DISTRIBUTED OPERATING SYSTEMS
Design Issues

Design Issues

- Transparency
- Reliability
- Flexibility
- Performance
- Scalability
- Heterogeneity
- Security

Transparency

- How to achieve the single-system image, i.e., how to make a collection of computers appear as a single computer.
- Hiding all the distribution from the users as well as the application programs can be achieved at two levels:

  1) hide the distribution from users

  2) at a lower level, make the system look transparent to programs.

  1) and 2) requires uniform interfaces such as access to files, communication.

Types of Transparency

- Access : User should not be able to recognize whether resource is remote or local
- Location :
  - Name : Name of resource should not reveal any hint as to the physical location of the resource
  - User Mobility : Regardless of machine where user has logged in, he should be able to access a resource with the same name

Types of Transparency

- Replication : The existence of multiple copies of a replicated resource (naming of replicas) and the replication activity (replication control) should be transparent to the users
- Failure : It deals with masking from the users' partial failures in the system, such as
  - A communication link failure
  - A machine failure
  - A storage device crash

  Complete failure transparency is not achievable

Types of Transparency

- Migration : For better performance and reliability reasons, an object that is capable of being moved is often migrated from one node to other .
- Concurrency : Each user has a feeling that he or she is the sole user of the system and other user do not exist in the system
  - Need to allow multiple users to concurrently access the same resource.
  - Lock and unlock for mutual exclusion.

Types of Transparency

- Performance : The aim is to allow the system to be automatically reconfigured to improve performance, as loads vary dynamically in the system.
- Scaling : The aim is to allow the system to expand in scale without disrupting the activities of the users.

Reliability

- Distributed system should be more reliable than single system.
- Example: 3 machines with .95 probability of being up. 1-.05**3 probability of being up.
  - Availability: fraction of time the system is usable. Redundancy improves it.
  - Need to maintain consistency
  - Need to be secure
  - Fault tolerance: need to mask failures, recover from errors.

Flexibility

- Make it easier to change
- Monolithic Kernel: systems calls are trapped and executed by the kernel. All system calls are served by the kernel, e.g., UNIX.
- Microkernel: provides minimal services.
  1) IPC
  2) some memory management
  3) some low-level process management and scheduling
  4) low-level i/o
  E.g., Mach can support multiple file systems, multiple system interfaces.

Performance

- Without gain on this, why bother with distributed systems.
- Performance loss due to communication delays:
- Performance loss due to making the system fault tolerant.
- Some design principles for better performance
  - Batch if possible
  - Cache whenever possible
  - Minimizing copying of data

Scalability

- It refers to the capability of a system to adapt to increased service load.
- Guiding principles for designing scalable distributed systems
  - Avoid centralized entities
  - Avoid centralized algorithms
  - Perform most operations on client workstations

Heterogeneity

- Heterogeneous DS are preferred by many users because they provide more flexibility
- But it is difficult to design DS because of diversity and hence incompatibility
- Different types of incompatibilities
  - Hardware
  - Software (OS)
  - Communication protocols and topologies
  - Data format

Security

- It is difficult to achieve security in DS compared to centralized system because of
  - The lack of single point of control
  - Use of insecure networks for data communication
- Cryptography is the known practical method for dealing with security aspects of a distributed system.