Client/Server computing is a computing model in which client and server computers communicate with each other over a network. In client/server computing, a server takes requests from client computers and shares its resources, applications and/or data with one or more client computers on the network, and a client is a computing device that initiates contact with a server in order to make use of a shareable resource.

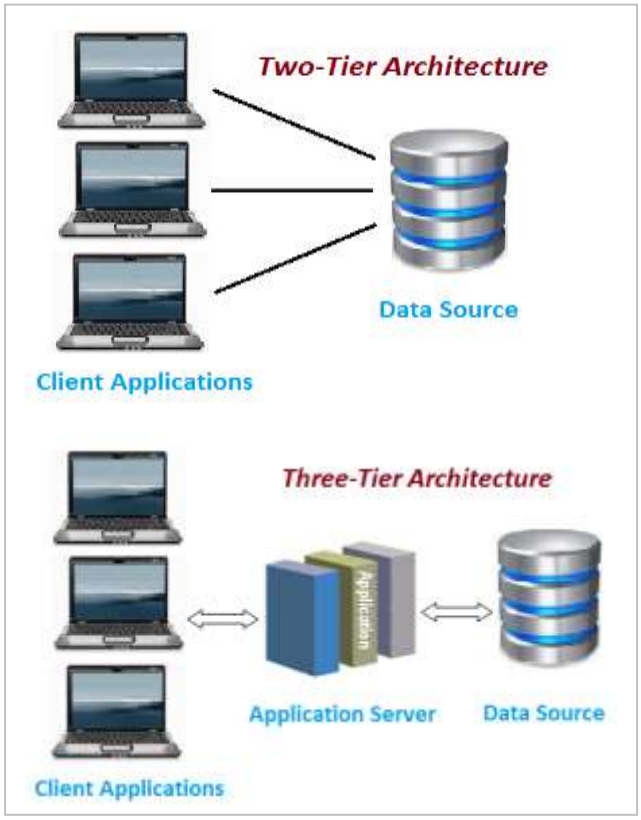
1.1 Introduction to Client / Server

A client-server network is a *central computer, also known as a server, which hosts data and other forms of resources*. *Clients such as laptops and desktop computers contact the server and request to use data or share its other resources with it.*

A client-server network is designed for *end-users, called clients, to access resources such as files, songs, video collections, or some other service from a central computer called a server.* A server's sole purpose is to do what its name implies - ***serve its clients!***

Client and Server Hardware : Client/server networking grew in popularity many years ago as personal computers (PCs) became the common alternative to older *mainframe* computers.

- *Client devices are typically PCs with network software applications installed that request and receive information over the network. Mobile devices, as well as desktop computers, can both function as clients.*
- A *server device typically stores files and databases* including more complex applications like Web sites. Server devices often feature higher-powered central processors, more memory, and larger disk drives than clients.
- *Servers are classified by the services they provide. For example, a web server serves web pages and a file server serves computer files. A shared resource may be any of the server computer's software and electronic components, from programs and data to processors and storage devices. The sharing of resources of a server constitutes a service.*



1.1.1 2-tier Architecture

The two-tier is *based on Client Server architecture. The direct communication* takes place between client and server. There is *no intermediate* between client and server as shown in figure.. Because of tight coupling a 2-tiered application will run faster.

The Two-tier architecture is divided into two parts:

1) Client Application (Client Tier or Presentation Tier) : it contains User Interface (UI) part of our application. This layer helps to display result after store or retrieve data to/from server. E.g. *login page of Gmail where an end user could see text boxes and buttons to enter user id, password and to click on sign-in.*

2) Database (Data Tier) : On client application side the code is written for saving the data in the database server e.g. Oracle, MySQL, MSSQL Server. *Client sends the request to server and it process the request & send back with data.* The main problem of two tier architecture is the server *cannot respond multiple request same time, as a result it cause a data integrity issue*.

Advantages:

1. Easy to maintain and *modification is bit easy*
2. *Communication is faster*

Disadvantages:

1. In two tier architecture application *performance will be degrade upon increasing the users.*
2. *Cost-ineffective*

Example : Most of the Desktop applications are 2-tier architecture

1.1.2 3-tier Architecture

Three-tier architecture typically *comprise a presentation tier, a business or data access tier, and a data tier*. Three tiers or layers in the 3 tier architecture are as follows:

1) Client Tier or Presentation Tier : It *contains UI part of our application*. This layer is used for the design purpose where *data is presented to the user or input is taken* from the user. *For example designing registration form which contains text box, label, button* etc.

2) Business Tier : In this layer all *business logic written like validation of data, calculations, data insertion* etc. This acts as a *interface between Client layer and Data Access Layer*. This layer is also called *the intermediary layer* helps to make communication faster between client and data layer.

3) Data Tier : In this layer actual database is comes in the picture. Data Access Layer contains *methods to connect with database and to perform insert, update, delete, get data* from database based on our input data.

Advantages

1. *High **performance**, **lightweight** persistent objects*
2. **Scalability** – *Each tier can scale* horizontally
3. *Performance* – Because the Presentation tier can cache requests, network utilization is minimized, and the load is reduced on the Application and Data tiers.
4. High degree of ***flexibility*** in deployment platform and configuration
5. Better ***Re-use*** *and Improve Data* ***Integrity, Security*** – Client is not direct access to database.
6. Easy to **maintain** and modification is bit easy, won't affect other modules
7. In three tier architecture application *performance is good.*

Disadvantages

1. Increase Complexity/Effort

Example: Most of the web applications are 3-tier architecture

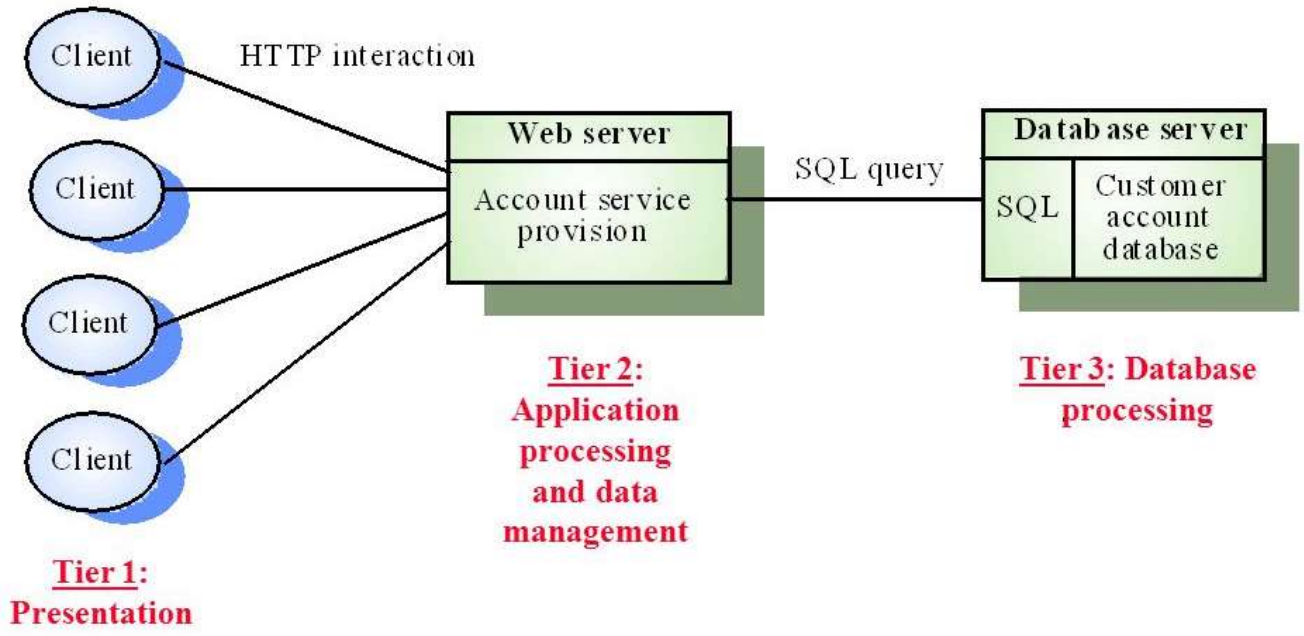


Fig. 3-tier architecture of Online Banking System

1.2 Benefits and Characteristics of Client / Server Architecture.

Benefits and Characteristics of Client / Server Architecture.

- **24x7 Accessibility**: With a peer-to-peer network, if a user needs to access a file residing on another computer, that computer needs to be powered on. This is not practical with client devices that are generally powered off when not in use. With a client-server network, *the server is always-on, always available,* so files and applications can be accessed at anytime.
- **Improved Collaboration**: The server in a client-server network can *act as a centralized hub for storing and sharing files.* This configuration allows multiple users to access files and makes *changes to a single centralized copy*. This also helps minimize version control issues that often arise from managing multiple versions of the same file.
- **Centralized, Client Backups**: Servers can be configured to *automatically backup* client computers and also restore data based on those backup images, in the case of a client hard drive failure.
- **Remote Access**: Servers *support remote access which enables employees, partners, and customers, to access data* on the server without physically being in front of the system.
- **Server Backups**: the server to seamlessly store *multiple copies of its data on additional internal hard drives,* so if one of its hard drives fails, it can quickly recover the data with minimal system downtime.

- **Enhanced Security**: Servers can be configured to *control access to the server's data and other resources on a per-user basis*. This ensures that only individuals with *proper permissions access* specific data and applications residing on the server. And with Intel *Advanced Encryption Standard* New Instructions (Intel AES-NI), data passing between the server and clients is encrypted to prevent data from being compromised in transit.
- **Better Client Performance**: In a peer-to-peer network, clients also have to act as servers, "serving up" services to other clients on the network. This can negatively impact performance of those clients. This *computational burden is lifted by having a high performance*, dedicated to supporting the clients.
- **Shared, System-Wide Services**: Servers provide *shared, centralized services* for clients to access such as file, print, email, database, and web hosting.
- **Enhanced Reliability**: servers support Error Correcting Code (ECC) memory which helps protect your business-critical data and prevent system errors by *automatically detecting and correcting memory errors.*
- **Business Growth**: Peer-to-peer networks are limited in terms of the number of users. A client-server network *server is scalable for your needs, allowing room for growth as you business grows.*

Disadvantages-compared to peer-peer network

- **Overloaded servers**
  When there are frequent simultaneous client requests, servers severely get overloaded, forming traffic congestion. But in a P2P network adding more nodes will increase its bandwidth since it's calculated as the sum of bandwidths of each node in the network.(slideshare 2011)
- **Impact of centralized architecture**
  Since its centralized if a critical server fails, client requests are not accomplished. Therefore client/server lacks robustness of a good P2P network (resources are distributed among many nodes).
- **Congestion in Network :** Too many requests from the clients may lead to congestion, which rarely takes place in P2P network. Overload can lead to breaking-down of servers. In peer-to-peer, the total bandwidth of the network increases as the number of peers increase.
- Client-Server architecture is **not as robust** as a P2P and if the server fails, the whole network goes down. Also, if you are downloading a file from server and it gets abandoned due to some error, download stops altogether. However, if there would have been peers, they would have provided the broken parts of file.
- **Cost** : It is very expensive to install and manage this type of computing.
- You **need professional IT people** to maintain the servers and other technical details of network.

*Characteristics Of Client Server

The client / server refers to a *mode of communication between multiple computers on a network that distinguishes one or more clients on the server*: each client software can send requests to a server. A server can be specialized in server applications, files, terminals, or e-mail. The client-server characteristic describes the *relationship of cooperating programs in an application*. The server component provides a *function or service to one or many clients*, which initiate requests for such services.

Characteristics of a server:

- It is initially *passive* (or slave, waiting for a query);
- It *is listening, ready to respond* to requests sent by clients;
- When a request comes, he treats it and *sends a response.*

Characteristics of a client:

- It is the *first active (or master*);
- *Sends requests* to the server;
- It expects and *receives responses* from the server.

The client and the server must of course use the same communication protocol. A server is generally capable of serving multiple clients simultaneously. Another type of network architecture is the peer (peer-to-peer in English, or P2P), in which each computer or software is both client and server.

Comparison between Client Server and Peer-to-peer model

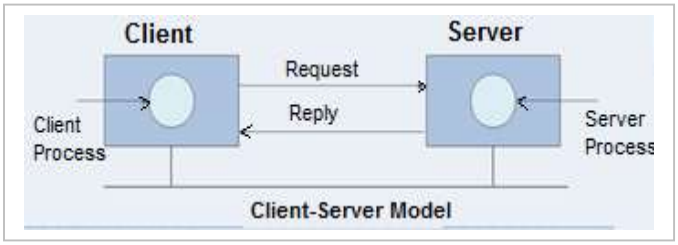
- In the client–server model, the server is often designed to operate as a centralized system that serves many clients. The computing power, memory and storage requirements of a server must be scaled appropriately to the expected work-load (i.e., the number of clients connecting simultaneously). Load-balancing and failover systems are often employed to scale the server implementation.
- In a peer-to-peer network, two or more computers (peers) pool their resources and communicate in a decentralized system. Peers are coequal, or equipotent nodes in a non-hierarchical network. Unlike clients in a client–server or client–queue–client network, peers communicate with each other directly. In peer-to-peer networking, an algorithm in the peer-to-peer communications protocol balances load, and even peers with modest resources can help to share the load. If a node becomes unavailable, its shared resources remain available as long as other peers offer it. Ideally, a peer does not need to achieve high availability because

other, redundant peers make up for any resource downtime; as the availability and load capacity of peers change, the protocol reroutes requests.

- Both client-server and master-slave are regarded as sub-categories of distributed peer-to-peer systems.

1.3 Client / Server Models

The **client-server model** is a *distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients*. Often clients and servers communicate over a computer network on separate hardware, but both client and server may reside in the same system. A *server* host *runs one or more server programs* which share their resources with clients. A *client does not share* any of its resources, but requests a server's content or service function. Clients therefore initiate communication sessions with servers which await incoming requests. *Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.*



The client-server model describes how a server provides resources and services to one or more clients. Examples of servers include web servers, mail servers, and file servers. Each of these servers *provide resources to client devices,* such as desktop computers, laptops, tablets, and smartphones. Most servers have a *one-to-many relationship with clients, meaning a single ser*ver can provide resources to multiple clients at one time.

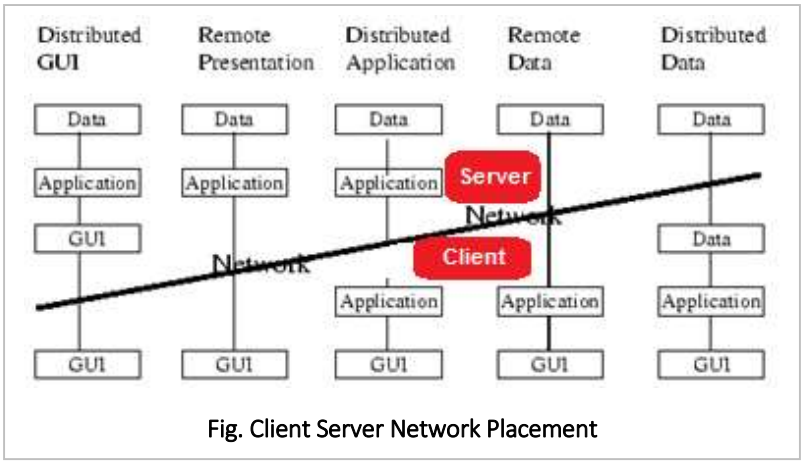


Fig. Client Server Network Placement

1.3.1 Distributed GUI or Presentation

The presentation layer can be split. The server that does the work for the *presentation layer is on one side of the network while the screen is on the other side.*

- only presentation management function shared between client and server
- everything else remains on the server
- screen-scraping (emulation-based) applications
- GUI placed in front of existing character based interface first step in migration of legacy applications to a GUI

1.3.2 Remote Presentation

The X windowing system is an example of this strategy. The *presentation layer talks to the X-Protocol.* The location of the screen is independent of the location of the rest of the application.

- presentation manager entirely on client
- presentation logic, data logic and data manager on server
- X Window System, Web applications where clients are Web browsers

1.3.3 Distributed Application/ Logic

Some of the *logic that makes the application resides on the client and some of the logic resides on the server.* This is popular with stored procedures in the database technology. The logic that is close to the data is placed in the stored procedures residing near the data. The logic that is more function driven is evoked in the client.

- application is split into presentation logic and data logic component
- all presentation management activities on workstation
- all data management activities on the server

1.3.4 Remote Data Access

This is sometimes called ``fat client''. The client *has both the presentation and application logic. Only the database component is access over the network*. Most of distributed applications written in the early 1990s are based on this technology.

- database manager resides on server
- presentation management and data logic reside on client
- typical of client/server DBMSs (DB2, Oracle, Informix, etc.)

1.3.5 Distributed Data.

Here the database is distributed over the network in both vertical and horizontal partitions. *The application needs not know where the data is located.*

- portions of the database reside on client
- portions of the database reside on server
- DBMS manages communication involved
- limited implementation of full-fledged DDBMS functionality

1.3.6 Fat vs. Thin

Thin Clients : A thin client is designed to be especially small so that the *bulk of the data processing occurs on the server*. Although the term thin client often refers to software, it is increasingly used for the computers, such as network computers and Net PCs, that are *designed to serve as the clients for client/server architectures*. A *thin client is a network computer without a hard disk drive.* They act as a simple terminal to the server and require constant communication with the server as well.

Thin clients provide a desktop experience in environments where the end user has a well-defined and regular number of tasks for which the system is used. *Thin clients can be found in medical offices, airline ticketing, schools, governments, manufacturing plants and even call centers*. Along with being *easy to install, thin clients also offer a lower total cost of ownership over thick clients.*

Fat Clients : In contrast, a thick client (also called a fat client) is one that *will perform the bulk of the processing in client/server applications.* With thick clients, there is ***no*** *need for continuous server communications* as it is mainly communicating archival storage information to the server. As in the case of a thin client, the term is often used to refer to software, but again is also used to describe the networked computer itself.

A fat client (sometimes called a thick client) is a networked computer with *most resources installed locally*, rather than distributed over a network as is the case with a thin client. Most PCs (personal computers), *for example, are fat clients because they have their own hard drive DVD drives, software applications and so on.*

Thick vs. Thin - A Quick Comparison

Thin Clients	Fat Clients
- Easy to deploy as they require *no extra or specialized software installation* - Needs *to validate with the server* after data capture - If the *server goes down, data collection is halted* as the client needs constant communication with the server - *Cannot be interfaced* with other equipment (in plants or factory settings for example) - Clients *run only and exactly* as specified by the server - More downtime -Portability in that all applications are on the server so *any workstation can access* - *Opportunity* to use older, outdated PCs as clients - *Reduced security threat*	- More expensive to deploy and more work for IT to deploy - *Data verified by client* not server (immediate validation) -*Robust technology* provides better uptime -Only needs *intermittent communication* with server - *Require more resources* but less servers - Can store local files and applications *- Reduced server demands* *- Increased security issues*