Unit: 1 Introduction to Distributed System

Design Issues in Distributed Operating System

The issues in designing a distributed system include:

Transparency

- Main goal of Distributed system is to make the existence of multiple computers invisible (transparent) and provide single system image to user.
- A transparency is some aspect of the distributed system that is hidden from the user (programmer, system developer, application).
- A transparency is provided by including some set of mechanisms in the distributed system at a layer below the interface such that a communication system appears to its users as a virtual uniprocessor.
- While users hit search in google.com (or any other search mechanism), they never notice that their query goes through a complex process before google (or any other search mechanism) shows them a result.

Access Transparency:

It hides differences in data representation and how a resource is accessed by a user.

Example: a distributed system may have a computer system that runs different operating systems, each having their own file naming conventions. Differences in naming conventions as well as how files can be manipulated should be hidden from the users and applications.

Location Transparency:

Hides where exactly the resource is located physically.

Example: by assigning logical names to resources like yahoo.com, one cannot get an idea of the location of the web page's main server.

Migration Transparency:

Distributed system in which resources can be moved without affecting how the resource can be accessed are said to provide migration transparency. It hides that the resource may move from one location to another.

Relocation Transparency:

This transparency deals with the fact that resources can be relocated while it is being accessed without the user who is using the application to know anything.

Example: using a Wi-Fi system on laptops while moving from place to place without getting disconnected.

Replication Transparency:

Hides the fact that multiple copies of a resource could exist simultaneously. To hide replication, it is essential that the replicas have the same name. Consequently, as system that supports replication should also support location transparency.

Concurrency Transparency:

It hides the fact that the resource may be shared by several competitive users.
Example: two independent users may each have stored their file on the same server and may be accessing the same table in a shared database. In such cases, it is important that each user should not notice that the others are making use of the same resource.

Failure Transparency:

Hides failure and recovery of the resources.

Example: a user cannot distinguish between a very slow or dead resource. Some error messages come when a server is down or when the network is overloaded or when the connection from the client side is lost. So here, the user is unable to understand what has to be done, either the user should wait for the network to clear up, or try again later when the server is working again.

Persistence Transparency:

It hides if the resource is in memory or disk.
Example: Object oriented database provides facilities for directly invoking methods on storage objects. First the database server copies the object states from the disk i.e. main memory performs the operation and writes the state back to the disk. The user does not know that the server is moving between primary and secondary memory.

Reliability

- Reliability in terms of data means that data should be available without any errors.
- Distributed system where multiple processors are available and the system become reliable.
- So on a failure, a backup file is available.
- In case of replication all copies should be consistent.

Scalability

- Distributed systems must be scalable as the number of user increases.
- A system is said to be scalable if it can handle the addition of users and resources without suffering a noticeable loss of performance or increase in administrative complexity.

- Scalability has 3 dimensions:
    - **Size:** Number of users and resources to be processed. Problem associated is overloading.
    - **Geography:** Distance between users and resources. Problem as sociated is communication reliability
    - **Administration:** As the size of distributed systems increases, many of the system needs to be controlled. Problem associated is administrative mess.

Flexibility

- The design of Distributed operating system should be flexible due to following reasons:
    - **Ease of Modification:** It should be easy to incorporate changes in the system in a user transparent manner or with minimum interruption caused to the users.
    - **Ease of Enhancement:** New functionality should be added from time to time to make it more powerful and easy to use.
- A group of users should be able to add or change the services as per the comfortability of their use.

Performance

- A performance should be better than or at least equal to that of running the same application on a single-processor system.
- Some design principles considered useful for better performance are as below:
    - **Batch if possible:** Batching often helps in improving performance.
    - **Cache whenever possible:** Caching of data at client's side frequently improves overall system performance.
    - **Minimize copying of data:** Data copying overhead involves a substantial CPU cost of many operations.
    - **Minimize network traffic:** It can be improved by reducing internode communication costs.

Heterogeneity

- This term means the diversity of the distributed systems in terms of hardware, software, platform, etc.
- Modern distributed systems will likely have different:
    - **Hardware devices:** computers, tablets, mobile phones, embedded devices, etc.
    - **Operating System:** MS Windows, Linux, Mac, Unix, etc.
    - **Network:** Local network, the Internet, wireless network, satellite links, etc.
    - **Programming languages:** Java, C/C++, Python, PHP, etc.
    - Different roles of software developers, designers, system managers.

Security

- System must be protected against destruction and unauthorized access.
- Enforcement of Security in a distributed system has the following additional requirements as compared to centralized system:
    - Sender of message should know that message is delivered.
    - Receiver of the message should know that the message was sent by genuine sender.

  Both sender and receiver should be guaranteed that the content of message were not changed while it is in transfer.