Table of Contents

Issues in designing distributed system
1. Heterogeneity
2. Openness
3. Security
4. Scalability
5. Failure handling
6. Concurrency
7. Transparency
8. Quality of service
9. Reliability
10. Performance

Issues in designing distributed system

1. Heterogeneity

The Internet enables users to access services and run applications over a heterogeneous collection of computers and networks .Internet consists of many different sorts of network their differences are masked by the fact that all of the computers attached to them use the Internet protocols to communicate with one another.

For e.g., a computer attached to an Ethernet has an implementation of the Internet protocols over the Ethernet, whereas a computer on a different sort of network will need an implementation of the Internet protocols for that network.

2. Openness

The openness of a computer system is the characteristic that determines whether the system can be extended and re-implemented in various ways .The openness of distributed systems is determined primarily by the degree to which new resource-sharing services can be added and be made available for use by a variety of client programs.

3. Security

Many of the information resources that are made available and maintained in distributed systems have a high intrinsic value to their users .Their security is therefore of considerable importance. Security for information resources has three components: confidentiality, integrity, and availability.

4. Scalability

Distributed systems operate effectively and efficiently at many different scales, ranging from a small intranet to the Internet. A system is described as scalable if it will remain effective when there is a significant increase in the number of resources and the number of users.

5. Failure handling

Computer systems sometimes fail. When faults occur in hardware or software, programs may produce incorrect results or may stop before they have completed the intended computation. Failures in a distributed system are partial – that is, some components fail while others continue to function. Therefore the handling of failures is particularly difficult.

6. Concurrency

Both services and applications provide resources that can be shared by clients in a distributed system. There is therefore a possibility that several clients will attempt to access a shared resource at the same time. Object that represents a shared resource in a distributed system must be responsible for ensuring that it operates correctly in a concurrent environment. This applies not only to servers but also to objects in applications. Therefore any programmer who takes an implementation of an object that was not intended for use in a distributed system must do whatever is necessary to make it safe in a concurrent

environment.

7. Transparency

Transparency can be achieved at two different levels. Easiest to do is to hide the distribution from the users. The concept of transparency can be applied to several aspects of a distributed system.
a) Location transparency: The users cannot tell where resources are located
b) Migration transparency: Resources can move at will without changing their names
c) Replication transparency: The users cannot tell how many copies exist.
d) Concurrency transparency: Multiple users can share resources automatically.
e) Parallelism transparency: Activities can happen in parallel without users knowing.

8. Quality of service

Once users are provided with the functionality that they require of a service, such as the file service in a distributed system, we can go on to ask about the quality of the service provided. The main nonfunctional properties of systems that affect the quality of the service experienced by clients and users are reliability, security and performance. Adaptability to meet changing system configurations and resource availability has been recognized as a further important aspect of service quality.

9. Reliability

One of the original goals of building distributed systems was to make them more reliable than single-processor systems. The idea is that if a machine goes down, some other machine takes over the job. A highly reliable system must be highly available, but that is not enough. Data entrusted to the system must not be lost or garbled in any way, and if files are stored

redundantly on multiple servers, all the copies must be kept consistent. In general , the more copies that are kept, the better the availability, but the greater the chance that they will be inconsistent, especially if updates are frequent.

10. Performance

Always the hidden data in the background is the issue of performance. Building a transparent, flexible, reliable distributed system, more important lies in its performance . In particular, when running a particular application on a distributed system, it should not be appreciably worse than running the same application on a single processor. Unfortunately, achieving this is easier said than done.