Distributed Systems (3 Cr.)

Course Code: CACS352

Compiled by Guru Sharan Giri

Course Descriptions

→Give an insight on how modern distributed systems operate.

Objectives

→To make familiar with different aspect of distributed system, middleware, system level support and different issues in designing algorithms and finally systems.

Unit 1: Introduction (4hrs)

Background

1. Characteristics
2. Design Goals
3. Types of Distributed Systems
4. Case Study: WWW

Background

- 1945→ when the modern computer era began.
- 1945-1985→computers were large and expensive.
- Moreover, for lack of a way to connect them, these computers operated independently from one another.
- mid-1980s → Development of powerful microprocessors

  → invention of high-speed computer networks

Invention of high-speed computer networks

- **Local-area networks or LANs** allow thousands of machines within a building or campus to be connected in such a way that small amounts of information can be transferred in a few microseconds or so.
- Larger amounts of data can be moved between machines at rates of billions of bits per second (bps).
- **Wide-area networks or WANs** allow hundreds of millions of machines all over the earth to be connected at speeds varying from tens of thousands to hundreds of millions bps, and sometimes even faster.

Smartphone as the most impressive outcome

- Packed with sensors, lots of memory, and a powerful CPU, these devices are nothing less than full-fledged computers.
- Of course, they also have networking capabilities.

plug computers and nano computers

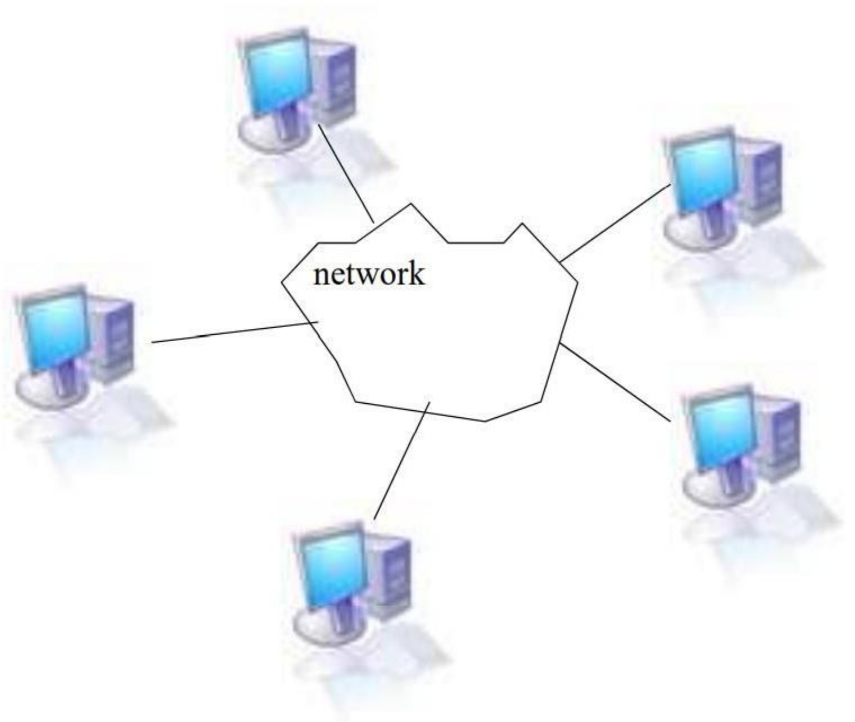
- These small computers, often the size of a power adapter, can often be plugged directly into an outlet and offer near-desktop performance.

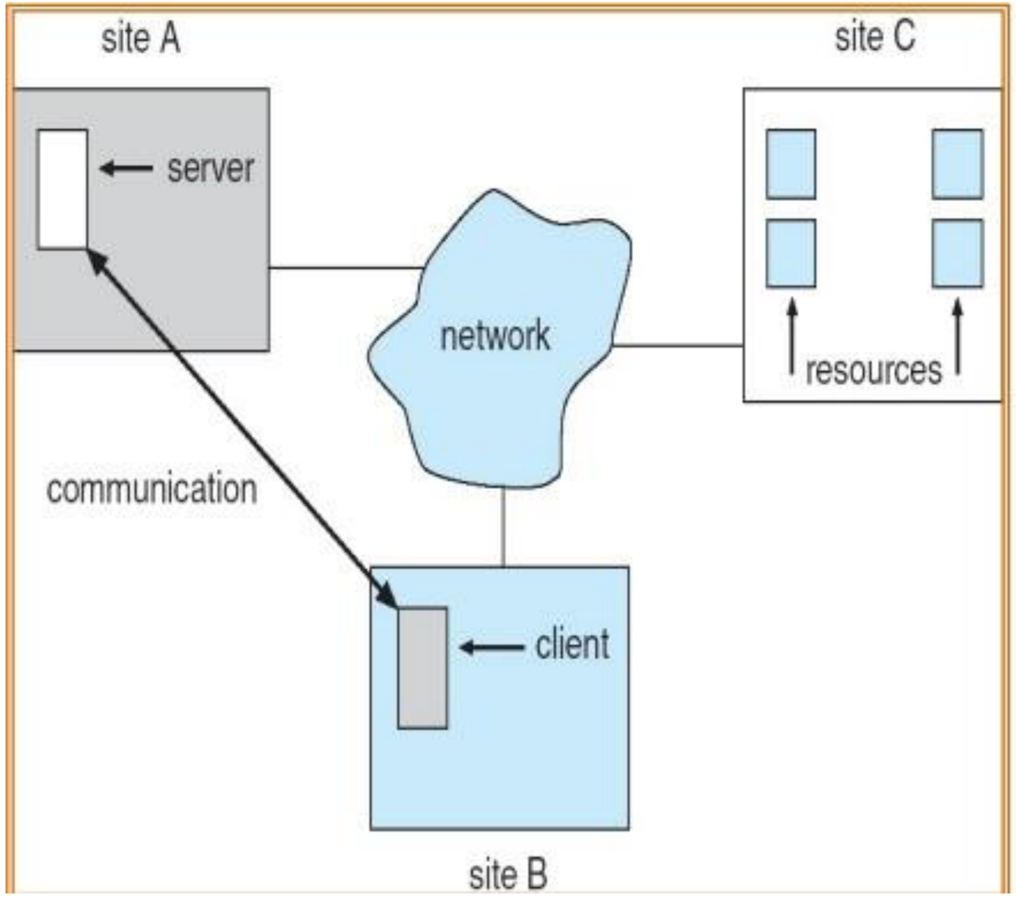
Distributed System

- The size of a distributed system may vary from a handful of devices, to millions of computers.
- The interconnection network may be wired, wireless, or combination of both.
- Moreover, distributed systems are often highly dynamic, in the sense that computers can join and leave, with the topology and performance of the underlying network almost continuously changing.

What is a distributed system?

- A distributed System is a collection of autonomous computer systems that are physically separated but are connected by a centralized computer network that is equipped with distributed system software. The autonomous computers will communicate among each system by sharing resources and files and performing the tasks assigned to them..
  - a collection of computing elements(Node: H/w device or S/w Process) each being able to behave independently of each other.
  - users (people or applications) believe they are dealing with a single system.





1. Group of autonomous hosts, each host executes components and operates distribution applications.
2. Hosts are Geographically dispersed/separated over (LAN, WAN,……)
3. Hosts Connected via a network
4. The network is used to: transfer messages and mail, and execute applications: airline reservation, stock control, ….)

1.1. Characteristic

1:Resource Sharing

- Resource sharing means that the existing resources in a distributed system can be accessed or remotely accessed across multiple computers in the system. Computers in distributed systems shares resources like hardware, software and data.

2:Concurrency

- Concurrency is property of system representing the fact that multiple activities are executed at the same time. The concurrent execution of activities takes place in different components running on multiple machine as part of distributed system. Concurrency reduces the latency and increases the throughput of the distributed system.

1.1 Characteristic

3:Network Communication:
The computers talk to each other over a network. This communication allows them to share data and coordinate their actions.
4: Decentralized Control:
There isn't a single boss computer telling all the others what to do. Instead, control is spread out among all the computers in the system.
5.Fault Tolerance:
If one computer fails, the system can still keep running. The other computers can take over the tasks of the failed one, making the system reliable.
6:Scalability:
You can add more computers the system easily, which helps handle more work or users. The system grows by adding more nodes.

1.1 Characteristic

7. Transparency:

From the user’s perspective, the distributed system should look like a single computer. Users shouldn’t need to know which computer is doing the work or where the data is stored.

8.Heterogeneity:

The computers in a distributed system can be different from each other. They might have different hardware, operating systems, or software, but they can still work together.

Development of powerful microprocessors

Initially, these were 8-bit machines, but soon 16-, 32-, and 64-bit CPUs became common.

Invention of high-speed computer networks

- **Local-area networks or LANs** allow thousands of machines within a building or campus to be connected in such a way that small amounts of information can be transferred in a few microseconds or so.
- Larger amounts of data can be moved between machines at rates of billions of bits per second (bps).
- **Wide-area networks or WANs** allow hundreds of millions of machines all over the earth to be connected at speeds varying from tens of thousands to hundreds of millions bps, and sometimes even faster.

Design Issues/Goals/challenges of Distributed System

Creating a distributed system is not simple, and there are a number of design considerations to take into account. The following are some of the major design issues of distributed systems:

Design issues of the distributed system –

Heterogeneity: Heterogeneity is applied to the network, computer hardware, operating system, and implementation of different developers. A key component of the heterogeneous distributed system client-server environment is middleware. Middleware is a set of services that enables applications and end-user to interact with each other across a heterogeneous distributed system.

Openness: The openness of the distributed system is determined primarily by the degree to which new resource-sharing services can be made available to the users. Open systems are characterized by the fact that their key interfaces are published. It is based on a uniform communication mechanism and published interface for access to shared resources. It can be constructed from heterogeneous hardware and software.

Design Issues/Goals/challenges of Distributed System

Creating a distributed system is not simple, and there are a number of design considerations to take into account. The following are some of the major design issues of distributed systems:

Design issues of the distributed system –

Heterogeneity: Heterogeneity is applied to the network, computer hardware, operating system, and implementation of different developers. A key component of the heterogeneous distributed system client-server environment is middleware. Middleware is a set of services that enables applications and end-user to interact with each other across a heterogeneous distributed system.

Openness: The openness of the distributed system is determined primarily by the degree to which new resource-sharing services can be made available to the users. Open systems are characterized by the fact that their key interfaces are published. It is based on a uniform communication mechanism and published interface for access to shared resources. It can be constructed from heterogeneous hardware and software.

Design Issues/Goals/challenges of Distributed System

Creating a distributed system is not simple, and there are a number of design considerations to take into account. The following are some of the major design issues of distributed systems:

Design issues of the distributed system –

Heterogeneity: Heterogeneity is applied to the network, computer hardware, operating system, and implementation of different developers. A key component of the heterogeneous distributed system client-server environment is middleware. Middleware is a set of services that enables applications and end-user to interact with each other across a heterogeneous distributed system.

Openness: The openness of the distributed system is determined primarily by the degree to which new resource-sharing services can be made available to the users. Open systems are characterized by the fact that their key interfaces are published. It is based on a uniform communication mechanism and published interface for access to shared resources. It can be constructed from heterogeneous hardware and software.

Design Issues of Distributed System

Scalability: The scalability of the system should remain efficient even with a significant increase in the number of users and resources connected. It shouldn’t matter if a program has 10 or 100 nodes; performance shouldn’t vary. A distributed system’s scaling requires consideration of a number of elements, including size, geography, and management.

Security: The security of an information system has three components Confidentially, integrity, and availability. Encryption protects shared resources and keeps sensitive information secrets when transmitted.

Failure Handling: When some faults occur in hardware and the software program, it may produce incorrect results or they may stop before they have completed the intended computation so corrective measures should to implemented to handle this case. Failure handling is difficult in distributed systems because the failure is partial i, e, some components fail while others continue to function.

Design Issues of Distributed System

Concurrency: There is a possibility that several clients will attempt to access a shared resource at the same time. Multiple users make requests on the same resources, i.e. read, write, and update. Each resource must be safe in a concurrent environment. Any object that represents a shared resource in a distributed system must ensure that it operates correctly in a concurrent environment.

Transparency: Transparency ensures that the distributed system should be perceived as a single entity by the users or the application programmers rather than a collection of autonomous systems, which is cooperating. The user should be unaware of where the services are located and the transfer from a local machine to a remote one should be transparent

Types of Distributed System

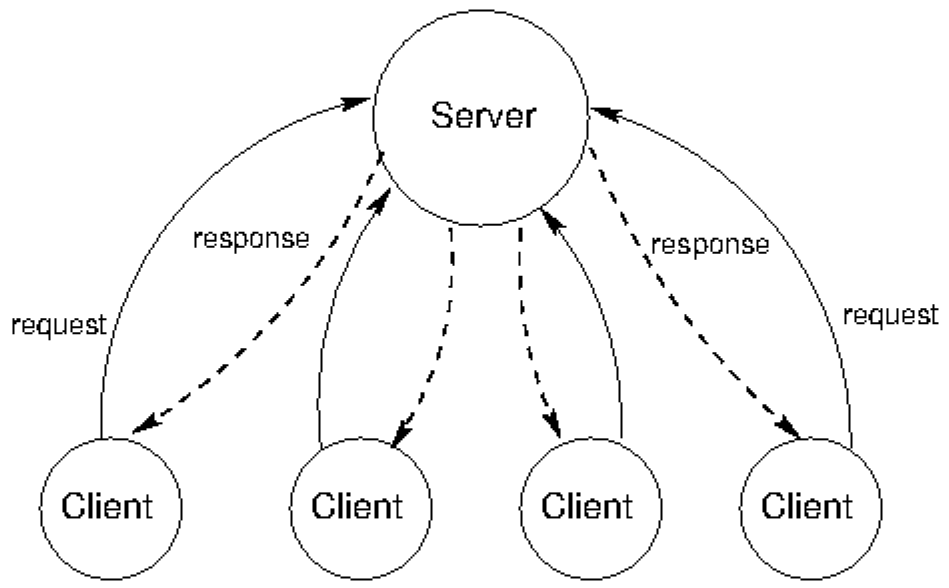
A Distributed System is a Network of Machines that can exchange information with each other through Message-passing. It can be very useful as it helps in resource sharing. It enables computers to coordinate their activities and to share the resources of the system so that users perceive the system as a single, integrated computing facility.

Types of Distributed Systems

1. Client/Server Systems
2. Peer-to-Peer Systems
3. Middleware
4. Three-tier
5. N-tier

1.Client/Server Systems

Client-Server System is the most basic communication method where the client sends input to the server and the server replies to the client with an output. The client requests the server for resources or a task to do, the server allocates the resource or performs the task and sends the result in the form of a response to the request of the client. Client Server System can be applied with multiple servers.



Cont..

Client:

- The client is a user-facing component or software application that requests services, resources, or data from the server.
- Clients are typically end-user devices such as computers, smartphones, tablets, or IoT devices.
- They initiate communication with the server, send requests for specific services, and wait for responses.
- Client applications can be graphical user interfaces (GUIs), command-line tools, or web browsers.

Cont..

Server:

- The server is a central component or software application that provides requested services or resources to clients.
- Servers are typically powerful computers or specialized machines designed to handle multiple client requests simultaneously.
- They listen for incoming client requests, process those requests, and send back the appropriate responses.
- Server applications can include web servers, database servers, email servers, and more.

Cont..

Examples of client-server applications include

Web Browsing: Web browsers (clients) request web pages from web servers, which respond with the requested web content.

Email: Email clients (e.g., Outlook, Gmail) retrieve emails from email servers (IMAP or POP3 servers) and send emails via SMTP servers.

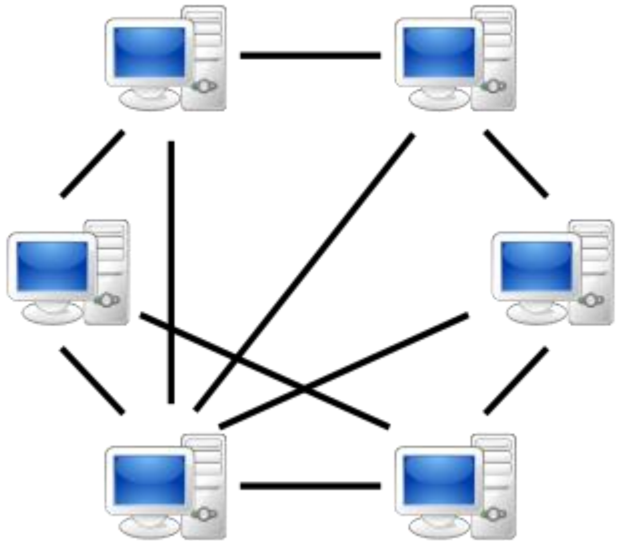
Online Gaming: In multiplayer online games, game clients interact with game servers to synchronize gameplay and handle multiplayer interactions.

Database Management: Database clients (e.g., database management systems) request data from database servers and submit queries for processing.

File Sharing: In a file-sharing system, client applications request files from file servers or peer-to-peer (P2P) networks.

2. Peer-to-Peer Systems:

Peer-to-Peer System communication model works as a decentralized model in which the system works like both Client and Server. Nodes are an important part of a system. In this, each node performs its task on its local memory and shares data through the supporting medium, this node can work as a server or as a client for a system. Programs in the peer-to-peer system can communicate at the same level without any hierarchy.



Cont..

Here are the key characteristics and concepts of P2P architecture.

Decentralization: P2P networks operate without a central server or authority, relying on direct communication between peers.

Resource Sharing: Peers share resources like files, computing power, storage, and bandwidth directly with each other.

Autonomy: Each peer in a P2P network operates independently and makes its own decisions.

Scalability: P2P networks can grow by adding more peers, increasing their capacity and robustness.

Examples of P2P Applications

File Sharing: BitTorrent and eMule enable users to share files directly.

Cryptocurrencies: Bitcoin and Ethereum use P2P networks for transaction verification.

3. Middleware:

Middleware can be thought of as an application that sits between two separate applications and provides service to both. It works as a base for different interoperability applications running on different operating systems. Data can be transferred to other between others by using this service.

Middleware Is software which lies between an operating system and the applications running on it.

Essentially functioning as hidden translation layer,middleare enables communication and data management for distributed applications.

Using middleware allows users to perform such requests as submitting forms on a web browser or allowing the web server to return dynamic web pages based on a users profile.

Services offered by middleware

1. Communication Services:

- Procedure calls across network
- Remote-object method invocation
- Message queuing system etc..

2. Information system service:

- Large scale system wide naming service
- Advance directory service
- Location service for tracking mobile object
- Data caching and replication

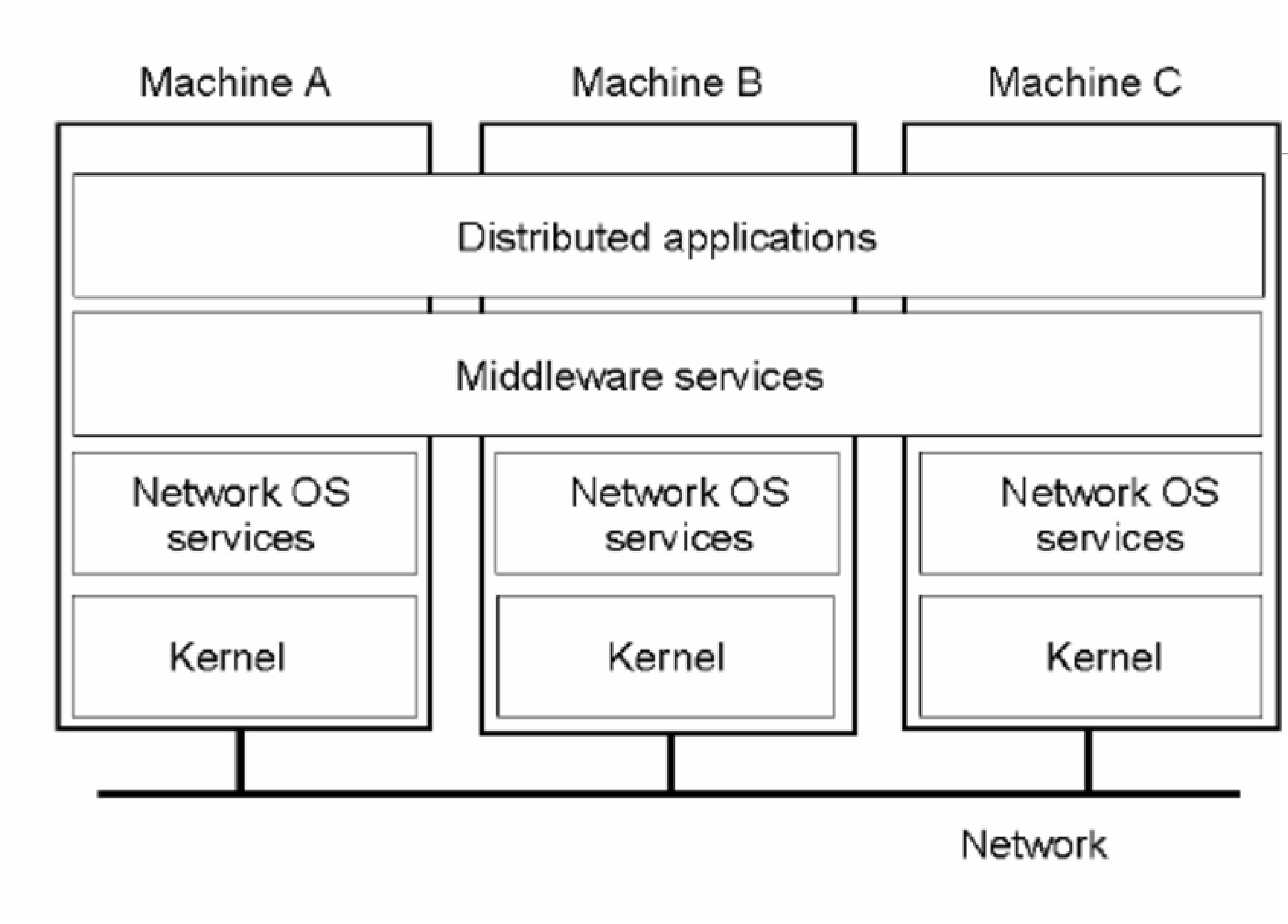
3. Control Service:

- Distributed transaction processing
- Code migration

4. Security service

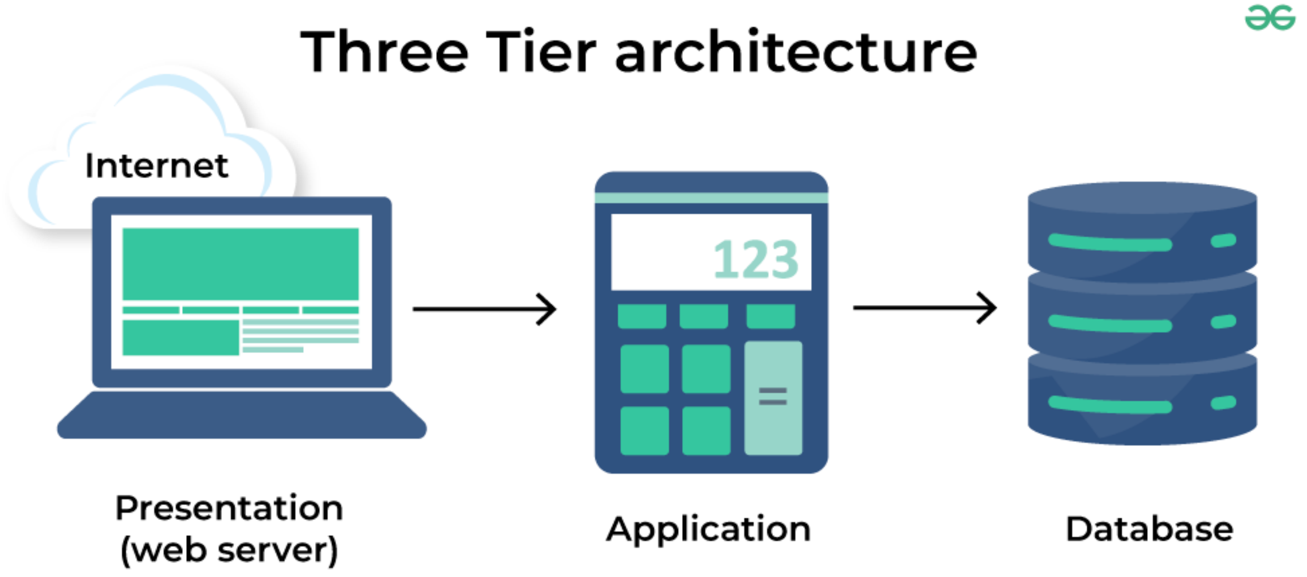
- Authentication and authorization service
- Encryption service etc..

Cont..


Machine A
Machine B
Machine C
Distributed applications
Middleware services
Network OS services
Network OS services
Network OS services
Kernel
Kernel
Kernel
Network

4. Three-tier:

Three-tier system uses a separate layer and server for each function of a program. In this data of the client is stored in the middle tier rather than sorted into the client system or on their server through which development can be done easily. It includes an Application Layer, Data Layer, and Presentation Layer. This is mostly used in web or online applications.



Cont..

Presentation Tier

It is the user interface and topmost tier in the architecture. Its purpose is to take request from the client and displays information to the client. It communicates with other tiers using a web browser as it gives output on the browser. If we talk about Web-based tiers then these are developed using languages like- HTML, CSS, JavaScript.

Application Tier

It is the middle tier of the architecture also known as the logic tier as the information/request gathered through the presentation tier is processed in detail here. It also interacts with the server that stores the data. It processes the client's request, formats, it and sends it back to the client. It is developed using languages like- Python, Java, PHP, etc.

Data Tier

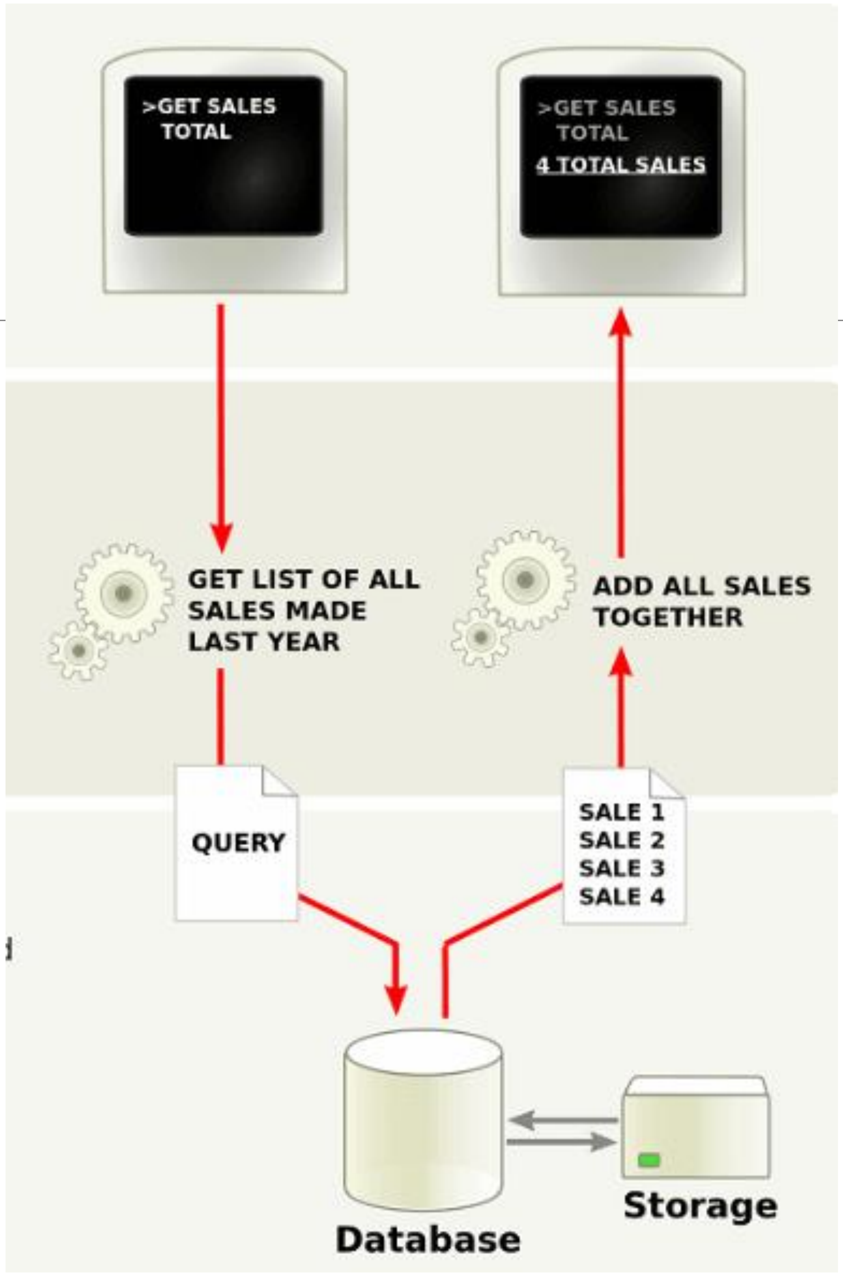
It is the last tier of the architecture also known as the Database Tier. It is used to store the processed information so that it can be retrieved later on when required. It consists of Database Servers like- Oracle, MySQL, DB2, etc. The communication between the Presentation Tier and Data-Tier is done using middle-tier i.e. Application Tier.

5. N-tier:

N-tier is also called a multitier distributed system. The N-tier system can contain any number of functions in the network. N-tier systems contain similar structures to three-tier architecture. When interoperability sends the request to another application to perform a task or to provide a service. N-tier is commonly used in web applications and data systems.

N-Tier Architecture, also known as Multi-Tier Architecture it divides the application into various number of tiers based on there complexity and requirements. following are the some of the tiers included in the architecture.

N-tier Architecture



Presentation Layer

Application Layer/Logic layer

Data Layer

The Eight Fallacies of Distributed Computing

In 1994, Peter Deutsch drafted 7 assumptions and in 1997 James Gosling father of java programing language added another fallacy.

The eight fallacies of distributed computing are:

The network is reliable: Assuming that the network will always work perfectly without any interruptions or delays.

Latency is zero: Believing that data can be sent and received instantly without any delay. **Bandwidth is infinite**: Thinking that the network can handle unlimited data without any restrictions.

The network is secure: Assuming that the data transferred over the network is always safe from attacks or unauthorized access.

Cont..

Topology doesn't change: Believing that the layout and connections of the network will remain constant.

There is one administrator: Assuming that only one person or entity is managing the network, making it simpler to control.

Transport cost is zero: Thinking that sending data over the network is free and doesn't incur any costs.

The network is homogeneous: Assuming that all parts of the network will work the same way and that there are no differences in the hardware, software, or protocols used.

Advantages:

Scalability: Distributed systems can easily grow by adding more computers. This means they can handle more work as needed without major changes.

Reliability: If one computer in the system fails, the others can keep working. This makes the system more reliable and less likely to completely fail.

Performance: By sharing tasks among multiple computers, distributed systems can process data and complete tasks faster.

Resource Sharing: Distributed systems allow different computers to share resources like storage, data, and applications, making better use of available resources.

Flexibility: Different parts of the system can be updated or replaced independently, making it easier to maintain and upgrade.

Disadvantages:

Complexity: Managing a distributed system is more complicated because you have to ensure all parts work together correctly.

Security: With data being transferred between multiple computers, there are more opportunities for security breaches and attacks.

Network Dependency: Distributed systems rely heavily on the network. If the network is slow or fails, it can affect the whole system.

Data Consistency: Ensuring that all computers have the most up-to-date data can be challenging, especially when updates happen frequently.

Cost: Setting up and maintaining a distributed system can be expensive due to the need for multiple computers and network infrastructure.