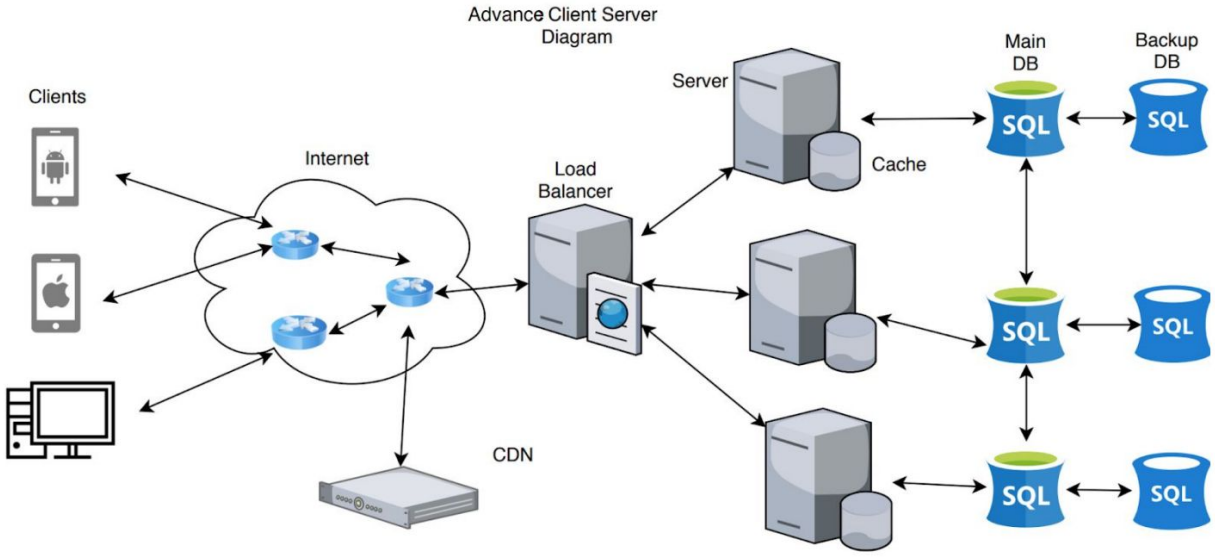
Client-server Architecture

Definition

- Client-server architecture is a computing model in which the server hosts, delivers and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to a central server over a network or internet connection. Client-server architecture is also known as a networking computing model or client-server network because all the requests and services are delivered over a network.
- **Vocabulary for components and connectors:**
    - Client: a piece of software or application that takes the input and sends request to the servers.
    - Server: a piece of software that receives and processes requests from clients.
    - Load balancer: responsible for distributing incoming network traffic across a group of backend servers to optimize resource usage
    - Network layer protocols such as TCP/IP

Topological constraints



The flow of the data is unidirectional and forms a cycle. It is usually initiated by the client requesting some kind of data and the server processing the request and sending some kind of data back to the client via a protocol. Clients cannot directly talk to each other.

A typical topological data flow goes as follows:

1. Client requests data from server

2. Load balancer routes the request to the appropriate server
3. Server processes the request client
4. Server queries appropriate database for some data
5. Database returns the queried data back to the server
6. The server processes the data and sends the data back to the client
7. This process repeats

Applicable problems

- The client-server architecture is most useful for applications that require a separation or abstraction of concerns between the client and the server; it is meant for systems with high interoperability. The client-server architectural style helps applications improve performance in scalability.
- In systems that need separation of functionality, the client-server architecture design is most applicable. Request validation and input could be handled from the client side while the load balancer routes the request to the server for adequate processing. The server will be responsible for processing the client's request, and returning the result via the right protocol. These layers (client and server) complete tasks independently and they are useful for abstracting functionality; for example, the client does not need to know how the server handles user authentication or request validation.
- With the separation of functionality comes the ability of each layer to function more efficiently at large scale. Modern techniques have been developed within the client-server architecture to solve scalability challenges like load balancing, sharding, and partitioning. These techniques provide performance improvements for multiple requests on the server side of the architecture and will be useful for software programs that deal with multiple requests/users.

Resilience to change

- This architecture style is extremely flexible and can be adapted to the user and problem set.
- It can also be combined with other architecture types on the client or server sides.
- As functional and nonfunctional requirements change, modules can be updated without altering the client-server architecture or disrupting service.
- Since the data being passed between the client and server, and services the server provides are entirely up to the developers there is an infinite amount of ways to use this architecture style to solve problems that may arise in the future.

Negative behaviours

- Could be a greater potential for failure because of centralized control:
    - Servers help in administering the whole set-up

    - If the servers go down then entire service goes down
- Particularly vulnerable to Denial of Service (DOS) attacks because the number of servers is considerably smaller than the number of clients
    - DOS attack is a cyber-attack in which perpetrators aim to make a network resource unavailable
        - Done by flooding the resource with requests in an attempt to overload the system
        - Results in legitimate requests not being fulfilled
    - DDOS attack is a distributed denial-of-service attack
        - The incoming traffic comes from multiple clients
        - Cannot stop the attack by blocking a single client
- Very expensive to install and manage the network
    - Server machines are much more powerful than standard client machines, which means that they are more expensive
    - Requires hiring employees with networking and infrastructure knowledge, in order to manage the system

Supported NFPs

- Scalability: capable of horizontal or vertical scaling of the system
    - Horizontal scaling: adding or removing servers in the network
    - Vertical scaling: migrating to larger and faster server machines
- Availability: servers typically do not have to shutdown or restart for many days
    - Server uptime is possible during maintenance, with server duplication
- Dependability: processing and resource allocation is done by a dedicated set of machines
    - These servers can be optimized to complete a certain task quickly and efficiently
- Heterogeneity: clients are consumers of services and servers are providers of services
    - Clear separation of concerns results in the system being composed of disparate parts
    - For example, one or more servers may fail, but the system can still function as long as the other servers offer the same services

Inhibited NFPs

- Robustness: if many clients send simultaneous requests to the server, and there is only a single centralized server, then this might overload the server and slow down the performance drastically. This could possibly lead to a server failure, in which case the whole system goes down and the clients are not able to get any responses back.
- Transparency: since clients only make requests to the server with their input data, they don't see how the data will be distributed across the servers. It may seem like there only exists a single, central server for a user.

Comparison with other architectures

- Client-server, layered, and pipe and filter architectures are similar in their objective.
    - Client-server can be thought of as a variation of layered architecture with two layers.
    - Pipe and filter only allows unidirectional flow of information, whereas client-server and layered architectures allow bidirectional flow.

Credit:

Students:

N2D2 (Noman Shahid, Naveed Hasan Sufi, Dylan Martyn Desrosier, Sangmyung Park)

SharePlay (Milosh Zelembaba, Ayomide Awoyelu, Hengzhe Zhang, Danliang Wang)

Instructor and TAs:

Mei Nagappan, Achyudh Ram Keshav Ram, Aswin Vayiravan, Wenhan Zhu