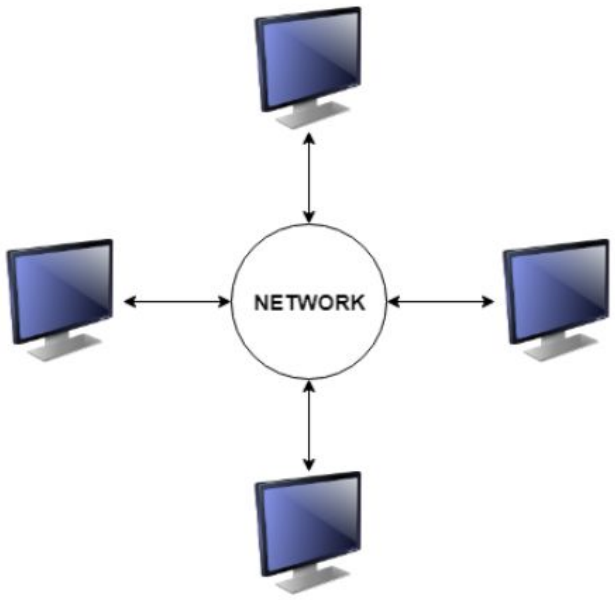
A **distributed system**, also known as **distributed computing**, is a **system** with multiple components located on different machines that communicate and coordinate actions in order to appear as a single coherent **system** to the end-user.



DISTRIBUTED OPERATING SYSTEM

Advantages of Distributed Systems

Some advantages of Distributed Systems are as follows −

- All the nodes in the distributed system are connected to each other. So nodes can easily share data with other nodes.
- More nodes can easily be added to the distributed system i.e. it can be scaled as required.
- The failure of one node does not lead to the failure of the entire distributed system. Other nodes can still communicate with each other.
- Resources like printers can be shared with multiple nodes rather than being restricted to just one.

Disadvantages of Distributed Systems

Some disadvantages of Distributed Systems are as follows −

- It is difficult to provide adequate security in distributed systems because the nodes as well as the connections need to be secured.

Compiled By Diwas Pandey

- Some messages and data can be lost in the network while moving from one node to another.
- The database connected to the distributed systems is quite complicated and difficult to handle as compared to a single user system.
- Overloading may occur in the network if all the nodes of the distributed system try to send data at once.

Examples of distributed systems/applications of distributed computing :

- Intranets, Internet, WWW, email
- Telecommunication networks: Telephone networks and Cellular networks.
- The network of branch office computers -Information system to handle automatic processing of orders,
- Real-time process control: Aircraft control systems,
- Electronic banking,
- Airline reservation systems,
- Sensor networks,
- Mobile and Pervasive Computing systems.

The key features of a distributed system are:

1. Components in the system are concurrent. A distributed system allows resource sharing, including software by systems connected to the network at the same time.
2. There can be multiple components, but they will generally be autonomous in nature.
3. A global clock is not required in a distributed system. The systems can be spread across different geographies.
4. Compared to other network models, there is greater fault tolerance in a distributed model.
5. The price/performance ratio is much better.

The key goals of a distributed system include:

- Transparency: Achieving the image of a single system image without concealing the details of the location, access, migration, concurrency, failure, relocation, persistence, and resources to the users
- Openness: Making the network easier to configure and modify
- Reliability: Compared to a single system, a distributed system should be highly capable of being secure, consistent, and have a high capability of masking errors.
- Performance: Compared to other models, distributed models are expected to give a much-wanted boost to performance.
- Scalability: Distributed systems should be scalable with respect to geography, administration, or size.

Challenges for distributed systems include:

- Security is a big challenge in a distributed environment, especially when using public networks.
- Fault tolerance could be tough when the distributed model is built based on unreliable components.
- Coordination and resource sharing can be difficult if proper protocols or policies are not in place.
- Process knowledge should be put in place for the administrators and users of the distributed model.

Transparency:

The distributed systems should be perceived as a single entity by the users or the application programmers rather than as a collection of autonomous systems, which are cooperating. The users should be unaware of where the services are located and also the transferring from a local machine to a remote one should also be transparent.

Types of Transparencies

The implementation of the distributed system is very complex, as a number of issues have to be considered to achieve its final objective. The complexities should not worry the user of the distributed system from using it i.e., the complexities should be hidden from the user who uses the distributed system. This property of the distributed system is called its transparency. There are different kinds of transparencies that the distributed system has to incorporate. The following are the different transparencies encountered in the distributed systems.

1. Access Transparency: Clients should be unaware of the distribution of the files. The files could be present on a totally different set of servers which are physically distant apart and a A single set of operations should be provided to access these remote as well as the local files. Applications written for the local file should be able to be executed even for the remote files. The examples illustrating this property are the File system in Network File System (NFS), SQL queries, and Navigation of the web.

2. Location Transparency: Clients should see a uniform file namespace. Files or groups of files may be relocated without changing their pathnames. A location transparent name contains no information about the named object's physical location. This property is important to support the movement of the resources and the availability of services. The location and access transparencies together are sometimes referred to as Network transparency. The examples are The File system in NFS and the pages of the web.

3. Concurrency Transparency: Users and Applications should be able to access shared data or objects without interference from each other. This requires very complex mechanisms in a distributed system since there exists true concurrency rather than the simulated concurrency of a central system. The shared objects are accessed simultaneously. The concurrency control and its implementation is a hard task. The examples are NFS, Automatic Teller Machine (ATM) Network.

4. Replication Transparency: This kind of transparency should be mainly incorporated for the distributed file systems, which replicate the data at two or more sites for more reliability. The client generally should not be aware that a replicated copy of the data exists. The clients should also expect operations to return only one set of values. The examples are Distributed DBMS and Mirroring of Web pages.

5. Failure Transparency: Enables the concealment of faults, allowing user and application programs to complete their tasks despite the failure of hardware or software components. Fault tolerance is provided by the mechanisms that relate to access transparency. The distributed system is more prone to failures as any of the components may fail which may lead to degraded service or the total absence of that service. As the intricacies are hidden the distinction between a failed and a slow running process is difficult. Examples are Database Management Systems.

6. Migration Transparency: This transparency allows the user to be unaware of the movement of information or processes within a system without affecting the operations of the users and the applications that are running. This mechanism allows for the load balancing of any particular client, which might be overloaded. The systems that implement this transparency are NFS and Web pages.

7. Performance Transparency: Allows the system to be reconfigured to improve the performance as the load varies.

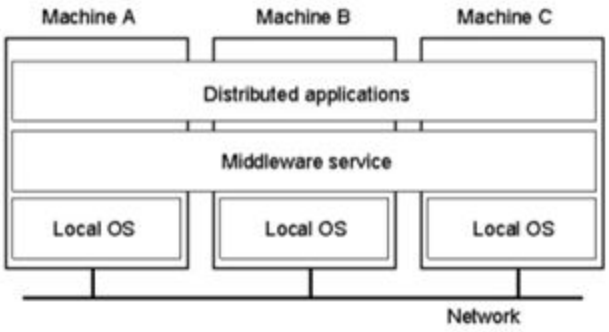
8. Scaling Transparency: A system should be able to grow without affecting application algorithms. Graceful growth and evolution is an important requirement for most enterprises. A system should also be capable of scaling down to small environments where required, and be space and/or time-efficient as required. The best-distributed system example implementing this transparency is the World Wide Web.

MIDDLEWARE IN DISTRIBUTED SYSTEM

Middleware in the context of distributed applications is software that provides services beyond those provided by the operating system to enable the various components of a distributed system to communicate and manage data. Middleware supports and simplifies complex distributed applications. It includes web servers, application servers, messaging, and similar tools that support application development and delivery.

Middleware sits "in the middle" between application software that may be working on different operating systems. Middleware is typically used in distributed systems where it simplifies software development by doing the following:

- Hides the intricacies of distributed applications
- Hides the heterogeneity of hardware, operating systems, and protocols
- Provides uniform and high-level interfaces used to make interoperable, reusable, and portable applications
- Provides a set of common services that minimize duplication of efforts and enhances collaboration between applications



Some of the abstractions provided by middleware include the following:

- Remote method invocation
- Group communication
- Event notification
- Object replication
- Real-time data transmission

Examples of middleware include the following:

- Java RMI
- CORBA
- DCOM

The users of a true distributed system should not know, on which machine their programs are running and where their files are stored.

Compiled By Diwas Pandey

CENTRALIZED COMPUTING:

This is done at a central location using terminals that are attached to a central computer. The computer itself may control all the peripherals directly (if they are physically connected to the central computer) or they may be attached via a terminal server. Alternatively, if the terminals have the capability, they may be able to connect to the central computer over the network. Centralized computing offers greater security because all the processing is controlled in a central location. In addition, if one terminal breaks down the user can simply go to another terminal and log in again and all their files will still be accessible. The central computer performs the computing functions and controls the remote terminals. The disadvantage of this system is that this system of computing relies totally on the central computer, should the central computer crash, the entire system will be unavailable (go down). Accessing the network may also be slow.

ADVANTAGES OF DISTRIBUTED SYSTEMS OVER CENTRALIZED SYSTEMS

- **RELIABILITY**: If one machine crashes, the system as a whole can still survive.
- **SPEED**: A distributed system may have more total computing power than a mainframe
- **OPEN SYSTEM**: Since it is an open system it is always ready to communicate with other systems. An open system that scales has an advantage over a perfectly closed and self-contained system.
- **ECONOMIC**: A collection of microprocessors offers a better price or performance than mainframes.
- **INCREMENTAL GROWTH:** Computing power can be added in small increments

MODELS OF DISTRIBUTED SYSTEM

Distributed System Models is as follows:

1. Architectural Models
2. Interaction Models (**Fundamental Model)**
3. Fault Models **(Fundamental Model)**

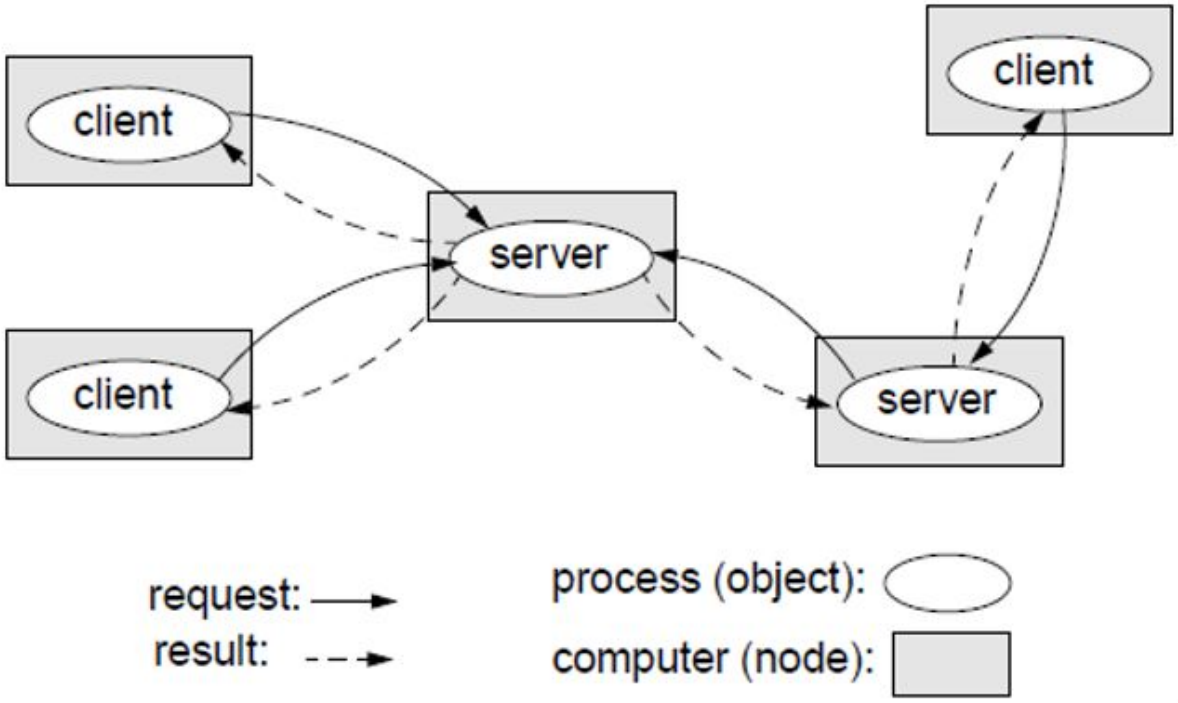
1. Architectural Models

The architectural model describes responsibilities distributed between system components and how are these components placed.

a)Client-server model

☞ The system is structured as a set of processes, called servers, that offer services to the users, called clients.

- The client-server model is usually based on a simple request/reply protocol, implemented with send/receive primitives or using remote procedure calls (RPC) or remote method invocation (RMI):
- The client sends a request (invocation) message to the server asking for some service;
- The server does the work and returns a result (e.g. the data requested) or an error code if the work could not be performed.

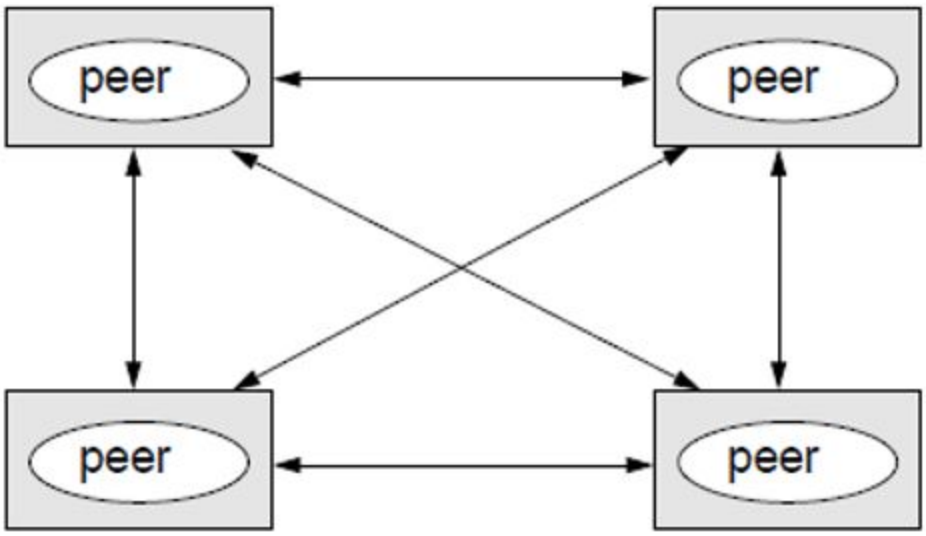


A server can itself request services from other servers; thus, in this new relation, the server itself acts as a client.

b)Peer-to-peer

☞ All processes (objects) play a similar role.

- Processes (objects) interact without a particular distinction between clients and servers.
- The pattern of communication depends on the particular application.
- A large number of data objects are shared; any individual computer holds only a small part of the application database.
- Processing and communication loads for access to objects are distributed across many computers and access links.
- This is the most general and flexible model.



- Peer-to-Peer tries to solve some of the above
- It distributes shared resources widely -> share computing and communication loads.

☞ Problems with peer-to-peer:

- High complexity due to
    - Cleverly place individual objects
    - retrieve the objects
    - maintain a potentially large number of replicas.

2.Interaction Model (Fundamental Model)

Interaction models are for handling time i. e. for process execution, message delivery, clock drifts, etc.

- Synchronous distributed systems

Main features:

- Lower and upper bounds on the execution time of processes can be set.
- Transmitted messages are received within a known bounded time.
- Drift rates between local clocks have a known bound.

Important consequences:

1. In synchronous distributed systems there is a notion of global physical time (with a known relative precision depending on the drift rate).
2. Only synchronous distributed systems have predictable behavior in terms of timing. Only such systems can be used for hard real-time applications.
3. In synchronous distributed system it is possible and safe to use timeouts in order to detect failures of a process or communication link.

☞ It is difficult and costly to implement synchronous distributed systems.

- Asynchronous distributed systems

☞ Many distributed systems (including those on the Internet) are asynchronous. - No bound on process execution time (nothing can be assumed about speed, load, and reliability of computers). - No bound on message transmission delays (nothing can be assumed about speed, load, and reliability of interconnections) - No bounds on drift rates between local clocks.

Important consequences:

1. In an asynchronous distributed system there is no global physical time. Reasoning can be only in terms of logical time (see lecture on time and state).
2. Asynchronous distributed systems are unpredictable in terms of timing.
3. No timeouts can be used.

☞ Asynchronous systems are widely and successfully used in practice.

In practice, timeouts are used with asynchronous systems for failure detection.

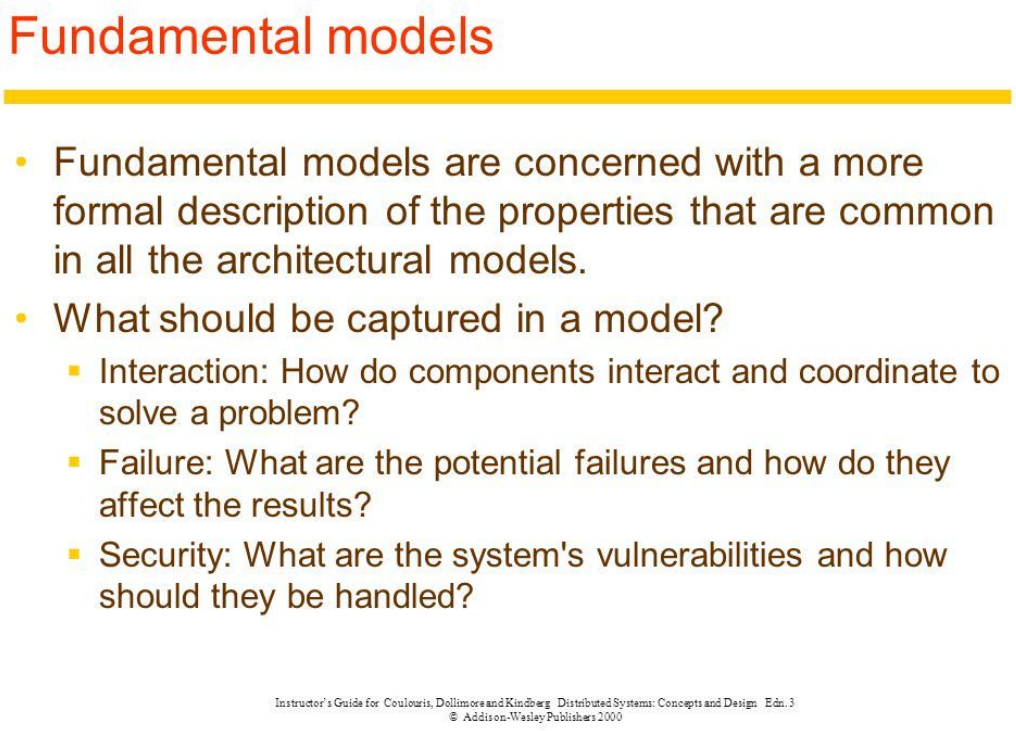
However, additional measures have to be applied in order to avoid duplicated messages, duplicated execution of operations, etc.

3. Fault Models (Fundamental Model)

☞ Failures can occur both in processes and communication channels. The reason can be both software and hardware faults.

☞ Fault models are needed in order to build systems with predictable behavior in case of faults (systems that are fault-tolerant).

☞ such a system will function according to the predictions, only as long as the real faults behave as defined by the “fault model”.



Resource Sharing

Resource sharing means that the existing *resources* in a distributed system can be accessed or remotely accessed across multiple computers in the system. Computers in distributed systems shares resources like *hardware* (disks and printers), *software* (files, windows, and data objects), and *data*. Hardware resources are shared for reductions in cost and convenience. Data is shared for consistency and exchange of information.

> Resources are managed by a software module known as a resource manager. Every resource has its own management policies and methods.

Resource sharing and web challenges in distributed systems include the following points

- Transparency is described as the ambush from the user and the utilization programmer of the division of components in a shared system so that the arrangement is perceived as a whole, preferably than as a combination of independent components.
- The openness of a computer system is the feature that determines whether the operation can be stretched and reimplemented in multiple forms or not.
- Both services and applications produce the support that can be yielded by consumers in a dispersed arrangement. There is, therefore, a probability that infrequent customers will endeavor to reach an accorded support at the selfsame beat.
- Several data resources that are created possible and sustained in shared methods have a huge inherent advantage to their users. Their salvation is, therefore, of significant interest.

What are the different types of distributed computing?

Distributed computing is a multifaceted field with infrastructures that can vary widely. It is thus nearly impossible to define all types of distributed computing. However, this field of computer science is commonly divided into three subfields:

- cloud computing
- grid computing
- cluster computing

Cloud Computing uses distributed computing to provide customers with highly scalable cost-effective infrastructures and platforms. Cloud providers usually offer their resources through hosted services that can be used over the internet. A number of different service models have established themselves on the market:

- **Software as a service (SaaS)**: In the case of SaaS, the customer uses the cloud provider's applications and associated infrastructure (e.g. servers, online storage, computing power). The applications can be accessed with a variety of devices via a thin client interface (e.g. a browser-based web app). Maintenance and administration of the outsourced infrastructure is handled by the cloud provider.
- **Platform as a service(PaaS)**: In the case of PaaS, a cloud-based environment is provided (e.g. for developing web applications). The customer retains control over the applications provided and can configure customized user settings while the technical infrastructure for distributed computing is handled by the cloud provider.
- **Infrastructure as a service (IaaS)**: In the case of IaaS, the cloud provider supplies a technical infrastructure that users can access via public or private networks. The provided infrastructure may include the following components: servers, computing and networking resources, communication devices (e.g. routers, switches, and firewalls), storage space, and systems for archiving and securing data. As for the customer, they retain control over operating systems and provided applications.

Grid computing is based on the idea of a supercomputer with enormous computing power. However, computing tasks are performed in many instances rather than just one. Servers and computers can thus perform different tasks independently of one another. Grid computing can access resources **in a very flexible manner when performing tasks**. Normally, participants will allocate specific resources to an entire project at night when the technical infrastructure tends to be less heavily used.

One advantage of this is that **highly powerful systems can be quickly used** and the **computing power can be scaled** as needed. There is no need to replace or upgrade an expensive supercomputer with another pricey one to improve performance.

Since grid computing can create a **virtual supercomputer** from a cluster of loosely interconnected computers, it is specialized in solving problems that are particularly computationally intensive. This method is often used for ambitious scientific projects and **decrypting cryptographic codes**.

Cluster computing cannot be clearly differentiated from cloud and grid computing. It is a more general approach and refers to all the ways in which individual computers and their computing power can be combined together in clusters. Examples of this include server clusters, clusters in big data and in cloud environments, database clusters, and application clusters. Computer networks are also increasingly being used in **high-performance computing** which can solve particularly demanding computing problems.