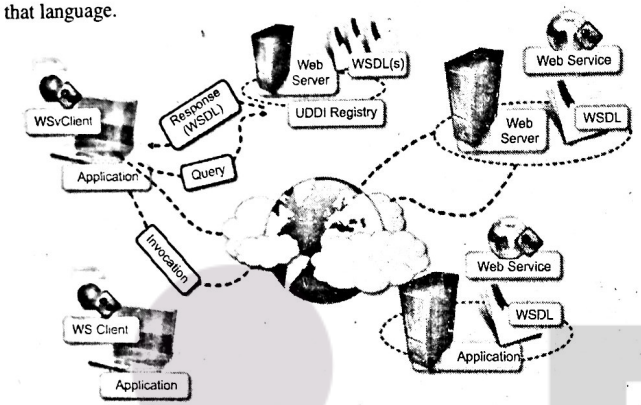

Fig. 1.8.7 : A Web Services Interaction Reference Scenario.

- System architects creates a web service with the technology of their choice and deploy it in a compatible web or application server.
- The service description document, defined through the Web Service Definition Language (WSDL), can either be uploaded to a global or universal registry or attached as metadata to the service itself.
- Using Universal Description Discovery and Integration (UDDI), service consumers can view and search for services in the global catalogues or, likely, retrieve service metadata by querying the web service first. Web Service Description Document grants service consumers to automatically generate clients for a given service and embed them in their existing application.
- Web services are now highly popular, therefore bindings exist for any mainstream programming language as a library or development support tool.
- This simplifies the use of web services in relation to technologies such as CORBA that require more integration efforts.
- In addition, being interoperable, web services definitely considered as a best solution for SOA with respect to multiple distributed object frameworks, like Java RMI, .NET remoting and DCOM / COM1, which are based on the single platform or environment.
- XML is considered as backbone of all these technologies, which is one of the reasons, because of which web service become more popular and easy to use.
- XML-based languages are utilized to control low-level interactions for the web service method calls (SOAP), for giving metadata for search services (WSDL), for discovery services (UDDI) and key operations.
- Basically, the main components that establish web services are SOAP and WSDL.

1.8.4.2 Features of Web Services

- **Loosely Coupled :** A tightly coupled system means client and server application are logical closed or depends on each other. So any changes to any application whether its client or server, other need to update. But this is not a case with Web Services they are loosely coupled means they are not attach directly to one and other. The Web services are provided whenever they are needed.
- **XML Based Services :** XML is powerful language used to facilitate extract and transfer of data from application to any other application over internet.
- **Supports Synchronous and Asynchronous Mode :** Synchronous means client should wait for the repose i.e. to get result from web service and Asynchronous means client can do other task while server provide web service or to get result from web service.
- **Interoperability :** Because of XML, web services can communication or transfer between the devices or application. Basic advantage of XML is that it can represent data and its complex structure specifically. Business integration can be facilitate because xml provide transparent exchange of information and document.
- **Distributed in Nature :** Any application can request web service at any time and web services can be provided to any sort application over internet.
- **Supports Remote Procedure Calls :** Remote Procedure Call (RPC) is an interprocess communication technique or protocol that one application can use to request a service from a other application located in another device on a network without knowing the network structure and its details.
- **Dynamic :** Unlike other application web services can be found and included dynamically whenever they are request or needed by other application. No need to provide prior or tightly binding.

1.9 Cloud Computing Architecture

- The utility-oriented data centers are the first result of cloud computing, and they works as an infrastructure through which services are carried out and delivered.
- Any type of cloud service, whether a virtual hardware, development platform, or application software, build on distributed infrastructure delivered by the provider or hired from a third party.
- It can be created by utilizing a data centre, a collection of clusters, or heterogeneous distribution systems made up of desktop PCs, workstations, and servers.
- Typically, clouds are created by based on one or more datacenters.
- In most cases hardware resources are virtualized to best exploit the isolation of workloads and infrastructure.
- Depending on the particular service provided to the end user, various layers can be stacked on top of the virtual infrastructure: a virtual machine manager, a development platform or a specific application middleware.
- The cloud computing methodology emerged as a result of the convergence of several existing technologies, models and concepts, which changed the way, IT services are delivered and used.

- A detailed definition of the cloud computing can be as follows : Cloud computing is a utility-oriented and Internet-centric approach that provides IT services on demand.
- These services cover the entire computing stack: from hardware infrastructure packaged to virtual machine to software services like development platforms and distributed applications.
- The architecture of cloud computing depicts relationship between different services (IaaS, PaaS, SaaS).

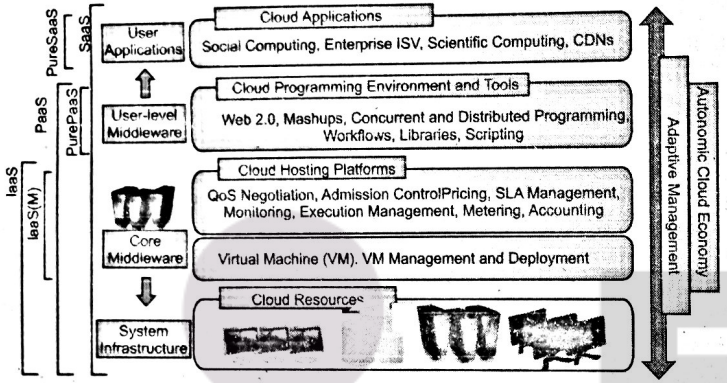

Fig. 1.9.1 : Cloud Computing Architecture

1.10 The Cloud Reference Model

GQ. 1.10.1 Write short note on Cloud Computing Reference Model.

The cloud reference model is also known as the cloud service model on which the cloud computing architecture is depend. Cloud computing technology provides three main types of services that users can access on the cloud platform : Infrastructure as a Service (IAAS), Platform as a Service (PaaS) and Software as a Service (SaaS).

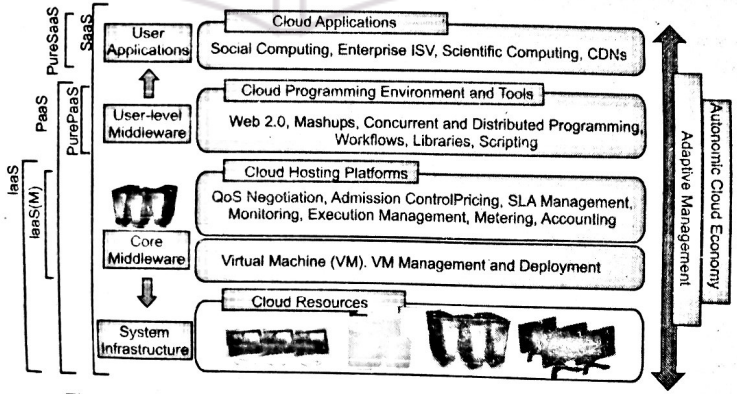

Fig. 1.10.1 : The Cloud Reference Model or Cloud Computing Architecture

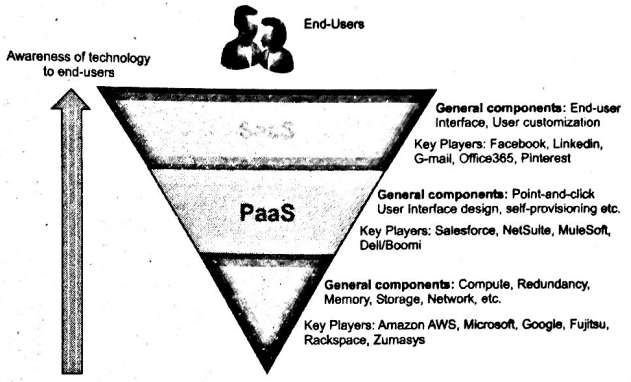

Fig. 1.10.2. Services of Cloud Computing

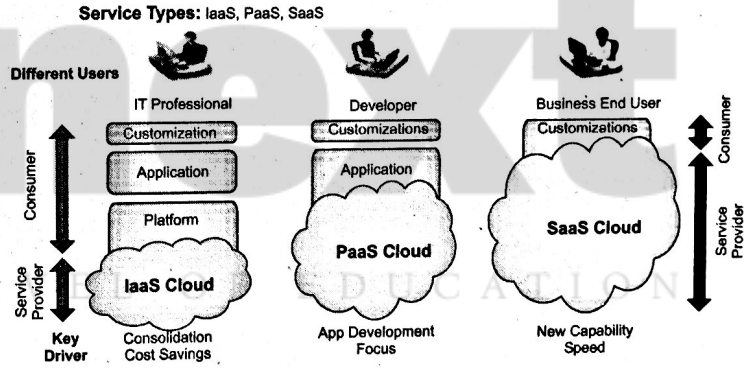

Fig. 1.10.3. Users associated with different Services of Cloud Computing

1.10.1 Infrastructure as a Service (IAAS)

- IaaS, as the term suggests, is a way of providing cloud computing infrastructure like virtual machines, servers, storage drives, operating systems and networks, which is also on same condition i.e. an on-demand service as like SaaS.
- Instead of purchasing servers or developing software, clients can purchase those resources as a fully outsourced service based on their need.
- "Public cloud" is considered as an infrastructure that consists of shared resources, based on a self-service over the Internet.
- In one word, it is the only layer of the cloud where the customer gets the platform for their organization to outsource IT infrastructure on a pay-per-use basis.

- The "public cloud" uses infrastructure consisting of shared resources.
- In another word, it is the only layer of the cloud where the client gets the platform for their organization to outsource the IT infrastructure on a pay-per-use basis.

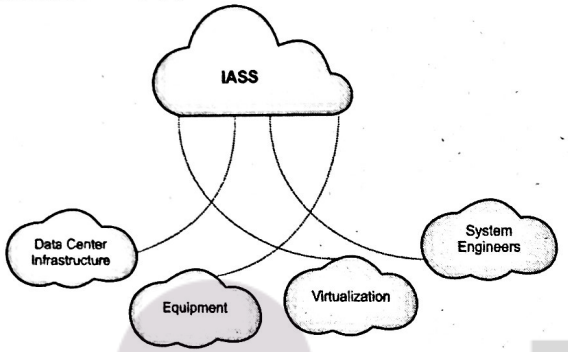

Fig. 1.10.4 : Infrastructure as a Service (IAAS)

1.10.1.1 IaaS Provides Users with

- Load balancers
- Disk storage via virtual machines
- Software Packages
- IP address
- VLANs

1.10.1.2 Advantages of IaaS

- **Dynamic :** The users can dynamically select and configure devices like as CPUs, storage drives, etc.
- **Easy Access :** Users can easily access various cloud computing infrasture.
- **Renting :** Flexible and efficient when hiring IT Infrastructure is quite flexible so user can use or pay as their requirement.
- User will get full control of computer resources as well as they are portability.

1.10.1.3 Disadvantages of IaaS

- Internet connection must be required.
- IaaS relies on virtualization services.
- This service restricts on user-privacy and customization.

1.10.2 Platform as a Service (PaaS)

- PaaS is a platform for programming developers and brings benefits - SaaS is utilized for but from a software development point of view.

- It is nothing but the computer platform that provide the use of web development quickly and easily, without having to buy and maintain web development PaaS similar to SaaS, except SaaS delivers software via the web, While PaaS provides a platform to create Software, delivered via internet.
- PaaS features point-and-click tools that provides facility to non-programmers to develop web applications.
- Examples are Google and Force.com's app-engines, Windows Azure, AppFog, Openshift and VMware Cloud.

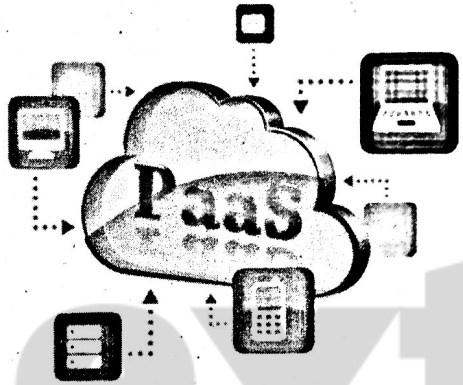

Fig. 1.10.5 : Platform as a Service (IAAS)

1.10.2.1 Advantages of PaaS

- **Scalability :** The number of users ranges from hundreds to thousands.
- **Prebuilt Business Plan :** PaaS vendors gives predefined business functionality to use to start projects directly.
- **Low cost :** The development through PaaS needs a computer and a good Internet connection and therefore less investment in hardware and software.
- **Accelerated Communities :** The PaaS providers provide user with online communities where a developer can get new ideas and share their experiences and advice.
- **Utilization :** Simple and easy to use.

1.10.2.2 Disadvantages of PaaS

- **Vendor migration :** Migration from one PaaS vendors application to another will cause some problems.
- **Data-privacy :** Data confidentiality can be hindered if it is not held within the limits of the company or organization.
- **Mix-up complexity :** Some applications developed may be localized while others are from the cloud, which can increase complexity.

1.10.3 Software as a Service (SaaS)

- SaaS is specified as a software delivery model deployed on the Internet in which the cloud service provider delivers applications.

- It is nothing but the "on-demand software" and "pay-as-you-go application".
- Here the customer can licenses his product through SaaS-providers.
- The SaaS market is a rapidly evolving one, and with this rapidly growing service. SaaS will soon become an active cloud services technology for all organization and companies.
- Therefore it is quite important for those users and buyers to understand the use of SaaS and why it is suitable.
- In SaaS, the associated software and its applications are centrally located on cloud servers, and users can access them through a thin client connecting application using a web browser.

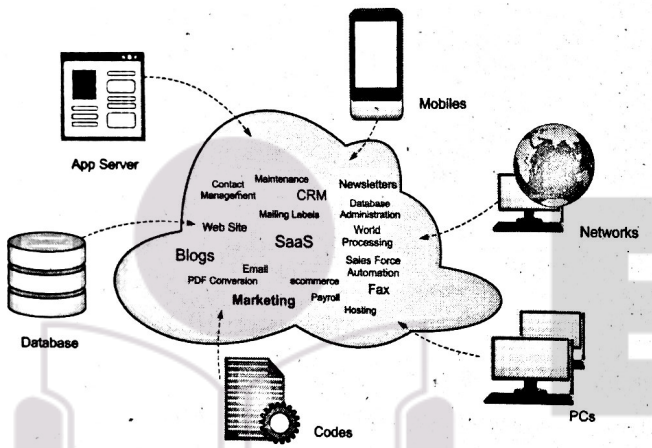

Fig. 1.10.6 : Software as a Service (IAAS)

1.10.3.1 Applications of SaaS

- CRM applications
- Solution to Human Resource (HR)
- Pre-existing Billing & Invoicing systems
- Other daily usable application suites

1.10.3.2 Advantages of SaaS

- **Easy to buy :** SaaS costs are based on a monthly or annual fee, which allows new organizations to enter the world of business at a lower cost, at least lesser as compared to a licensed application.
- **Minimization of hardware requirement :** All SaaS softwares are hosted remotely, therefore there is little or no hardware requirement for organizations.
- **Special software :** No special software version is required because all users will use the same software version. SaaS used to reduces IT costs by outsourcing hardware and software maintenance.
- **Low maintenance :** SaaS removes the problem of installation, maintenance and updation software. SaaS also has lower setup costs than enterprise software.

1.10.3.3 Disadvantages of SaaS

- **Latency factor :** Occurs due to a variable distance of data between the cloud and the end user. Therefore the possibility of latency may arise when interacting with applications.
- **Internet connection :** It is the major issue. Without an internet connection, SaaS applications are not available.
- Switching between SaaS vendors is very difficult in case of any change
- SaaS cloud service is not very secure as compared to in-house deployment.

1.11 Types of Clouds

UQ. 1.11.1 *Explain the concept of public, private and hybrid cloud.* (MU - April 19, 5 Marks)

GQ. 1.11.2 *What is Cloud Deployment Model?*

The boundaries, within which cloud computing services are build provide clues on the based infrastructure adopted to support such services, and make them eligible. It is easily possible to separate four different types of cloud: Public, Private, Hybrid and Community Cloud.

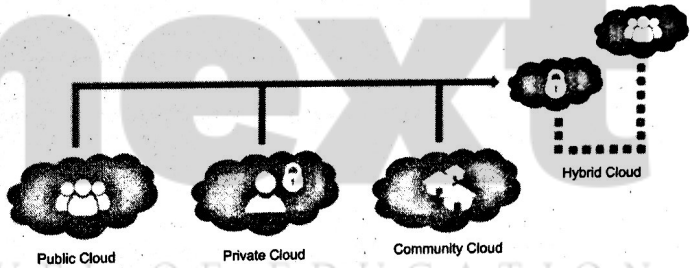

Fig. 1.11.1 : Types of Clouds

1.11.1 Public Cloud

- The Public cloud is a type of cloud that is hosted on the public domain and allows every user to use their services and systems easily.
- Some popular public cloud facilities are Google, Microsoft Azure, IBM etc. Public clouds are typically owned and operated by government, academic or business organizations.
- Public cloud is best suited for load for business purposes. This type of cloud is economical because of the reduction in capital overheads.

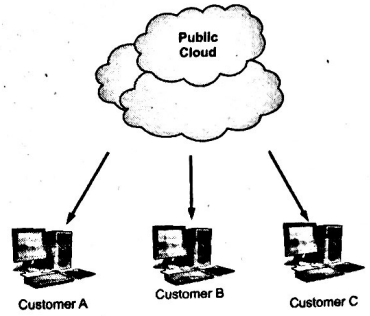

Fig. 1.11.2 : Public Cloud

1.11.1.2 Advantages of Public Cloud

- Cost Effective
- Public clouds are cost effective.
- It is reliable as a public cloud provides a large number of resources from many regions
- It is more flexibility as you can use resources whenever you want.
- It can be used from any place via the Internet.
- High Scalability as you can increase the storage, resource any point of time
- Invoices will be generated according to the usage of resources i.e. pay as per usages.

1.11.1.3 Disadvantages of Public Cloud

- There is problem related to Security and privacy.
- It is less customizable i.e. you can change resources as per your requirement.

1.11.2 Private Cloud

- Private cloud can known as "internal cloud". This clouds compromise under a single organization and under a single organization's datacenter.
- Strong security is needed is provided by firewalls because only authorized users can access resources, this gives better control over data and security.
- The business organizations which requires a dynamic, secure, demand-based need for management should go for a private cloud.
- Private cloud is control and maintained by the organization's IT specialists or third parties.

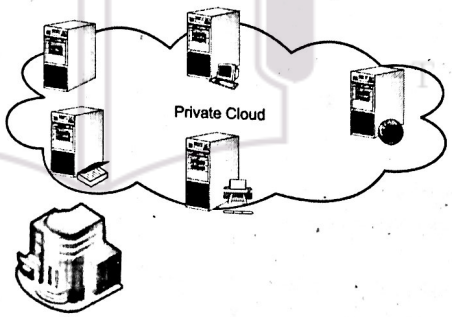

Fig. 1.11.3 : Private Cloud

1.11.2.1 Advantages of Private Cloud

- It is highly secured.
- Recourses are as per need of organization.
- Organization's scrutiny manages private cloud, which provides greater control.

1.11.2.2 Disadvantages of Private Cloud

- Scalability is less, as it can be scalable on limited resources
- The high cost involved as an organization may require setting up datacenters, hardware, etc.
- It is difficult to deploy globally, as it is built under organization boundary.
- For implementation of private cloud requires cloud specialists who can deploy, maintain, etc the cloud resources.

1.11.2.3 Hybrid Cloud

- Hybrid cloud is a mixture of two or more clouds i.e. it is combination of public, private or community cloud.
- All of the organization's sensitive operations or resources reside on the private cloud such that the organization's data handling and non-critical operations such as deployment or testing can be executed on the public or community cloud.
- The organization used to implement network servers that permits interaction between multiple clouds.
- This provides separation to the organization and allows it to cross the boundaries set by public clouds.

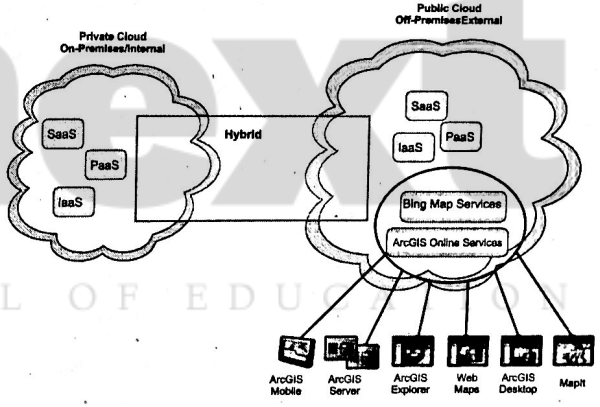

Fig. 1.11.4. Hybrid Cloud

1.11.2.4 Advantages of Hybrid Cloud

- The private cloud in hybrid cloud structure provide the high secure zone for the organization
- It is scalable as it provides both public as well as private scalabilty
- It is more flexibility as you can use resources whenever you want.
- These clouds are cost effective as it uses public cloud also.

1.11.2.5 Disadvantages of Hybrid Cloud

- Complex networking problem because of the presence of both private and public clouds
- It is necessary to ensure that cloud services conform to the security policies of an organization

- The hybrid cloud model relies on internal IT infrastructure, so it is necessary to ensure redundancy in data centers.

1.11.4 Community Cloud

- The community cloud model enables multiple organizations within the same community or region to share the same cloud environment or cloud systems or cloud services.
- The cloud service shares between different organizations and companies that belong to the same community with similar concerns.
- It can be managed either by third parties or internally.
- Organizations with similar computing concerns and shared interest may share it.
- This type of cloud computing is best appropriate for enterprises, business organizations, tenders and research organizations.
- It benefits the users of Community Cloud to be aware, understand and analyze business needs and demands. The community cloud can be present at either onsite or offsite.
- The example of a community cloud is where there are organizations / firms with financial institutions / banks.

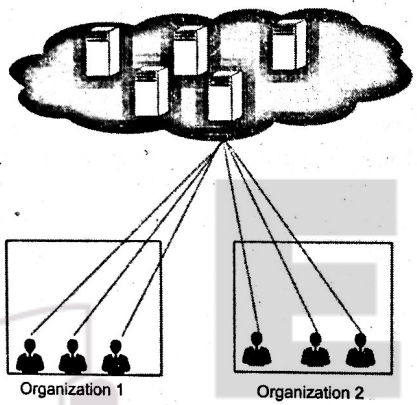

Fig. 1.11.5 : Community Cloud

1.11.4.1 Advantages of Community Cloud

- It is cost effective
- Community cloud is more secure than public cloud but less secure as compare to private cloud.

1.11.4.2 Disadvantages of Community Cloud

- Privacy concerns when multiple organizations share data.
- Allocation of responsibility is also challenging.

Chapter Ends...

❑❑❑

Virtualization

University Prescribed Syllabus

Characteristics of Virtualized Environments. Taxonomy of Virtualization Techniques. Virtualization and Cloud Computing. Pros and Cons of Virtualization. Virtualization using KVM, Creating virtual machines, oVirt - management tool for virtualization environment. Open challenges of Cloud Computing.