System Design

- Vedant Dhiman

Schedule

- Generic Components Of System Design
- Dimensions of system design
- Trade-Offs in Large Scale System
  - Performance vs Scalability
  - Latency vs Throughput
- Practice – System Design Problems

Design Problem

Design Vending Machine

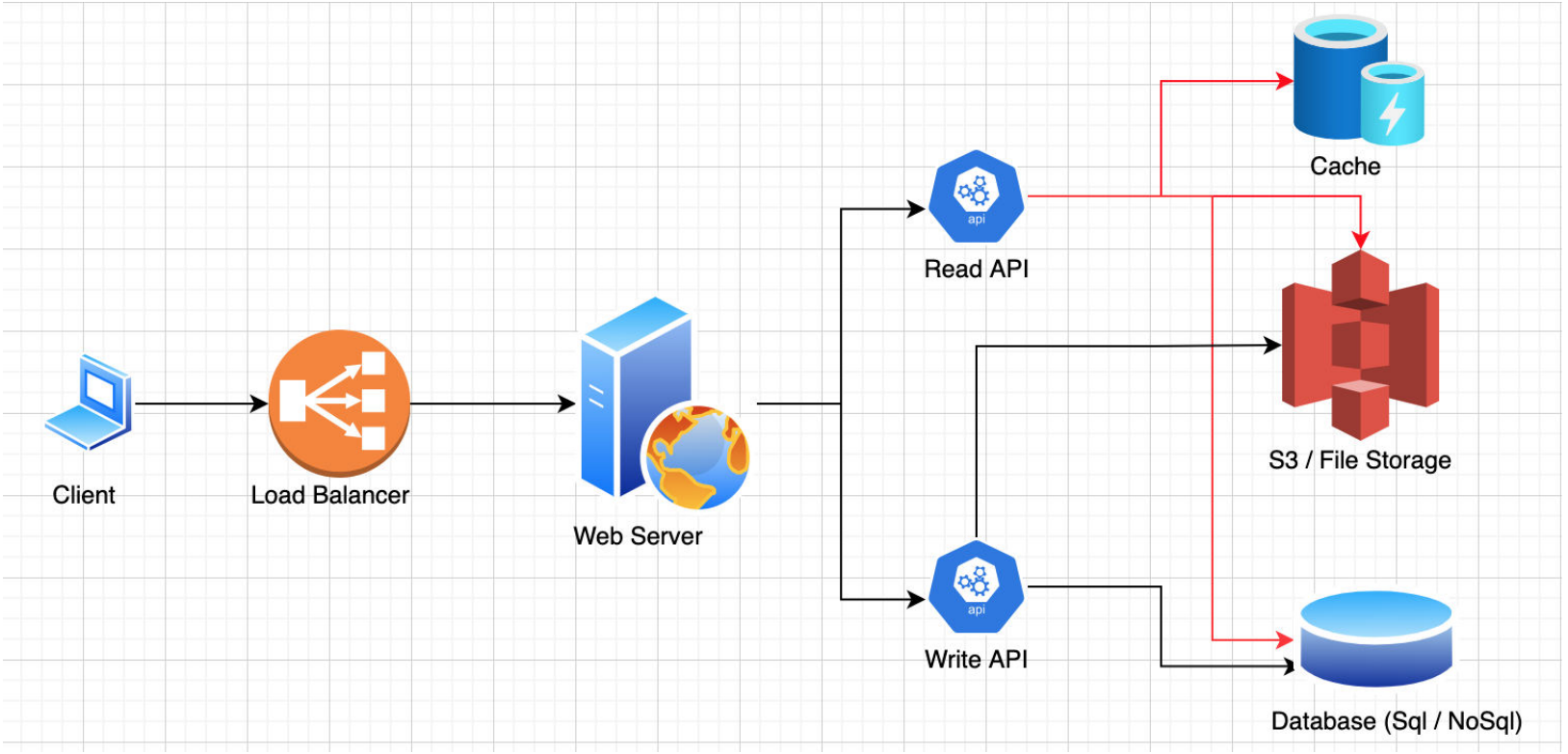
Generic Components

What are the most frequently used system design components ?

Generic Components

- Load Balancer
- CDN
- API
- DB
  - Database Sharding
  - Memory Management Techniques
- Cache
  - Caching Strategies
  - Consistent Hashing
- File Storage
- HTTP vs HTTPS vs Web sockets
- MVC Architecture

Generic Components



System Design Toolbox

- Many components, algorithms and architectures that contribute to a good design
- A component may be great with respect to one dimension at the cost of another dimension
  - Ex: Component may be extremely efficient but not as reliable
  - Ex: Component may provide amazing read access but not efficient write access (Cache)

Dimensions of System Design

How to design systems in the best way possible?

Dimensions

- Scalability
- Reliability
- Availability
- Efficiency
- Maintainability

Scalability

- System is scalable if it can handle additional load and operate efficiently
- The following components help facilitate additional load on the system
  - Load Balancer
  - Caches
  - Choice of database (SQL or NoSQL)
- There are multiple types of scaling:
  - Manual Scaling
  - Dynamic Scaling
  - Predictive Scaling
  - Scheduled Scaling
  - Warm Pools

Reliability

- System is reliable if it can always perform as expected
- System should tolerate user mistakes and should not crash
- System should prevent unauthorized access or abuse

Availability

- System is available if it is able to provide the service
- Availability is measure as (uptime/total time)
- Reliability and availability are related but not the same
- Reliability implies availability
- Availability does not imply reliability

Efficiency

- System can provide the functionality quickly
- Efficiency is measured in terms of
- Latency
- Response Time
- Bandwidth
- Efficiency can be improved by the following:
- Caching allows for greater efficiency for read requests by the user
- Load balancer distributes load more evenly across servers thus improving efficiency

Maintainability

- System is easy to operate and modify
- Simple for new engineers to understand
- Easy to modify for unanticipated use cases
- Modular APIs

Performance vs Scalability

- Performance of a system is an aggregated latency for rendering the product
- Performance can be bad due to the following reasons:
- Badly written code
- Low performing machines
- Too many network calls
- Slow DB queries
- Too much data passing over the network
- Performance is a qualitative metric
- Scalability is measured by assessing the ease of adding or removing nodes in a system
- Scalability helps us handle additional load without impacting the performance

Latency vs Throughput

- Latency is the time taken for data to traverse from one point to another
- Example: Latency of sending data from India to America will be higher than sending data from Europe to America.
- Throughput is the number of requests a system can complete in a given amount of time.
- Throughput of a system can be increased by adding more nodes even if the latency of the system is high
- Throughput can be increased by reducing the latency of the system

Design Problem

Design Parking Garage