System Design Handbook

System Design Basics

1) Try to break the problem into simpler modules (Top down approach)
2) Talk about the trade-offs
   (No solution is perfect)
   Calculate the impact on system based on all the constraints and the end test cases.

Scope the problem

Focus on interviewer's intentions.

Ask questions (Constraints & Functionality Requirements)

Abstract Design

Finding & addressing bottlenecks

Rationalize ideas and inputs.

System Design Basics (contd.)

1) Architectural pieces / resources available
2) How these resources work together
3) Utilization & Tradeoffs

- Consistent Hashing
- CAP Theorem
- Load balancing
- Queues
- Caching
- Replication
- SQL vs No-SQL
- Indexes
- Proxies
- Data Partitioning

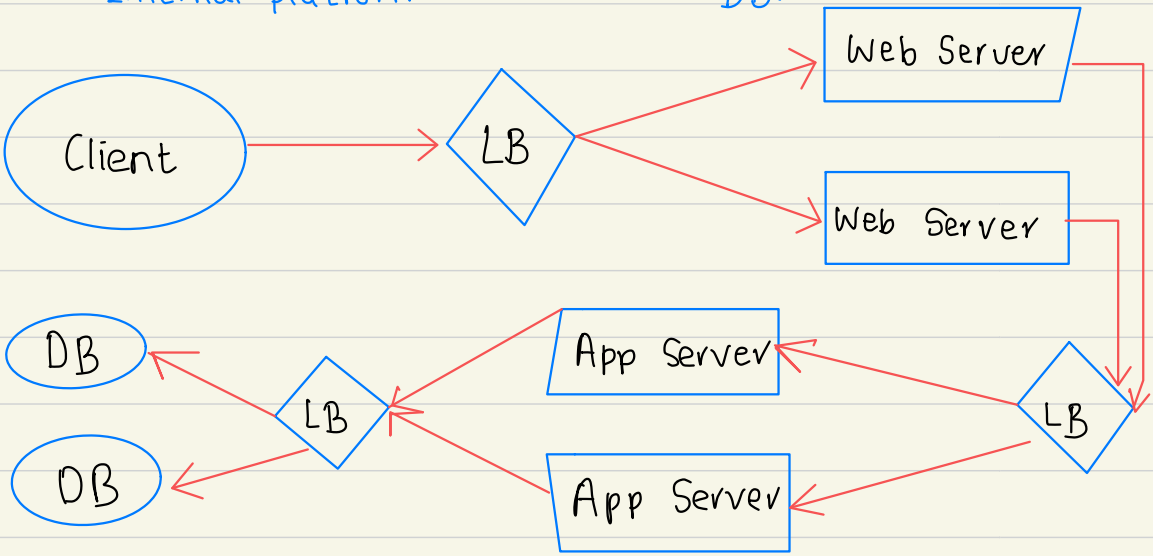
Load Balancing

(Distributed System)

Types of distribution
- Random
- Round - robin
- Random (weights for memory & CPU cycles)

To utilize full scalability & redundancy, add 3 LB

1) User $\xleftrightarrow{LB1}$ Web Server
2) Web Server $\xleftrightarrow{LB2}$ App Server / Cache Server (Internal platform)
3) Internal platform $\xleftrightarrow{LB3}$ DB.



Smart Clients

Takes a pool of service hosts & balances load.

→ detects hosts that are not responsive

→ recovered hosts

→ addition of new hosts

Load balancing functionality to DB (cache, service)

* Attractive solution for developers (Small scale systems)

As system grows → LBs (Standalone servers)

Hardware Load Balancers:

Expensive but high performance.

e.g. Citrix NetScaler

Not trivial to configure.

Large companies tend to avoid this config. or use it as 1st point of contact to their system to serve user requests &

Intra network uses smart clients / hybrid solution → (Next page) for load balancing traffic.

Software Load Balancers

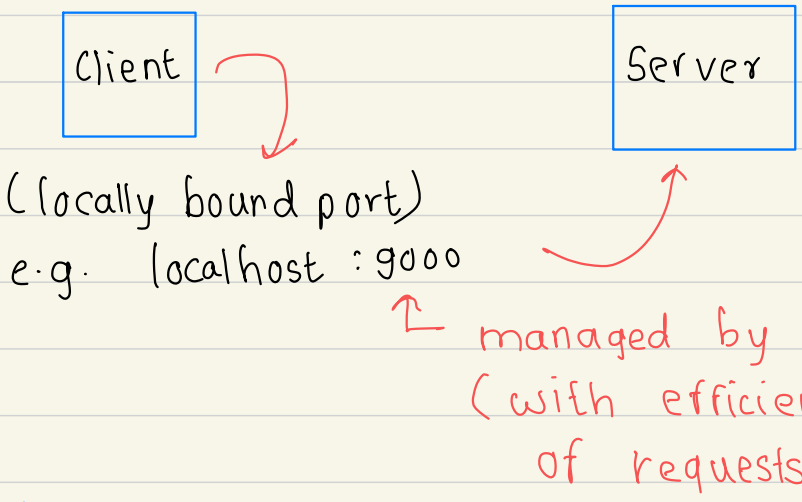
- No pain of creation of smart client
- No cost of purchasing dedicated hardware

→ hybrid approach

HAProxy ⇒ OSS Load balancer

⇓

1) Running on client machine



2) Running on intermediate server: Proxies running betn diff. server side components

HAProxy
- manages health checks
- removal & addition of machines
- balances requests a/c pools.

World of Databases

SQL vs. NoSQL

Relational Database

1) Structured
2) Predefined schema
3) Data in rows & columns

Row ⇒ One Entity Info

Column ⇒ Separate data points

- MySQL
- Oracle
- MS SQL Server
- SQLite
- Postgres
- MariaDB

Non-relational Database

1) Unstructured,
2) distributed
3) dynamic schema

- Key-Value Stores
- Document DB
- Wide-Column DB
- Graph DB

NoSQL

Key-Value Store

Data ⇒ array of key-value pair

Key ⇒ attribute

Value ← linked to

Redis

Voldemart

Dynamo

Document DB

Data ⇒ documents

⇓ grouped into

Collections

Each doc can be different.

CouchDB

MongoDB

Wide-Column DB

Instead of tables, column families.

↳ Container for rows

No need of knowing all columns upfront.

Each row ⇒ diff. no of columns.

Analysis of large datasets.

Cassandra

HBase

Graph DB

Data whose relations are best represented in Graphs.

⇒ Nodes (Entities)

⇒ Properties (information of entities)

⇒ Lines (Connections betn entities)

Neo 4J

InfiniteGraph