1DT057
DISTRIBUTED INFORMATION SYSTEM

Distributed Systems
Characterisation and Design

OUTLINE

1. What is a Distributed System
2. Examples of Distributed Systems
3. Common Characteristics
4. Basic Design Issues
5. Summary
1. **Distributed System Types**

![Diagram showing types of distributed systems]

- Fully Distributed
- Local data, local directory
- Not fully replicated master directory
- Fully replicated
- Homog. general purpose
- Heterog. general purpose
- Homog. special purpose
- Heterog. special purpose
- Master-slave
- Autonomous transaction based
- Autonomous fully cooperative

1. **What is a Distributed System?**

   Definition: A *distributed system* is one in which **components** located at networked computers communicate and **coordinate** their actions only by passing **messages**. This definition leads to the following characteristics of distributed systems:

   - Concurrency of components
   - Lack of a global ‘clock’
   - Independent failures of components ‘acceptable’
1.1 **CENTRALIZED SYSTEM CHARACTERISTICS**

- One component with non-autonomous parts
- Component shared by users all the time
- All resources accessible
- Software runs in a single process
- Single point of **control**
- Single point of **failure**

1.2 **DISTRIBUTED SYSTEM CHARACTERISTICS**

- Multiple autonomous components
- Components are not shared by all users
- Resources may not be accessible
- Software runs in concurrent processes on different processors
- Multiple points of control
- Multiple points of failure
2. Examples of Distributed Systems

- Google Datacenters
- Local Area Network and Intranet
- Database Management System
- Automatic Teller Machine Network
- Internet/World-Wide Web
- Mobile and Ubiquitous Computing

2.0 Google Datacenters
2.1 **LOCAL AREA NETWORK**

- Local area network
- Print and other servers
- Web server
- Email server
- File server
- The rest of the Internet
- Router/firewall

- Desktop computer
- Print
- Other servers

2.2 **DATABASE MANAGEMENT SYSTEM**

- Display 1
- Display 2
- Appl 1
- Appl 2
- Database Engine
- Database Monitor
- Database Engine
- Database Engine
- DB 1
- DB 2
- DB 3

- Machine
- Process
- Network/Interprocess Communication
- OS Communication to external devices
2.3 Automatic Teller Machine Network

2.4 Internet
2.4.1 WORLD-WIDE-WEB

2.4.2 WEB SERVERS AND WEB BROWSERS
3. COMMON CHARACTERISTICS

- What are we trying to achieve when we construct a distributed system?
- Certain common characteristics can be used to assess distributed systems
  - Heterogeneity
  - Openness
  - Security
  - Scalability
  - Failure Handling
  - Concurrency
  - Transparency
3.1 Heterogeneity

- Variety and differences in
  - Networks
  - Computer hardware
  - Operating systems
  - Programming languages
  - Implementations by different developers

- Middleware as software layers to provide a programming abstraction as well as masking the heterogeneity of the underlying networks, hardware, OS, and programming languages (e.g., CORBA).

- Mobile Code to refer to code that can be sent from one computer to another and run at the destination (e.g., Java applets and Java virtual machine).

3.2 Openness

- Openness is concerned with extensions and improvements of distributed systems.

- Detailed interfaces of components need to be published.

- New components have to be integrated with existing components.

- Differences in data representation of interface types on different processors (of different vendors) have to be resolved.
3.3 Security

- In a distributed system, clients send requests to access data managed by servers, resources in the networks:
  - Doctors requesting records from hospitals
  - Users purchase products through electronic commerce
- Security is required for:
  - Concealing the contents of messages: security and privacy
  - Identifying a remote user or other agent correctly (authentication)
- New challenges:
  - Denial of service attack
  - Security of mobile code

3.4 Scalability

- Adaptation of distributed systems to
  - accommodate more users
  - respond faster (this is the hard one)
- Usually done by adding more and/or faster processors.
- Components should not need to be changed when scale of a system increases.
- Design components to be scalable!
3.5 **Failure Handling (Fault Tolerance)**

- Hardware, software and networks fail!
- Distributed systems must maintain *availability* even at low levels of hardware/software/network *reliability*.
- Fault tolerance is achieved by
  - recovery
  - redundancy

3.6 **Concurrency**

- Components in distributed systems are executed in concurrent processes.
- Components access and update shared resources (e.g. variables, databases, device drivers).
- Integrity of the system may be violated if concurrent updates are not coordinated.
  - Lost updates
  - Inconsistent analysis
3.7 Transparency

- Distributed systems should be perceived by users and application programmers as a whole rather than as a collection of cooperating components.
- Transparency has different aspects.
- These represent various properties that distributed systems should have.

3.7.1 Access Transparency

- Enables local and remote information objects to be accessed using identical operations.
- Example: File system operations in NFS.
- Example: Navigation in the Web.
- Example: SQL Queries
3.7.2 Location Transparency

- Enables information objects to be accessed without knowledge of their location.
- Example: File system operations in NFS
- Example: Pages in the Web
- Example: Tables in distributed databases

3.7.3 Concurrency Transparency

- Enables several processes to operate concurrently using shared information objects without interference between them.
- Example: NFS
- Example: Automatic teller machine network
- Example: Database management system
### 3.7.4 Replication Transparency

- Enables multiple instances of information objects to be used to increase reliability and performance without knowledge of the replicas by users or application programs.
- Example: Distributed DBMS
- Example: Mirroring Web Pages.

### 3.7.5 Failure Transparency

- Enables the concealment of faults
- Allows users and applications to complete their tasks despite the failure of other components.
- Example: Database Management System
### 3.7.6 Mobility Transparency

- Allows the movement of information objects within a system without affecting the operations of users or application programs

- Example: NFS

- Example: Web Pages

---

### 3.7.7 Performance Transparency

- Allows the system to be reconfigured to improve performance as loads vary.

- Example: Distributed make.
3.7.8 Scaling Transparency

- Allows the system and applications to expand in scale without change to the system structure or the application algorithms.

- Example: World-Wide-Web

- Example: Distributed Database

4. Basic Design Issues

- General software engineering principles include rigor and formality, separation of concerns, modularity, abstraction, anticipation of change, ...

- Specific issues for distributed systems:
  - Naming
  - Communication
  - Software structure
  - System architecture
  - Workload allocation
  - Consistency maintenance
4.1 Naming

- A name is resolved when translated into an interpretable form for resource/object reference.
  - Communication identifier (IP address + port number)
  - Name resolution involves several translation steps
- Design considerations
  - Choice of name space for each resource type
  - Name service to resolve resource names to comm. id.
- Name services include naming context resolution, hierarchical structure, resource protection

4.2 Communication

- Separated components communicate with sending processes and receiving processes for *data transfer* and *synchronization*.
- Message passing: *send* and *receive* primitives
  - synchronous or blocking
  - asynchronous or non-blocking
  - Abstractions defined: channels, sockets, ports.
- Communication patterns: client-server communication (e.g., RPC, function shipping) and group multicast
4.3 **SOFTWARE STRUCTURE**

- Layers in centralized computer systems:

  - Applications
  - Middleware
  - Operating system
  - Computer and Network Hardware

- Layers and dependencies in distributed systems:

  - Applications
    - Distributed programming support
    - Open system kernel services
  - Open services
  - Computer and network hardware
4.4 SYSTEM ARCHITECTURES

- Client-Server
- Peer-to-Peer
- Services provided by multiple servers
- Proxy servers and caches
- Mobile code and mobile agents
- Network computers
- Thin clients and mobile devices

4.4.1 CLIENTS INVOKE INDIVIDUAL SERVERS
4.4.2 Peer-to-peer Systems

Peer1
Peer2
Peer3
Peer 5... N

Application

Peer4

Application

4.4.3 A Service by Multiple Servers

Client
Server
Client
Server
4.4.4 Web Proxy Server

- client request results in the downloading of applet code
- client interacts with the applet
4.4.6 Thin Clients and Compute Servers

5. Summary

- Definitions of distributed systems and comparisons to centralized systems.
- The characteristics of distributed systems.
- The eight forms of transparency.
- The basic design issues.
- Read Chapter 1 and Chapter 2 of the textbook.