CHARACTERIZATION OF DISTRIBUTED SYSTEMS

محاضرة الثلاثاء ٢٢/٣/٢٠٢٢

From Chapter 1 of Distributed Systems Concepts and Design,

By G. Coulouris, J. Dollimore, T. Kindberg and G. Blair

Published by Addison Wesley/Pearson Education

Topics

- Defining Distributed Systems
- Problems to be addressed
- Examples of Distributed Systems
- Resource sharing
- Design Challenges of Distributed Systems

Defining Distributed Systems

- Various definition of distributed systems.
- With any definition, sharing of resources is a main motivation for constructing distributed systems.
- In this course, we define distributed systems:
  - A distributed system is one in which hardware or software components located at networked computers communicate and coordinate their actions only by message passing.
- Definition above covers the entire range of distributed systems in which networked computers can usefully be deployed.

Defining Distributed Systems

- Examples of networks of computers are:
  - Mobile networks
  - Corporate networks
  - Factory networks
  - Campus networks
  - Home networks
  - In-car networks
  - On board networks in aero planes and trains

- *Computer Network*: the independent computers are explicitly visible (can be explicitly addressed).
- *Distributed System*: existence of multiple independent computers is transparent.

Problems to be addressed

- Our definition of distributed systems has the following significant consequences:
  - Concurrency

❖Different Computers on a network. Each computer can be doing work at the same time.
❖What happens if two computers want to access a resource at the same time?
❖Network delays are not constant make synchronization difficult.

Problems to be addressed

- Our definition of distributed systems has the following significant consequences:

  - No global clock

❖When programs need to cooperate they coordinate their actions by exchanging messages.
❖Close coordination often depends on a shared idea of the time at which events occur.
❖There are limits to the accuracy with which components in a network can synchronize their clocks. There is no global notion of the correct time.

Problems to be addressed

➢ Independent Failures

❖ All computer systems can fail.

❖Distributed systems fail in new ways.

❖Network faults might result in components being isolated but not Stopping.

❖What happens if your waiting for an acknowledge message and you never get it?

❖How do you know if the remote system got your message?

❖Each component of the system can fail independently, leaving the others still running.

Examples of Distributed Systems

Internet: a very large distributed system consisting of many interconnected computer networks.

The implementation of the internet and the services that it supports, has entailed the development of practical solutions to many distributed system issues.

Intranets: a “mini-internet” which uses the same technologies as the internet, but is controlled by an organization.

Cluster: A type of parallel or distributed processing system, which consists of a collection of interconnected stand-alone computers cooperatively working together as a single, integrated computing resource.

Other Examples of Distributed Systems

- Grid: A type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed autonomous resources dynamically at runtime depending on their availability, capability, performance, cost, and users' quality-of-service requirements.
- Cloud computing: A cloud is defined as a set of Internet-based application, storage and *computing services* sufficient to support most users' needs, thus enabling them to largely or totally dispense with local data storage and application software.
  - The term also promotes a view of everything as a *service*, from physical or virtual infrastructure through to software, often paid for on a per-usage basis rather than purchased.
  - Cloud computing reduces requirements on users' devices, allowing very simple devices to access a wide range of resources and services.

Resource sharing

- What is a resource?

❖ Hardware (shared printer).
❖Pieces of running software, distributed objects.
❖Data (files).

- The terms “resources” and “components” are used to describe how a distributed system is built up.

- Resources in a distributed system are:
  - Encapsulated within computers.
  - Can only accessed from other computers by communication.
  - Managed by a program that offers a communication interface.

Design Challenges of Distributed Systems

- Designers of distributed systems need to take the following challenges into account:
  - ➢ Heterogeneity
    - ❖ Heterogeneous components (Networks, Hardware architectures, Operating systems, Programming languages, Different developers) must be able to interoperate.
    - ❖ Examples that mask differences in network, operating systems, hardware and software to provide heterogeneity are:
      - ✓ Internet protocols
      - ✓ Middleware(CORBA , Java RMI)
      - ✓ Mobile code (Java applets)

Design Challenges of Distributed Systems

- Openness: Interfaces should allow components to be added or replaced.
  - 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
  - The first step in openness is publishing the documentation of software components and interfaces of the components to make them available to software developers.

Design Challenges of Distributed Systems

- Security
  - The system should only be used in the way intended.
  - Security of a computer system is the characteristic that the resources are accessible to authorized users and used in the way they are intended.
  - Security for information resources has three components:
    - Confidentiality: Protection against disclosure to unauthorized individual.
    - Integrity: Protection against alteration or corruption.
    - Availability: Protection against interference with the means to access the resources.

Design Challenges of Distributed Systems

- Scalability
  - System should work efficiently with an increasing number of users.
  - System performance should increase with inclusion of additional resources.
  - Scalable distributed systems operate effectively and efficiently at many different scales, ranging from a small Intranet to the Internet.
  - **Caching and replication in Web are examples of providing scalability.**

Design Challenges of Distributed Systems

- Failure handling: Failure of a component (partial failure) should not result in failure of the whole system.
  - Failures in distributed systems are partial, that is some components fail while others continue to function.
  - Techniques for dealing with failures:
    - Detecting failures:Checksums
    - Masking failures: Retransmission of corrupt messages
    - Tolerating failures: Exception handling
    - Recovery from Failure: Rollback mechanisms
    - Redundancy: Redundant components

Design Challenges of Distributed Systems

- Concurrency: Any object that represents a shared resource in a distributed system must be responsible for ensuring that it operates correctly in a concurrent environment.
  - With concurrency, services and applications can be shared by clients in a distributed system.
  - For an object to be safe in a concurrent environment, its operations must be synchronized in such a way that its data remains consistent.
  - Concurrency can be achieved by standard techniques such as semaphores, which are used in most operating systems.

Design Challenges of Distributed Systems

- Transparency
  - Distribution should be hidden from the user as much as possible.
  - Transparency is defined as the hiding of the separation of components in a distributed systems from the user and the application programmer.
  - With transparency the system is perceived as a whole rather than a collection of independent components.

Transparency

- Forms of transparencies:
  - Access transparency
    - Enables local and remote resources to be accessed using identical operations.
  - Location transparency
    - Enables resources to be accessed without knowledge of their physical or network location (for example, which building or IP address).
  - Concurrency transparency
    - Enables several processes to operate concurrently using shared resources without interference between them.

Transparency

- Replication transparency
  - Enables multiple instances of resources to be used to increase reliability and performance without knowledge of the replicas by users or application programmers.
- Failure transparency
  - Enables the concealment of faults, allowing users and application programs to complete their tasks despite the failure of hardware or software components.
- Mobility transparency
  - Allows the movement of resources and clients within a system without affecting the operation of users or programs.

Transparency

- Performance transparency
  - Allows the system to be reconfigured to improve performance as loads vary.
- Scaling transparency
  - Allows the system and applications to expand in scale without change to the system structure or the application algorithms.
- The two most important transparencies are access and location transparency referred to together as network transparency.