QuickStart

Over 35 years in Technology Education

Principles of Software Engineering II: System Design

Course Outline

- Module 1: System Design Introduction
- Module 2: System Design Elements
- Module 3: System Design Details
- Module 4: Case Study

QuickStart
Over 35 years in Technology Education

Module 1
System Design Introduction

Content

- What is System Design? Why is it so Important?
- What to do as an Architect?
- Where is System Design? What is in System Design?
- Distributed Systems, Distributed Features
- Requirements, Requirements Details
- Design steps
- Data Flow
- High-level design, Detailed design
- Bottlenecks
- CAP theorem
- Redundancy

Definition and Importance of System Design?

What is System Design?

- The process of defining system elements includes modules/components, interfaces, and data for a system based on a specific set of requirements.
- The process of defining, developing, and designing systems has to satisfy a specific set of stakeholders' requirements.
- „System design could be seen as the application of system theory to product development.” [Wikipedia]

Why is it so Important?

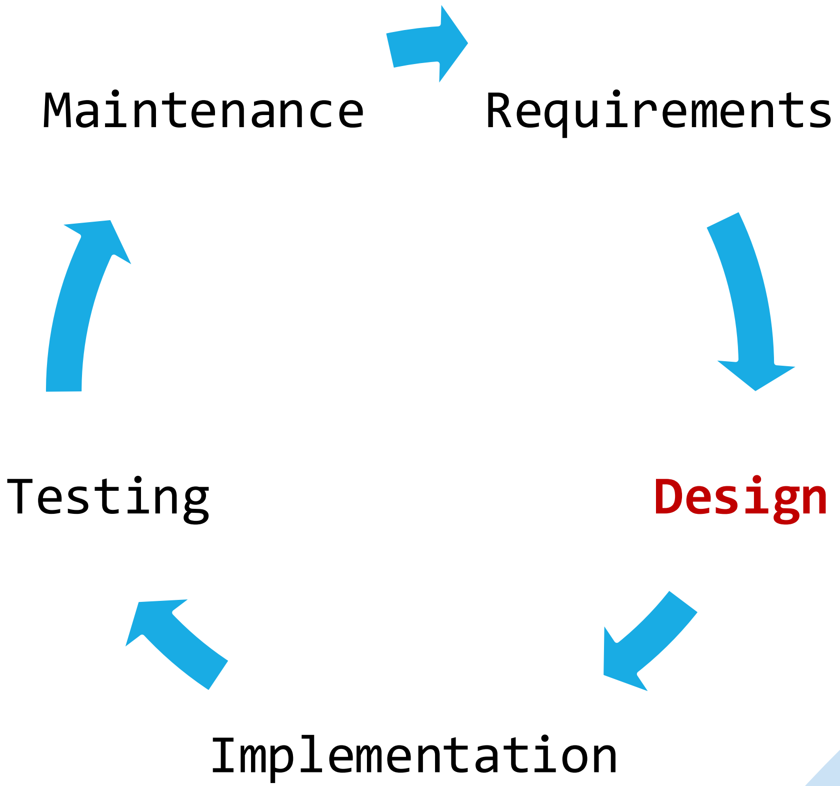
- Advancements in large-scale web applications
- Distributed systems
- Scalable systems
- Handle large amounts of traffic and data

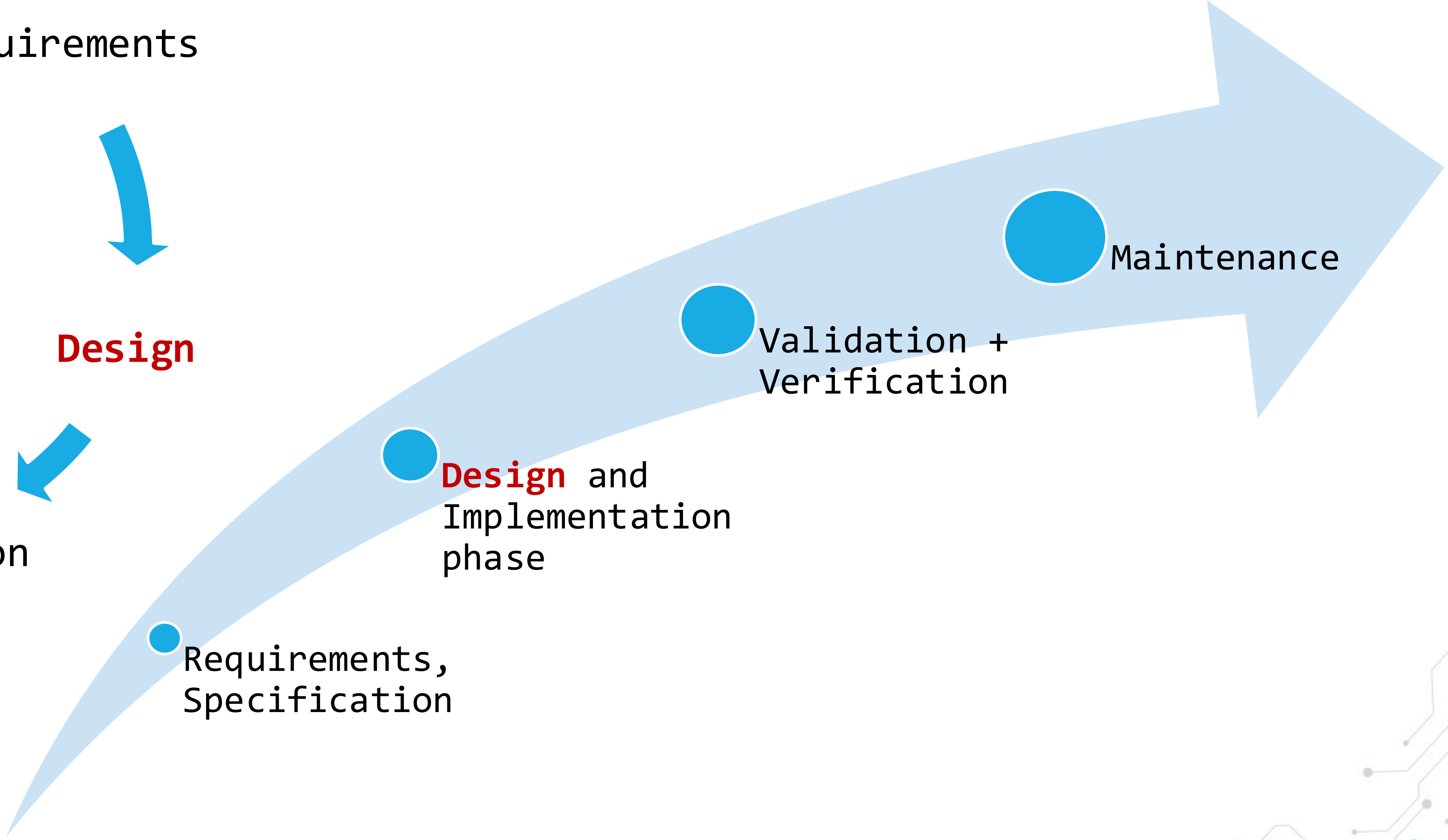
https://wiki.beparanoid.de/wiki/Systems_design?lang=en

What to do as an Architect?

1. Understand system design concepts
2. Apply design principles
3. Interviewing for higher-level positions

Where is System Design?





What is in System Design?

1. Architectural design: views, models, behavior, infrastructure
2. Logical design: data flow and inputs/outputs
3. Physical design: add information, represents information, store data

Distributed Systems

1. What is it?
    a. A group of computers working together
    b. All the complexity should be hidden from the user
    c. User interacting with a simple computer (however not)

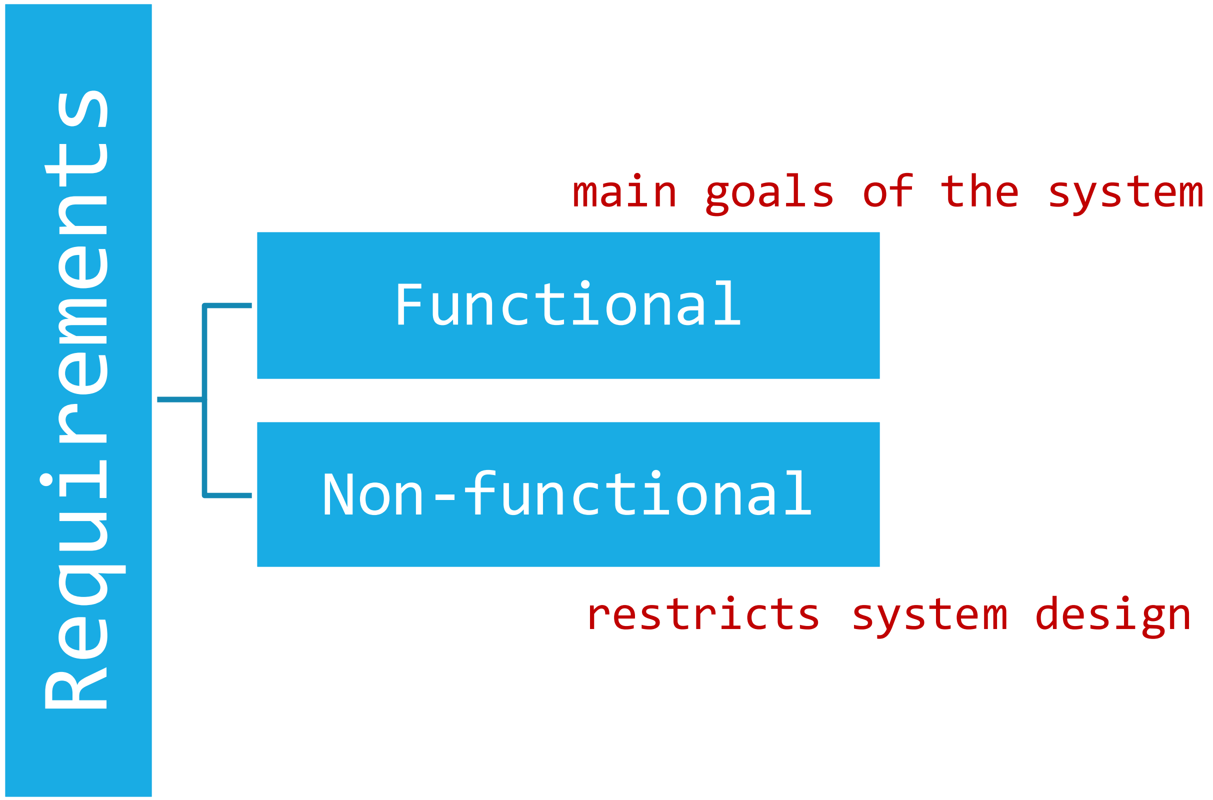
1. Characteristics:
    b. Scaling, Modularity
    c. Fault tolerance
    d. Low latency, Parallelism
    e. Effective, Efficiency

Distributed Features

1. Shared resources
2. Different memory and system clocks
3. Communication between clients and servers
4. Scalability

Requirements

1. Clarifying your goal
2. Focus on the main features



Requirements Details

1. Scalability
2. Security
3. Performance
4. Consistency
5. Reliability (e.g. MTBF: Mean Time Between Failure)
6. Availability (e.g. A%: Availability percentage)

Design steps

1. Requirements
2. Estimation
3. Data Flow
4. High-level design
5. Detailed design
6. Bottlenecks

Data Flow

Data flows between the different components

a. Relational Databases
b. NoSQL Databases
c. Graph Databases

High-level design

1. Splitting into major high-level components
2. Identify the main components and their connections

Detailed design

1. Analyze system components
2. Design detailed specification of components: e.g. speed, bandwidth, storage limits

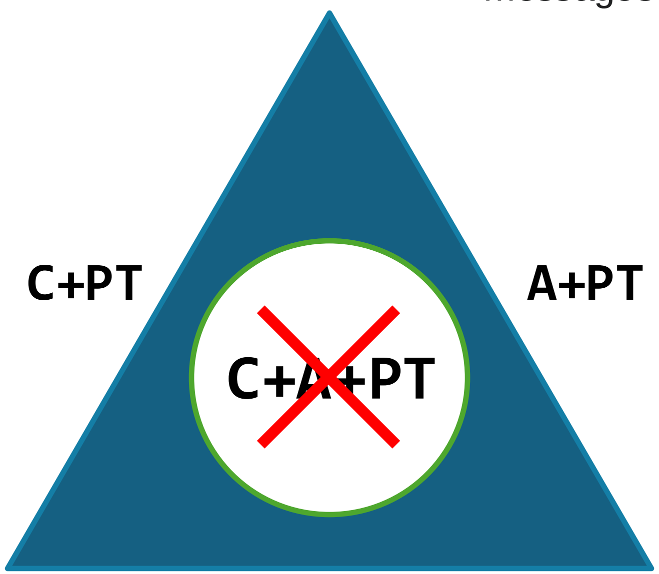
Bottlenecks

Identifying and mitigating/solve bottlenecks in the system: e.g. traffic, storage or consistency

CAP theorem

A distributed system can provide just two of the three properties simultaneously

Partition tolerance The system continues to operate despite an arbitrary number of messages being dropped



Every read receives the most recent write **Consistency**

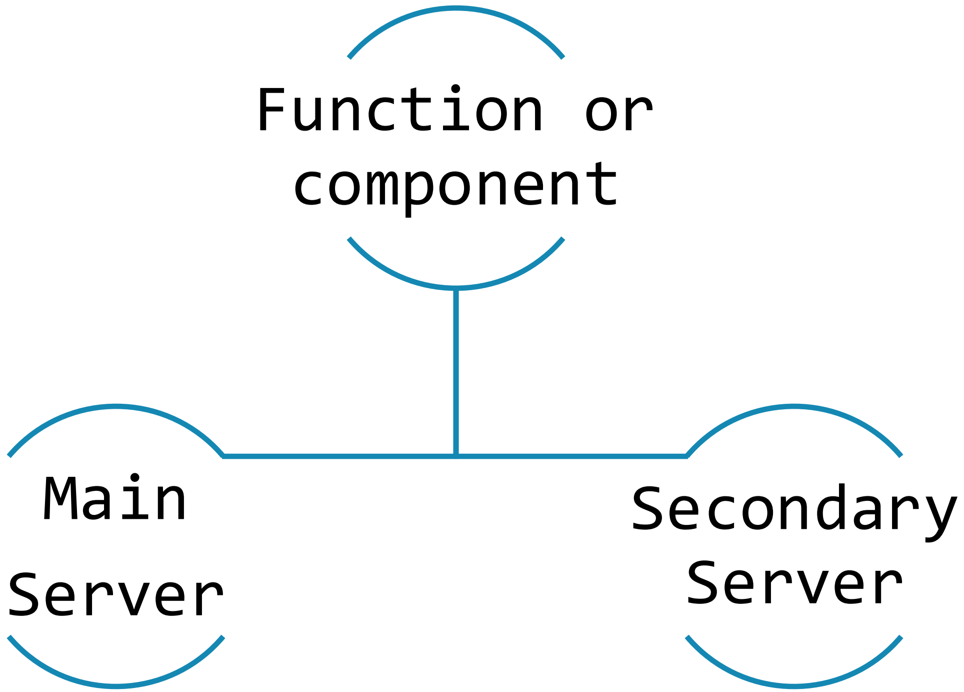
C+A

Availability Every request receives a (non-error) response

https://en.wikipedia.org/wiki/CAP_theorem

Redundancy

1. Duplicate critical components of a system
2. Ensure information consistency between redundant resources



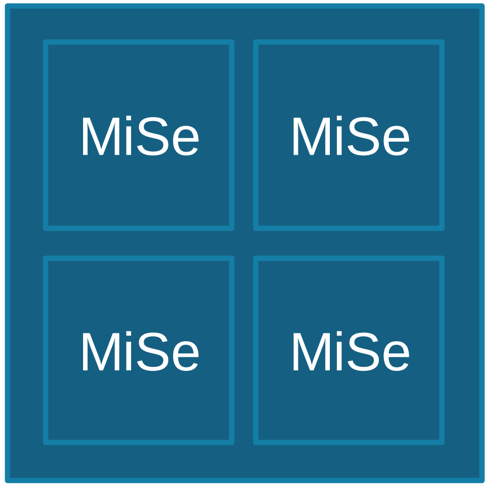
Module 2
System Design Elements

Content

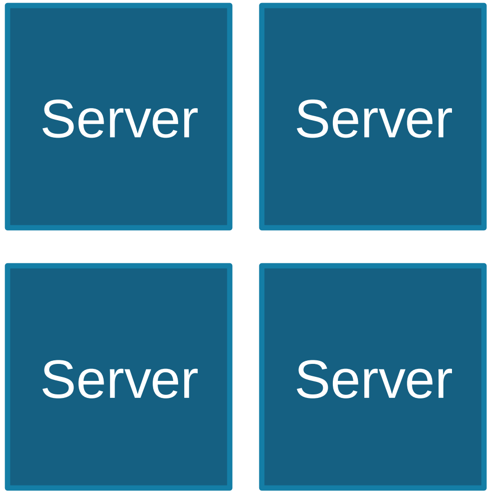
- How to Deal with Requirements?
- Scalability, Database Scaling
- Storage, Databases
- Distributed System Design Pattern
- Load Balancer, OSI Modell
- Layer-4 vs. -7 Load Balancer, Algorithms
- REST (Representational State Transfer), Service Properties
- Caching, Layers, Logistics, Strategies, CDN
- Containerization
- Cloud Technology
- Read Replicas
- Sharding
- Stateless and Stateful Systems
- Message Queues

How to Deal with Requirements?

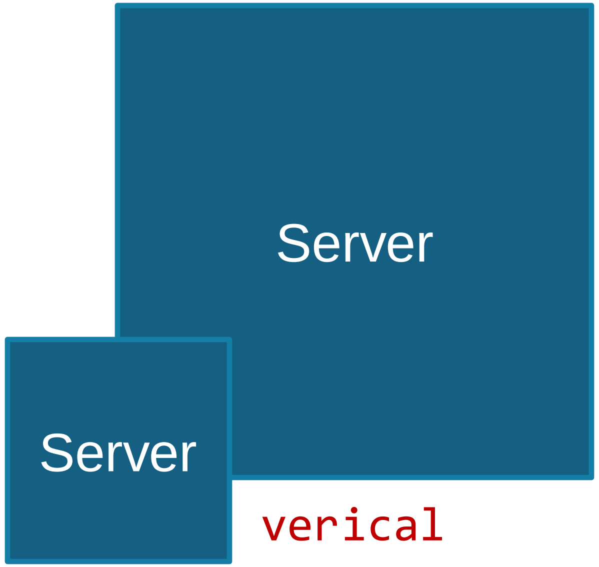
1. Monolithic applications
2. Horizontal Scaling
3. Vertical Scaling
4. Microservices



microservices

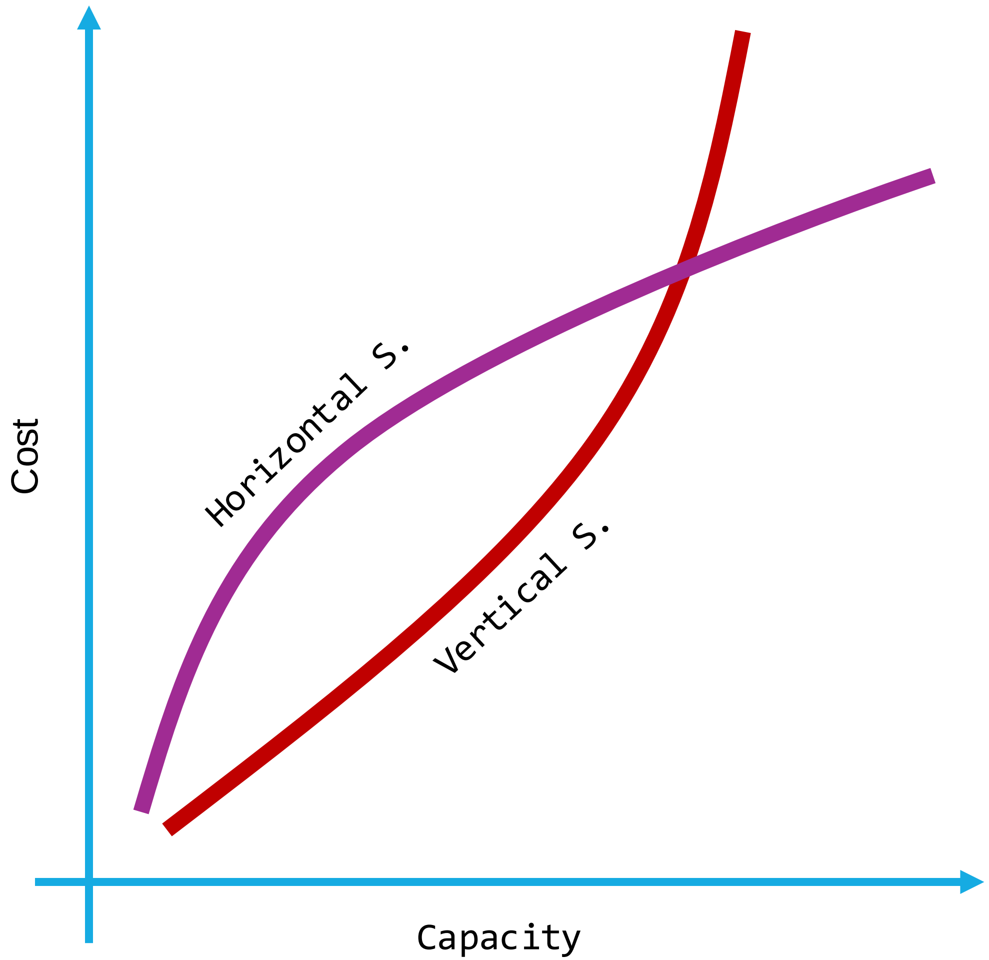


horizontal



verical

Scalability

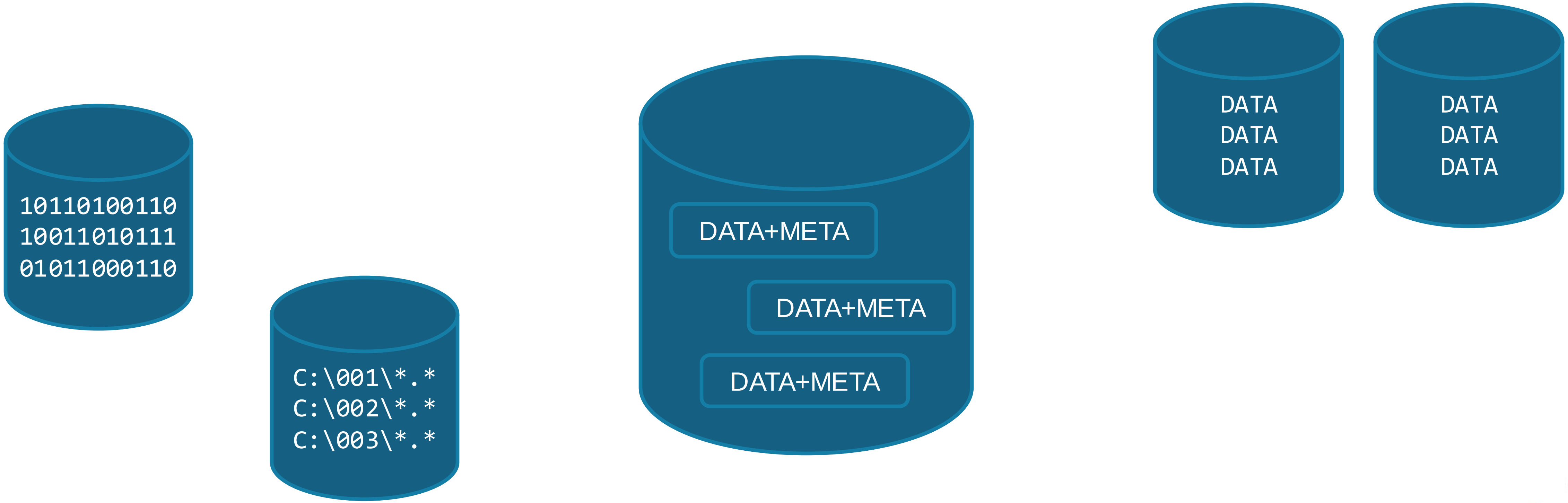


Database Scaling

1. Reading 95%, writing 5% (in general)
2. Vertical scaling
3. Horizontal scaling
4. Indexing (frequently or recently used)
5. Denormalization
6. Connection pooling

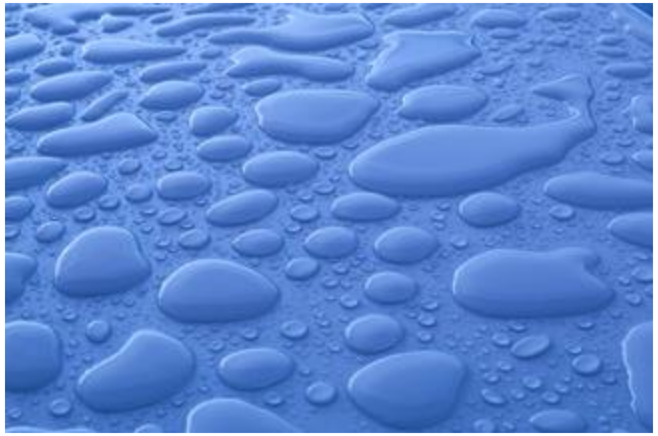
Storage

1. Block storage: fixed-size blocks in specific locations
2. File storage: folders in a fixed order
3. Object storage: scalable storage with metadata
4. RAID storage: data mirroring



BLOB Storage

1. A binary large object (BLOB): a collection of binary data stored as a single entity
2. BLOBs are typically images, audio or other multimedia objects, sometimes binary executable codes
3. Three types of resources:
   a. Storage account e.g. user account
   b. Container in the storage account e.g.images or videos
   c. Blob in a container e.g. image or video files
4. Types of BLOBs:
   a. Append BLOBs
   b. Block BLOBs
   c. Page BLOBs



https://en.wikipedia.org/wiki/Binary_large_object

Databases

1. Relational databases (data in tables): e.g. MySQL, PostgreSQL, Maria DB, SQLite, Oracle Database
2. Non-relational databases (data in any form): e.g. Document DB like MongoDB, Columnal DB like Cassandra, Graph DB, Key-Value DB

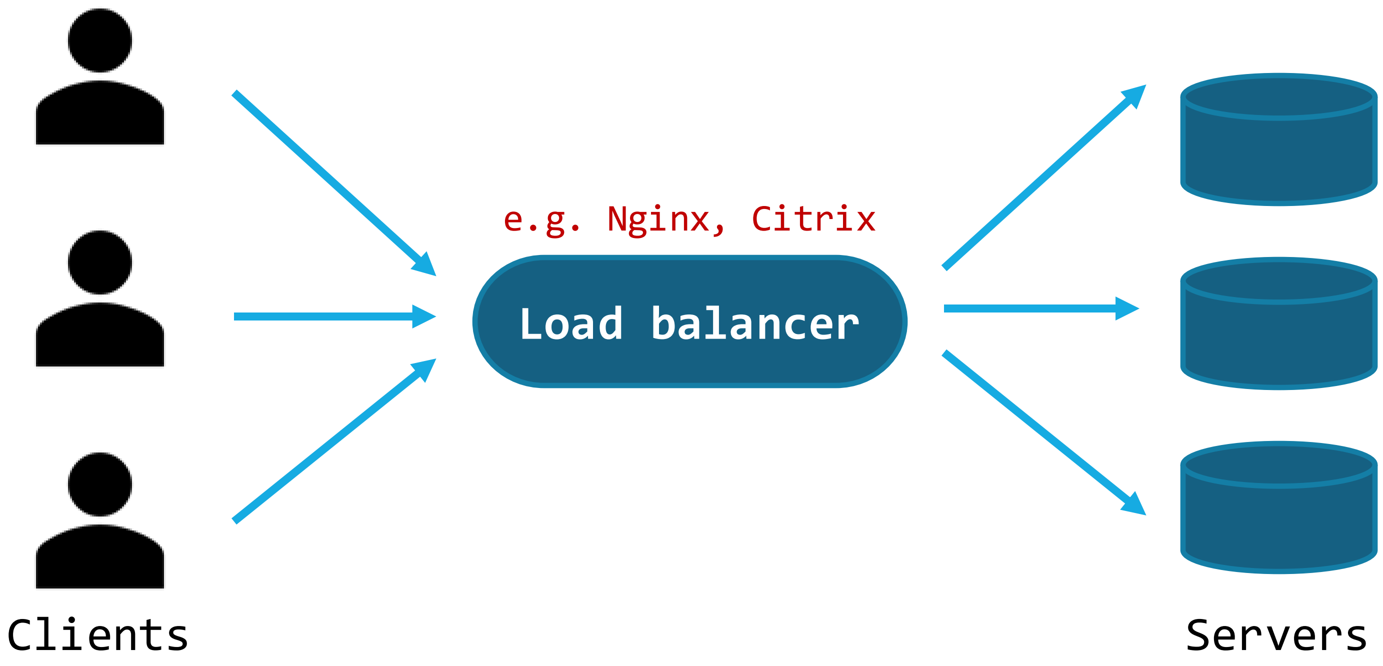
Distributed System Design Pattern

1. Types:
   a. Object communication
   b. Security
   c. Event-driven

1. Top patterns:
   b. Command and Query Responsibility Segregation (CQRS)
   c. Two-Phase Commit (2PC)
   d. Saga
   e. Replicated Load-Balanced Services (RLBS)
   f. Sharded Services

Load Balancer

The process of distributing a set of tasks over a set of resources (computing units), with the aim of making their overall processing more efficient.



https://en.wikipedia.org/wiki/Load_balancing_(computing)

OSI Model

The Open Systems Interconnection model (OSI model) is a conceptual model that provides a common basis for the coordination of ISO standards development for the purpose of systems interconnection.

	Layer		Protocol data unit	Function
Host layers	7	Application	Data	High-level protocols such as for resource sharing or remote file access, e.g. HTTP.
	6	Presentation		Translation of data between a networking service and an application; including character encoding, data compression and encryption/decryption
	5	Session		Managing communication sessions, i.e., continuous exchange of information in the form of multiple back-and-forth transmissions between two nodes
	4	Transport	Segment, Datagram	Reliable transmission of data segments between points on a network, including segmentation, acknowledgement and multiplexing
Media layers	3	Network	Packet	Structuring and managing a multi-node network, including addressing, routing and traffic control
	2	Data link	Frame	Transmission of data frames between two nodes connected by a physical layer
	1	Physical	Bit, Symbol	Transmission and reception of raw bit streams over a physical medium

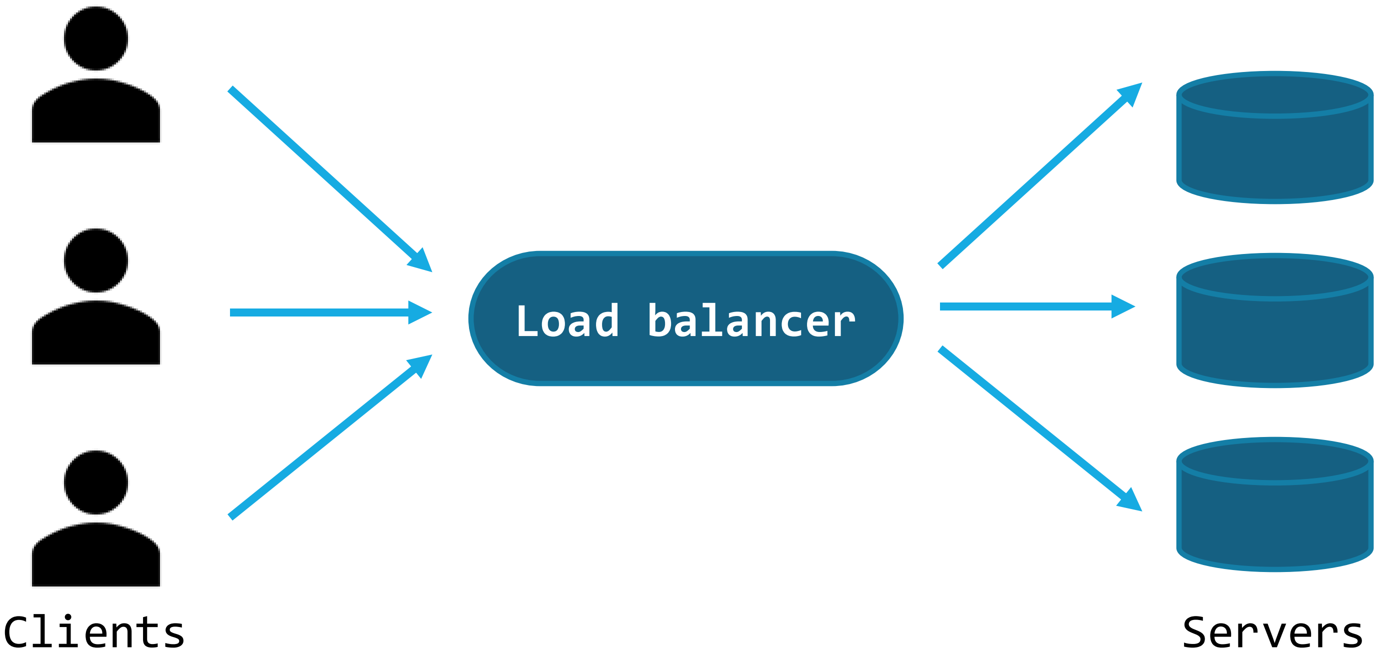
https://en.wikipedia.org/wiki/OSI_model

Layer-4 vs. -7 Load Balancer

1. Layer 4 (Transport Layer) LB
   a. TCP/UDP
   b. packet-level balancing
   c. quick and efficient
2. Layer 7 (Application Layer) LB
   a. Based on the URL
   b. Smart routing
   c. Caching
   d. Expensive
   e. Decryption is required

Load Balancer Algorithms

1. Round robin
2. Least resources
3. Least connections
4. IP hash



REST (Representational State Transfer)

A software architectural style that describes a uniform interface between physically separate components, often across the Internet in a server - client architecture.

https://en.wikipedia.org/wiki/Representational_state_transfer

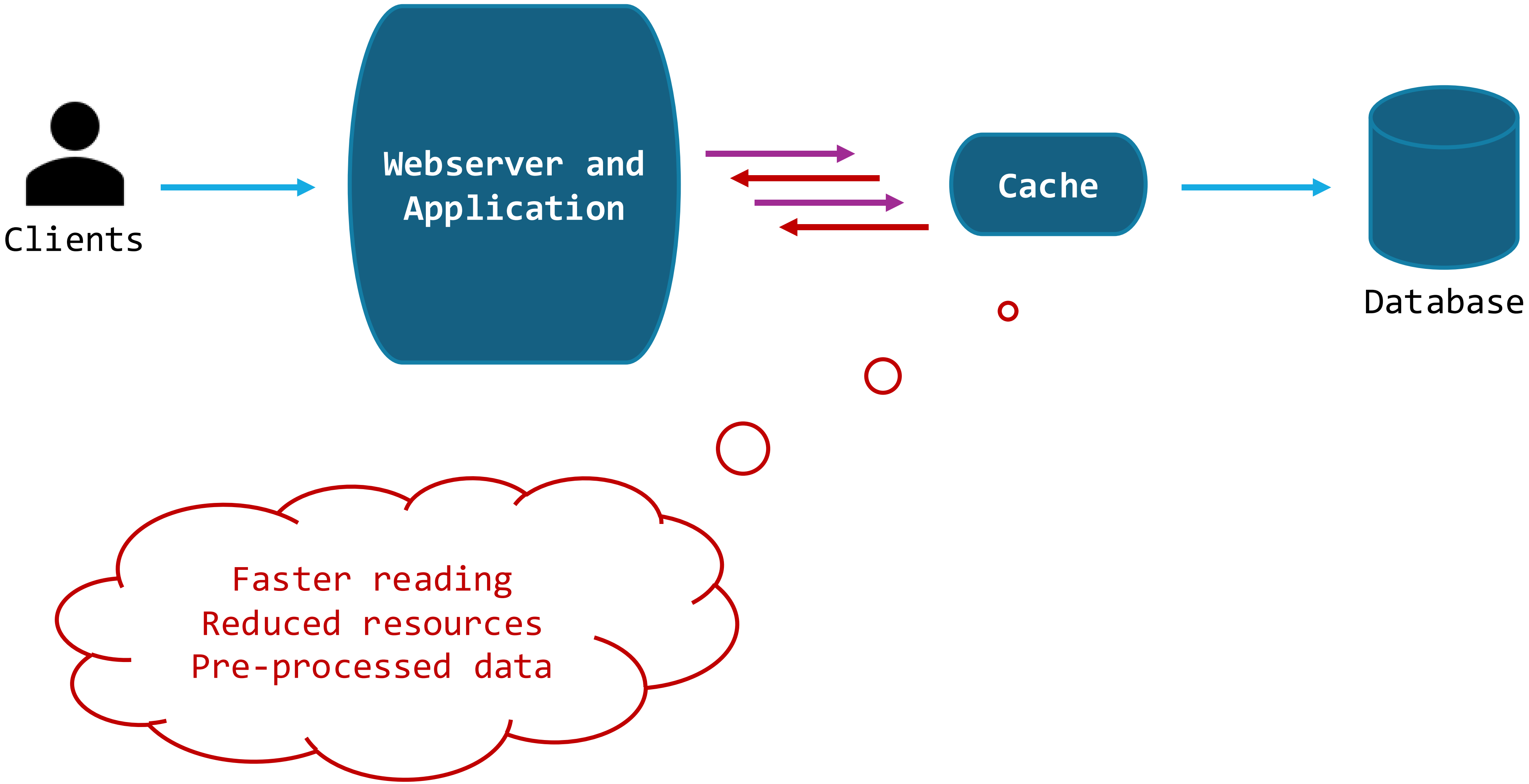
REST Service Properties

1. Visible communication between components
2. Resistance to failure at the system level
3. Scalable (huge numbers of components)
4. Uniform interface
5. Components modification
6. Portability of components portability
7. High-quality component interactions

Caching

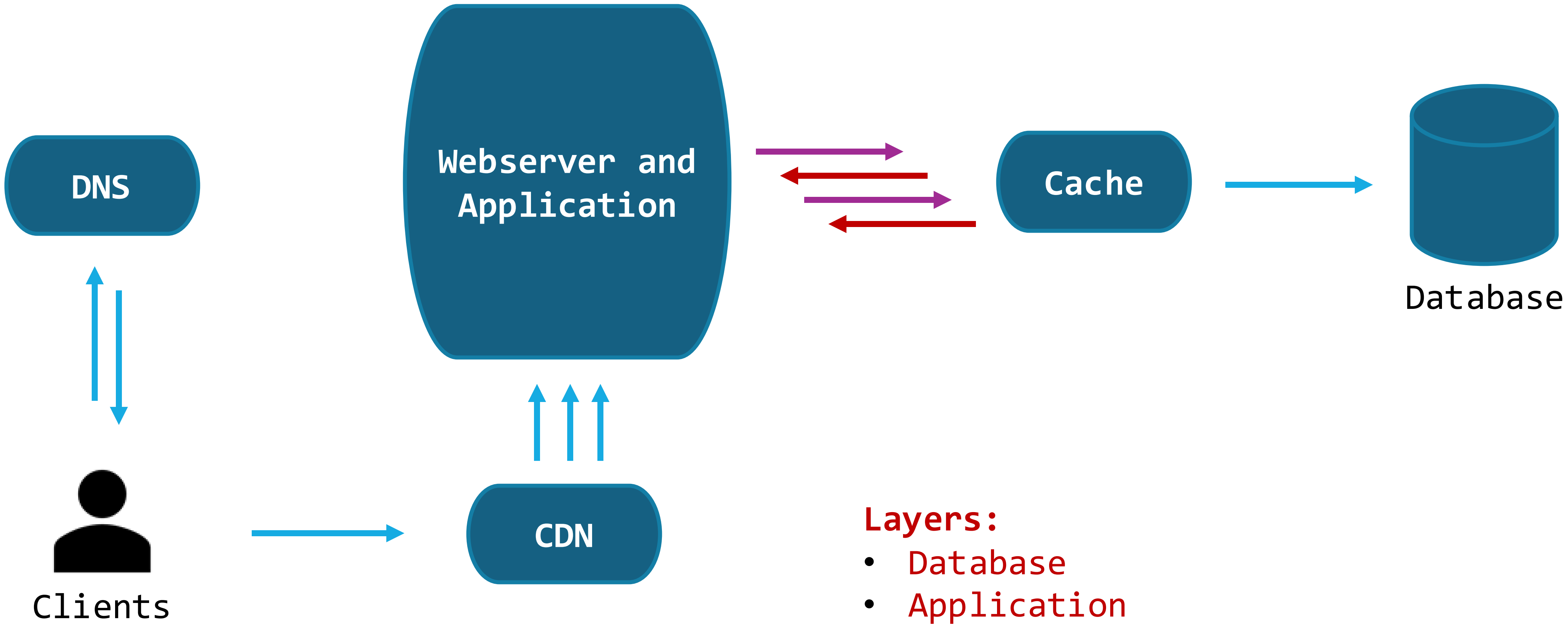
A cache is a hardware/software component. Stores data so the requests for the data can be served quicker. The data in a cache will be the result of an earlier computation or a copy of data stored elsewhere.

Caching



Clients
Webserver and Application
Cache
Database
Faster reading
Reduced resources
Pre-processed data

Caching Layers



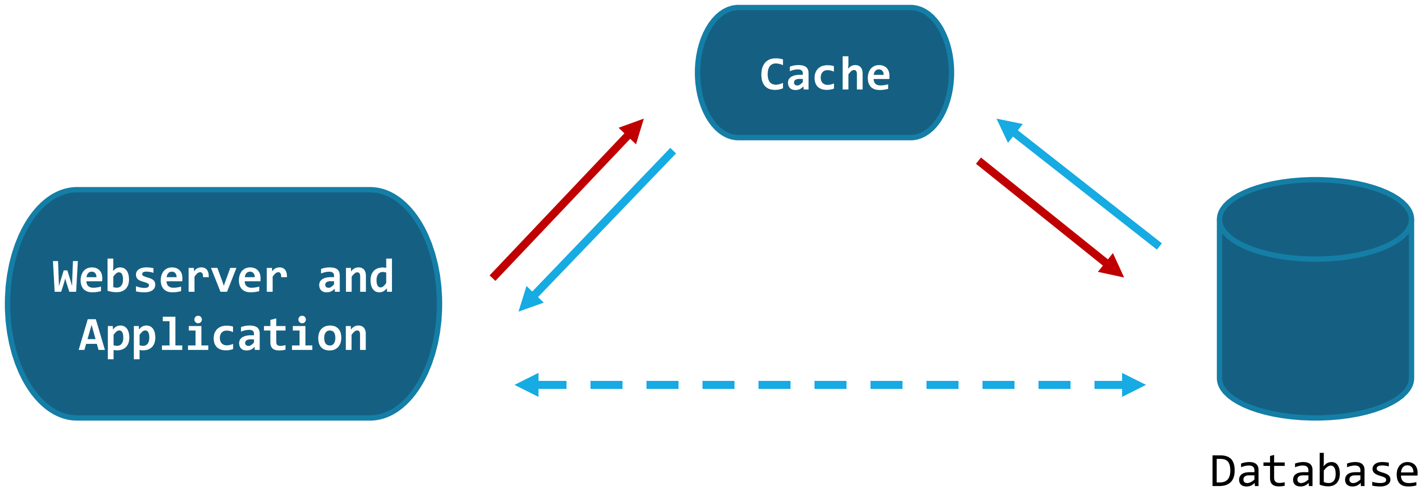
Layers:
- Database
- Application
- CDN
- DNS

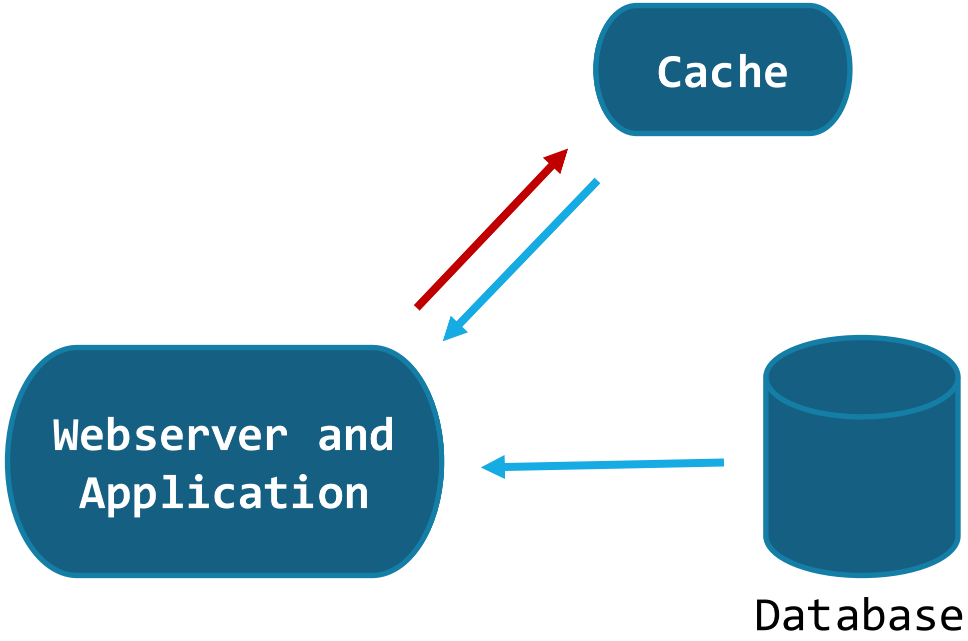
Caching Logistics

1. Least frequently used elements (LFU)
2. Least recently used elements (LRU)

Caching Strategies

1. Cache aside
2. Read through
3. Write through
4. Write back
5. Write around





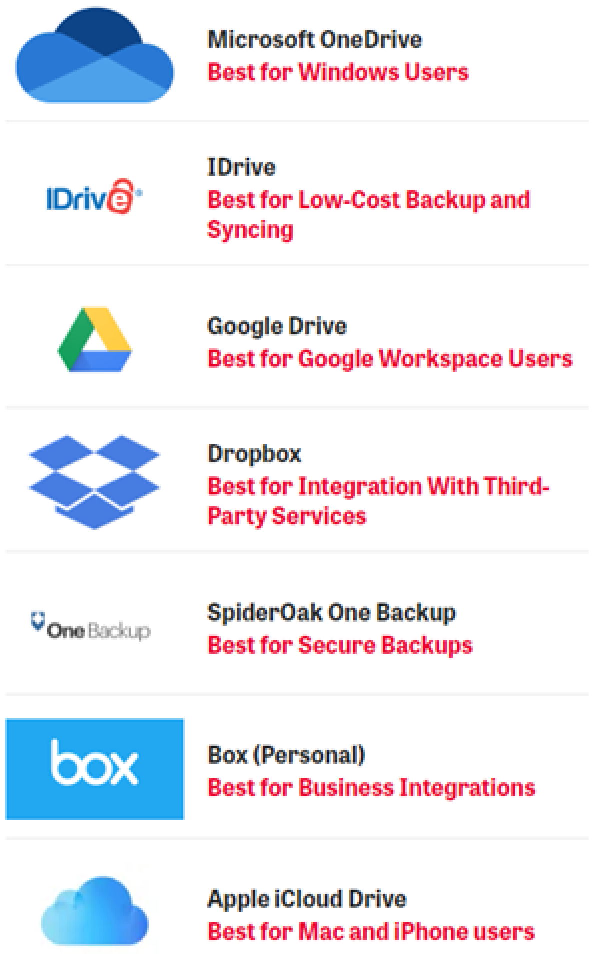
Containerization

Packaging of software components with dependencies, create a container or package that is platform independent.

https://www.docker.com/

https://kubernetes.io/

Cloud Technology



Cloud storage is a model of computer data storage in which the digital data is stored in logical pools, said to be on "the cloud".

Cloud computing is an on-demand availability of system resources, mainly data storage and computing capacity, without direct management by the user.

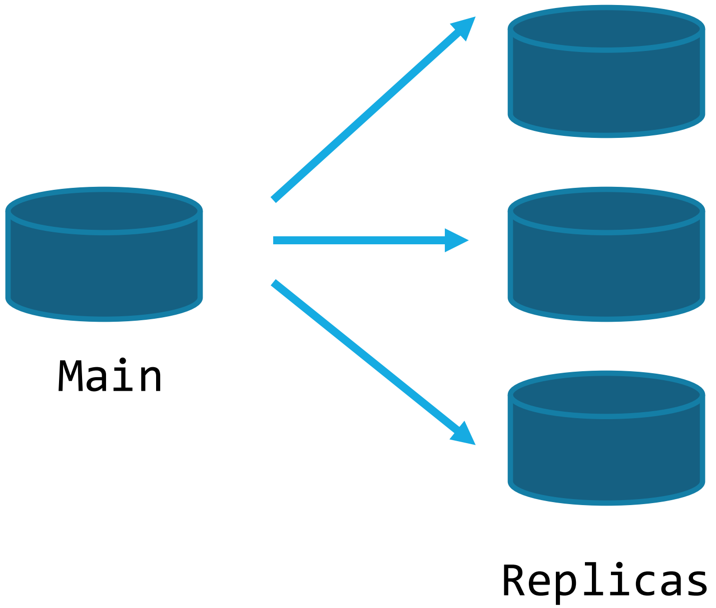
https://www.pcmag.com/picks/the-best-cloud-storage-and-file-sharing-services
https://en.wikipedia.org/wiki/Cloud_storage

Cloud vs Local Server Storage

1. Cloud
   a. Automatic maintenance and updates
   b. Storage space scaling
   c. Remoted storing of data
   d. Only Internet needed
   e. Internet needed
   f. Cloud-local data transfer
2. Local Server
   a. Upload and download speed
   b. Overall server system control
   c. Cyber security
   d. Hardware costs
   e. Manual maintenance and updates

Read Replicas

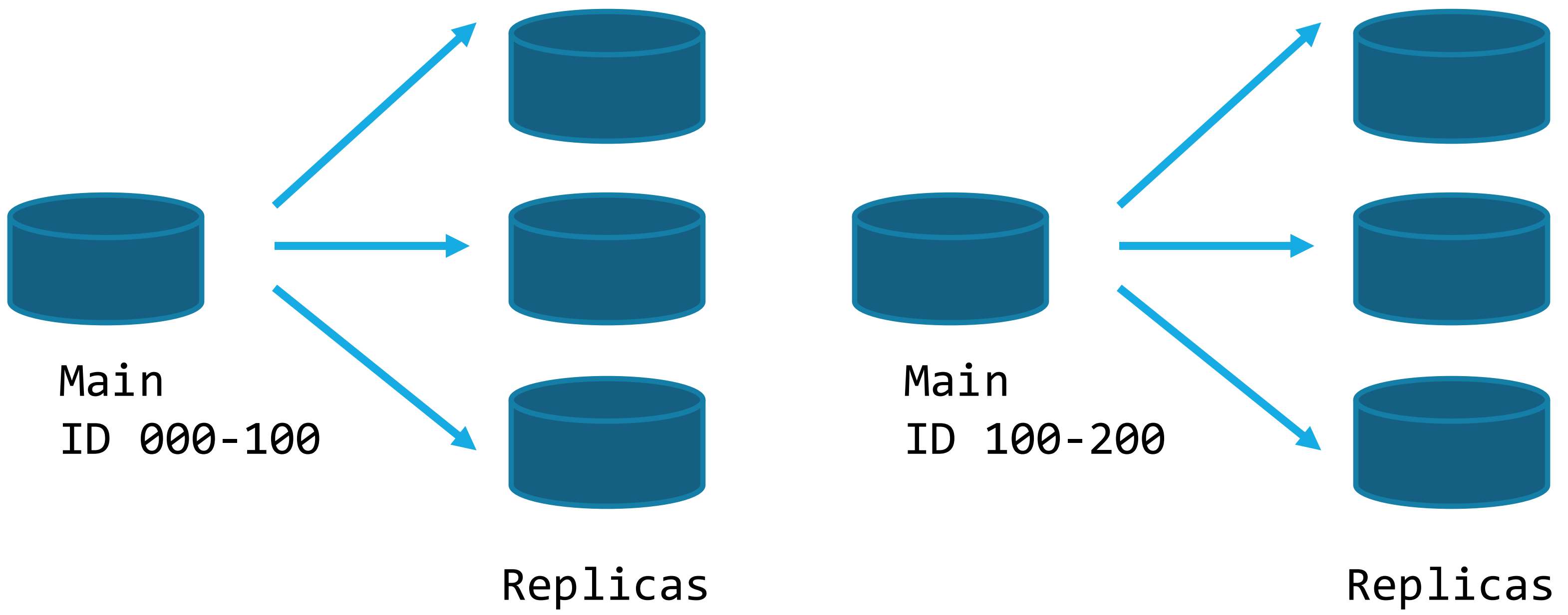
Replication in computing involves sharing information so as to ensure consistency between redundant resources, such as software or hardware components, to improve reliability, fault-tolerance, or availability.



https://en.wikipedia.org/wiki/Replication_(computing)

Sharding

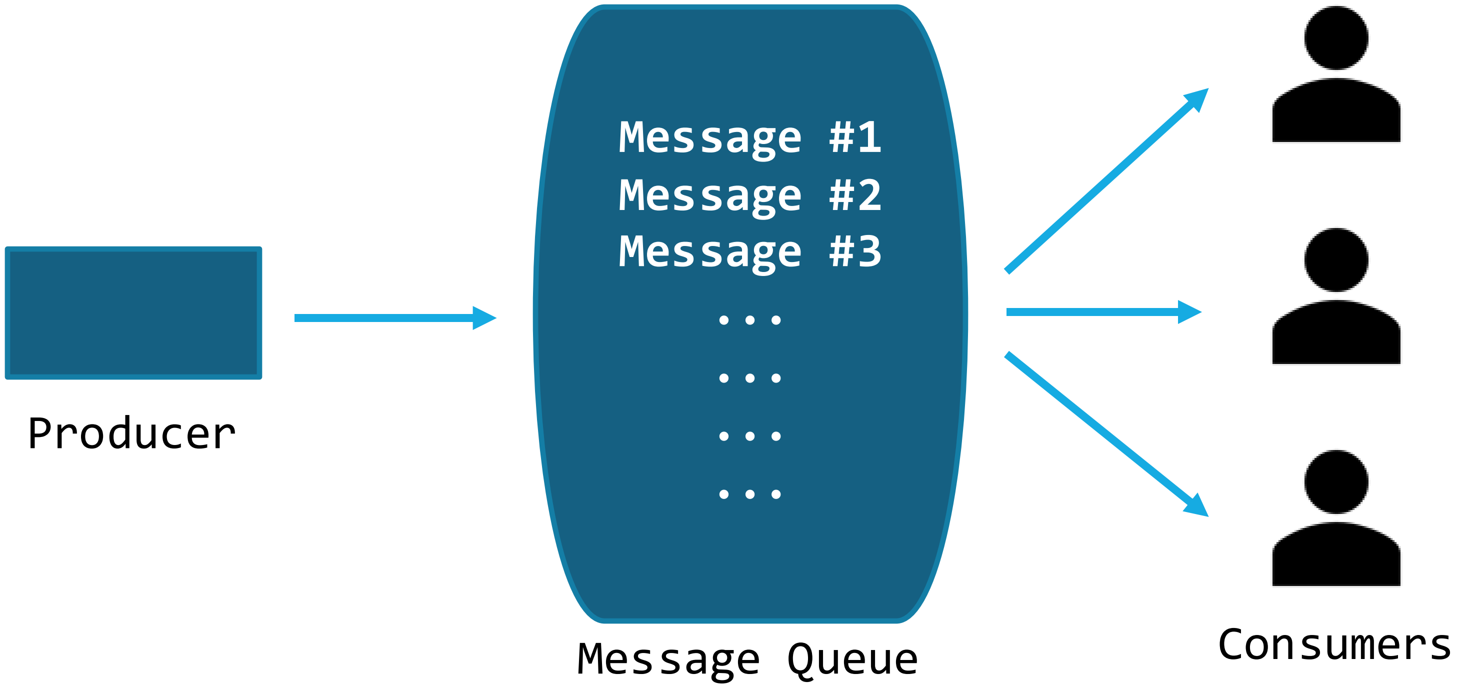
A database shard, or simply a shard, is a horizontal partition of data in a database or search engine. Each shard is held on a separate database server instance, to spread load.



https://en.wikipedia.org/wiki/Shard_(database_architecture)

Message Queues

Message queues are software components used for inter-process communication, or for inter-thread communication within the same process. A form of asynchronous service-to-service communication.



https://en.wikipedia.org/wiki/Message_queue

Module 3
System Design Details

Content

- Estimation
- Latency
- Conversions and Data Types
- Traffic estimation
- Memory, Bandwidth & Storage
- MTBF: Mean Time Between Failure
- Availability
- Example: LMS web application for a bootcamp

Estimation

Scale of the system:

a. e.g. Storage: e.g. MB or GB / day
b. e.g. Bandwidth: e.g. KB or MB / sec

Latency

Latency, from a general point of view, is a time delay between the cause and the effect of some physical change in the system being observed.

e.g.

- CPU cycle ~ 0.3 ns
- CPU L1 cache ~ 1 ns
- CPU L2 cache ~ 2 ns
- CPU L3 cache ~ 14 ns
- RAM ~ 100 ns
- SSD ~ 0.150 ms
- HDD ~ 10-20 ms
- WAN (~5000 km) ~ 50-200 ms

Fast

Slow

https://en.wikipedia.org/wiki/Latency_(engineering)

Latency

Latency, from a general point of view, is a time delay between the cause and the effect of some physical change in the system being observed.

e.g.

- CPU cycle ~ 0.3 ns
- CPU L1 cache ~ 1 ns
- CPU L2 cache ~ 2 ns
- CPU L3 cache ~ 14 ns
- RAM ~ 100 ns
- SSD ~ 0.150 ms
- HDD ~ 10-20 ms
- WAN (~5000 km) ~ 50-200 ms

Horizontal scaling

Fast

Slow

https://en.wikipedia.org/wiki/Latency_(engineering)

Conversions and Data Types

1. Conversions
    a. 8 bit = 1 byte
    b. 1024 bytes = 1 Kilobyte
    c. 1024 Kilobytes = 1 Megabyte
    d. 1024 Megabytes = 1 Gigabyte
    e. 1024 Gigabytes = 1 Terrabyte

2. Data types
    a. Integer: 2-8 bytes
    b. Char: 1 byte
    c. Float: 4-16 bytes

Traffic estimation

Necessary data:

a. Average daily active users (DAU)
b. Average action per user (AAU)

e.g.

DAU = 400 daily user
AAU = 55 request per day

Daily data request = DAU x AAU

Daily data request = 400 x 55
Daily data request = 22.000 req./day
...

Traffic estimation

e.g.

Seconds in a day = 24 x 60 x 60
Seconds in a day = 86.400 sec

Data request per second =
22.000 / 86.400 = 0,2546 req./sec

Memory, Bandwidth & Storage

e.g.

Required memory = Request per day x Average request size x 20%
Required memory = 22.000 x 250 bytes x 0.2 **=~ 1 Mbyte**

Req. bandwidth = (Request per day x Average request size) / 86.400
Req. bandwidth = (22.000 x 250 bytes) / 86.400 =~ 64 byte/sec
Req. bandwidth = 64 byte/sec **= 0.5 Kbit/sec**

Req. Storage = Request per day x
Average request size x Time
Req. Storage = 22.000 x 250 bytes x 1 year
Req. Storage = 5.25 Gbyte/year

MTBF: Mean Time Between Failure

$$MTBF = \left(\frac{Total\ elapsed\ time\ - Total\ downtime}{Number\ of\ Failures}\right)$$

e.g.

MTBF = (72 hours – 2 hours) / 2 failures
MTBF = 35 hours / failure

Availability

$$A\% = \left(\frac{Available\ time}{Total\ time}\right) \times 100$$

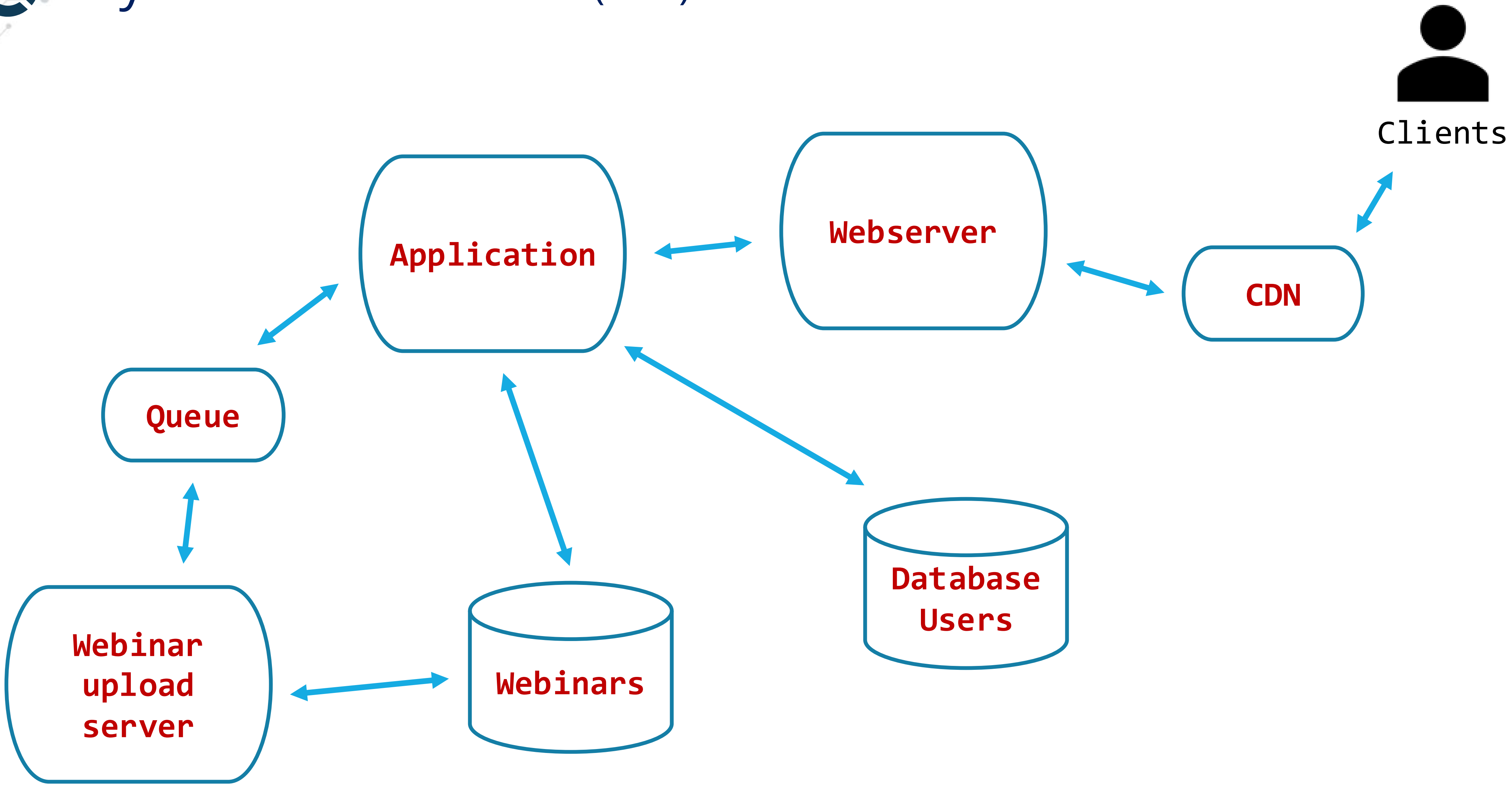
e.g. A% = (70 hours / 72 hours) x 100
A% = 97.22 %

A% = ((365 x 24 - 1 hours) /
((365 x 24 hours)) x 100
A% = 99.9885… %
A% = ((365 x 24 – 0.05 hours) /
((365 x 24 hours)) x 100
A% = 99.999… %

A Simple Example

1. **Design an LMS web application for a bootcamp**
2. Requirements
    a. View / download videos
    b. View / download documents
    c. Upload videos
    d. Upload documents
    e. Search for materials / videos
    f. Upload messages on lessons

System Architecture (LMS)



Database (LMS)

```
Questions
---
Id
Subject
Message
User
```

```
Webinars
---
Id
Title
Description
Thumbnail
Questions
AdminUser
```

```
Users
---
Id
Name
Role
Contact
```

Estimates (LMS)

1. Material uploaded: 10 hours of video per week
2. Material watched: 1500 hours of video watching per week
3. Average size of 1 hour long video: 500 MB

Storage: 500 x 10 / 7 ~= **714 MB/day**
Bandwidth: 500 x 1500 / 7 / 24 / 60 / 60 = 1.24 MB/s ~= **9,92 Mbps**

Module 4
Case Study

Content

- Goal
- Users and Scale
- Requirements
- Traffic estimates
- Storage estimates
- High-level System Design
- Database Design

Goal

1. The reason of the design?
   a. Design a Bootcamp Chat Application

1. What is the main concept?
   b. A tool for communication and mentoring in a standard IT bootcamp setting between students and mentors

Users and Scale

1. Who are the users? Is there any special user segment?
    a. Anyone who wants to learn a given topic
    b. Teachers/mentors, students (generally youngers)

1. How many users use the system initially?
    b. About 10,000 users

2. Scaling?
    a. 10,000 -> 1,000,000 users

Requirements

1. Functional requirements
    a. One-to-one messaging
    b. Group messaging
    c. Channels
    d. User status display
    e. File upload and download

2. Non-functional requirements
    a. Low latency during messaging (instant reactions) - max. 1 second
    b. High availability (using at any time) - max. 1 day, 99.7% availability
    c. Scalability (more and more users) - 10.000 → 1.000.000

Traffic Estimates

1. Traffic estimates
   a. Total number of users: 1,000,000
   b. Daily active users (DAU): 25%
   c. DAU: 250,000 users per day
   d. Areal distribution (timezones):
      i. 65% Europe
      ii. 25% North America
      iii. 10% Other continents
   e. Typical server capacity (in 2022): **250,000 req./s**
   f. Minimum number of servers: **1**

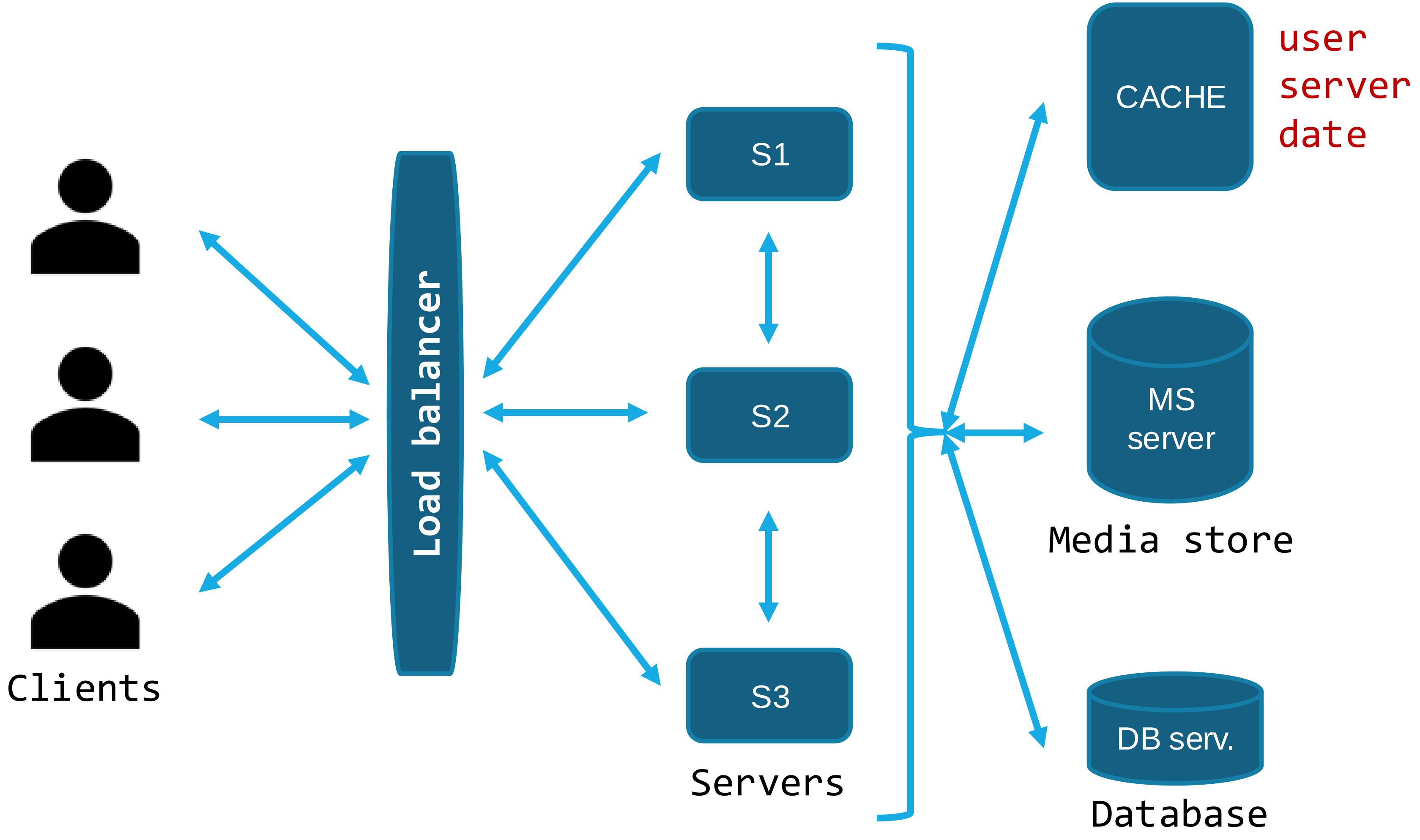
https://developerpitstop.com/how-many-concurrent-requests-can-a-web-server-handle/

Storage Estimates

1. Storage estimates for text messages (for 10 years)
    a. Average character size: 2 bytes
    b. Average message size: 160 characters
    c. Average number of messages per day: 15 messages
    d. Requests per day: 162,500
    e. Total required storage: 2 x 160 x 15 x 162,500 =~ 750 MB/day
    f. Total required storage for 10 years: 750 x 365 x 10 =~ **2.6 TB/10y**

2. Storage estimates for images (for 10 years)
    a. Average image size: 500 KB
    b. Average number of images per day: 3 images
    c. Maximum requests per day: 162,500
    d. Total required storage: 500 x 3 x 162,500 =~ 230 GB/day
    e. Total required storage for 10 years: 230 x 365 x 10 =~ **820 TB/10y**

High-level System Design



Database Design