1. What is a Distributed System??

A distributed system is a collection of independent computers that appears to its users as a single coherent system.

Distributed computing is a field of computer science that studies distributed systems. A distributed system consists of multiple autonomous computers that communicate through a computer network. The computers interact with each other in order to achieve a common goal. A computer program that runs in a distributed system is called a distributed program, and distributed programming is the process of writing such programs.

Distributed computing also refers to the use of distributed systems to solve computational problems. In distributed computing, a problem is divided into many tasks, each of which is solved by one or more computers.

A distributed system is a collection of independent computers, interconnected via a network, capable of collaborating on a task.

Distributed computing is computing performed in a distributed system. Distributed computing has become increasingly common due advances that have made both machines and networks cheaper and faster.

Some examples of distributed systems:

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

2. Centralized vs. Distributed Computing
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

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

3. Advantages of Distributed Systems over Centralized System
- **Economics**: a collection of microprocessors offer a better price/performance than mainframes. Low price/performance ratio: cost effective way to increase computing power.
- **Speed**: a distributed system may have more total computing power than a mainframe.
- **Inherent distribution**: Some applications are inherently distributed. Ex. a supermarket chain.
- **Reliability**: If one machine crashes, the system as a whole can still survive. Higher availability and improved reliability.
- **Incremental growth**: Computing power can be added in small increments. Modular expandability
- **Another deriving force**: the existence of large number of personal computers, the need for people to collaborate and share information.

4. Advantages of Distributed Systems over Independent PCs
- Data sharing: allow many users to access to a common data base
- Resource Sharing: expensive peripherals like color printers
- Communication: enhance human-to-human communication, e.g., email, chat
- Flexibility: spread the workload over the available machines

5. Design Issues of Distributed Systems
6. Basics of Distributed Systems:

- Networked computers (close or loosely coupled) that provide a degree of operation transparency.
- Distributed Computer System = independent processors + networking infrastructure
- Communication between processes (on the same or different computer) using message passing technologies is the basis of distributed computing.

7. Components of Distributed Software Systems

- Distributed systems
- Middleware
- Distributed applications

8. Goals of Distributed Systems:

8.1 Resource sharing: easy for users to access remote resources.
8.2 Transparency: to hide the fact that processes and resources are physically distributed across multiple computers.
8.3 Openness: to offer services according to standard rules.
8.4 Scalability: easy to expand and manage.
8.5 Heterogeneity: Variety and differences in
   - Networks
   - Computer hardware
   - Operating systems
   - Programming languages
   - Implementations by different developers
8.6 Concurrency Components in distributed systems are executed in concurrent processes, Components access and update shared resources (e.g. variables, databases, device drivers).

8.7 Transparency: Distributed systems should be perceived by users and application programmers as a whole rather than as a collection of cooperating components.

8.1 Make Resources Accessible:
Access resources and share them in a controlled and efficient way. Printers, computers, storage facilities, data, files, Web pages, and networks, ...

8.2 Distribution Transparency

<table>
<thead>
<tr>
<th>Transparency</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Hide differences in data representation and how a resource is accessed</td>
</tr>
<tr>
<td>Location</td>
<td>Hide where a resource is located</td>
</tr>
<tr>
<td>Migration</td>
<td>Hide that a resource may move to another location</td>
</tr>
<tr>
<td>Relocation</td>
<td>Hide that a resource may be moved to another location while in use</td>
</tr>
<tr>
<td>Replication</td>
<td>Hide that a resource is replicated</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Hide that a resource may be shared by several competitive users</td>
</tr>
<tr>
<td>Failure</td>
<td>Hide the failure and recovery of a resource</td>
</tr>
</tbody>
</table>

8.3 Openness
Goal: Open distributed system -- able to interact with services from other open systems, irrespective of the underlying environment:

- Standard rules (protocols/interfaces) to describe services/components
- Flexibility – ability to integrate multiple components
- Achieving openness: At least make the distributed system independent from heterogeneity of the underlying environment:
  - Hardware
  - Platforms
  - Languages

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

9. User Requirements:

- What services the system can provide?
- How easy to use and manage the system?
- What benefits the system can offer?
• What is the ratio of performance/cost?
• How reliable the system is?
• How secure the system can guarantee?

10. Types of Distributed Systems

10.1 Distributed Computing Systems
Many distributed systems are configured for High-Performance Computing Cluster Computing: Essentially a group of high-end systems connected through a LAN:

10.2 Distributed Information Systems
The vast amount of distributed systems in use today are forms of traditional information systems, that now integrate legacy systems. Example: Transaction processing systems.

10.3 Distributed Pervasive Systems
There is a next-generation of distributed systems emerging in which the nodes are small, mobile, and often embedded as part of a larger system.

11. Criterion of Distributed Computer System (Metrics)

• **Latency** – network delay before any data is sent

• **Bandwidth** – maximum channel capacity (analogue communication Hz, digital communication bps)

• **Granularity** – relative size of units of processing required. Distributed systems operate best with coarse grain granularity because of the slow communication compared to processing speed in general

• **Processor speed**

• **Reliability** – ability to continue operating correctly for a given time

• **Fault tolerance** – resilience to partial system failure

• **Security** – policy to deal with threats to the communication or processing of data in a system

• **Administrative/management** domains – issues concerning the ownership and access to distributed systems components

12. Communication Hardware Characteristics: Circuit vs. Packet Switching

• **Circuit switching**
  – Example: telephony
  – Resources are reserved and dedicated during the connection
  – Fixed path between peers for the duration of the connection

• **Packet switching**
–Example: internet

–Entering data (variable-length messages) are divided into (fixed-length) packets

–Packets in network share resources and may take different paths to the destination

**Routing in a wide area network**

![Routing Diagram]

**Routing Table of the network**

<table>
<thead>
<tr>
<th>Routes from A</th>
<th>Routes from B</th>
<th>Routes from C</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td>Link</td>
<td>Cost</td>
</tr>
<tr>
<td>A</td>
<td>local</td>
<td>0</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Routes from D</th>
<th>Routes from E</th>
</tr>
</thead>
<tbody>
<tr>
<td>To</td>
<td>Link</td>
</tr>
<tr>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>6</td>
</tr>
<tr>
<td>D</td>
<td>local</td>
</tr>
<tr>
<td>E</td>
<td>6</td>
</tr>
</tbody>
</table>
13. System Architectures

The architecture include:

– The division of responsibilities between system components.
– The placement of the components on computers in the network.

• **Client-server model:**
  - Most important and most widely distributed system architecture.
  - Client and server roles are assigned and changeable.
  - Servers may in turn be clients of other servers.
  - Services may be implemented as several interacting processes in different host computers to provide a service to client processes:
    - Servers partition the set of objects on which the service is based and distribute them among themselves (e.g. Web data and web servers)

---

**Clients invoke individual servers**

![Diagram of client-server model](attachment://client-server-model.png)

**A service provided by multiple servers**

![Diagram of multiple servers](attachment://multiple-servers.png)
• **Web proxy server**

  – Provides a shared cache of web resources for client machines at a site or across several sites.
  – Increase availability and performance of a service by reducing load on the WAN and web servers.

![Diagram of a web proxy server](image)

• **Peer processes Model**

  – All processes play similar roles without destination as a client or a server.
  – Interacting cooperatively to perform a distributed activity.
  – Communications pattern will depend on application requirements.

![Diagram of a distributed application based on peer processes](image)