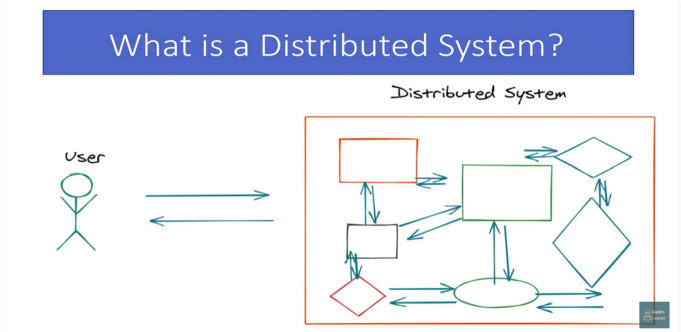
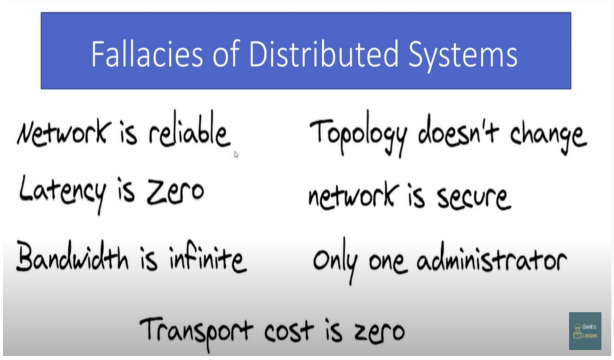
System Design

Distributed Design

- Keep it simple





- Characteristics of distributed Systems:
  - No shared clock
  - No shared memory
  - Shared resources
  - Concurrency and consistency

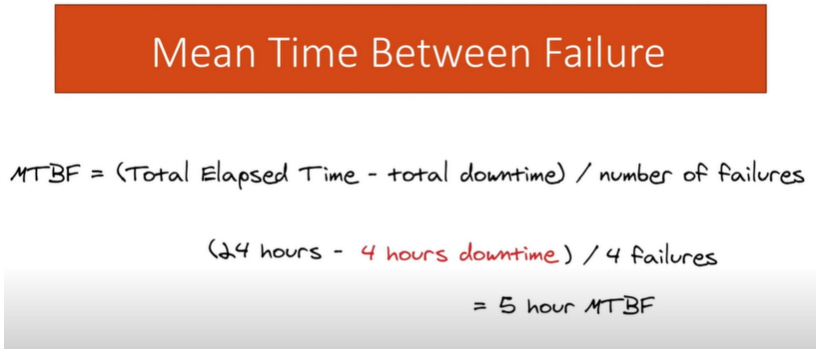
- Different parts of distributed system need to be able to talk.
- Requires agreed upon format or protocol.

- Things that can go wrong and must be handled:
  - Client can't find server
  - Server crash mid request
  - Server response is lost
  - Client crash

- Benefits :
  - More reliable, fault tolerant
  - Scalability
  - Lower latency, increased performance
  - Cost effective

➢ System Design Performance Metrics

- Scalability
  - Ability of a system to grow and manage increased traffic
  - Increased volume of data or requests
- Reliability
  - Probability of a system will fail during a period of time.
  - Slightly harder to define than hardware reliability



MTBF = (Total elapsed time - total downtime)/ number of failures
= (24-4)/4 failures = 5 hour MTBF

- Availability
  - Amount of time a system is operational during a period of time
  - Poorly designed software requiring downtime for updates is less available.
- Availability % : = (available time/total time)* 100
  - =(23 hours/24 hours) * 100 = 95.83%
- 

- Reliability v/s availability
  - Reliable is always available
  - Available may be maintained by redundancy but system may not be reliable.
  - Reliable software will be more profitable because providing same services requires less backup resources.
  - Requirements will depend on function of the software.
- Efficiency :
  - How well a system performs
  - Latency and throughput often used as metrics.
- Manageability
  - Speed and difficulty involved with maintaining system.
  - Observability, how hard to track bugs
  - Difficulty of deploying updates

- Want to abstract away infrastructure so product engineers don't have to worry about it.

Latency Numbers

Access Type	Time	Converted Time
CPU Cycle	.3 ns	1 S
CPU L1 Cache	1 ns	3 S
CPU L2 Cache	3 ns	9 S
CPU L3 Cache	13 ns	43 S
Main Memory(RAM)	120 ns	6 minutes
SSD	150 micro seconds	6 days
HDD	10 ms	12 months
SF to NYC	40 ms	4 years
SF to Australia	183 ms	19 years

- Latency Key Takeaways
  - Avoid network calls whenever possible.
  - Replicate data across data centres for disaster recovery as well as performance.
  - Use CDNs to reduce latency.
  - Keep frequently accessed data in memory if possible rather than seeking from disk, caching.
- 86400 second x 30 days = 2,500,000 seconds = 2.5 Million seconds approx. per month
- UNIX timestamp = 4 Bytes

➢ **Instagram rough estimate**

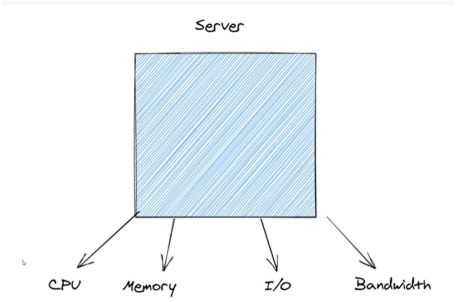
- Traffic Estimates
  - Average daily active users(DAU) X average reads/writes per user
    - 10 million DAU x 30 photos viewed = 300 Million Photo requests
    - 10 million DAU x 1 photo upload = 10 million Photo writes

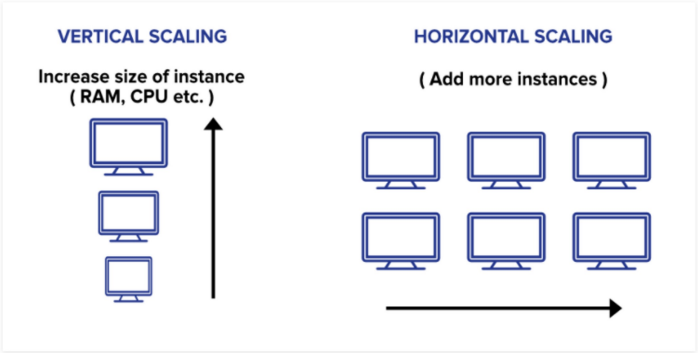
    - 300 Million Requests//86400 = 3472 Requests per second
    - 10 Million writes //86400 = 115 writes per second
- Memory requests:
  - Read Requests per day x Av. Request size x 0.2
    - 0.2 is because of 80-20 rule
    - The 80-20 rule, also known as the Pareto Principle, is an aphorism which asserts that 80 % of the outcomes result from 20 % of inputs/causes.
  - 300 Million Request x 500 Bytes = 150 GB
  - 150 GB x 0.2 x 3(replication) = 90 GB

- Bandwidth :
  - Requests per day x Request size
  - 300 Million Requests x (1MB[photo] + 0.5 MB [text/data]) = 450,000 GB
  - 450,000 GB // 86400 sec. = 5.2 GB per second

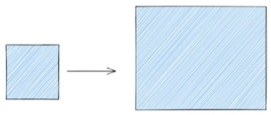
- Storage :
  - Writes per day x Size of write x time to store data
  - 10 Million writes x 1.5 MB = 15 TB per day
  - 15 TB x 365 days x 10 years = 55 Petabytes

➢ <u>Horizontal V/S Vertical Scaling</u>



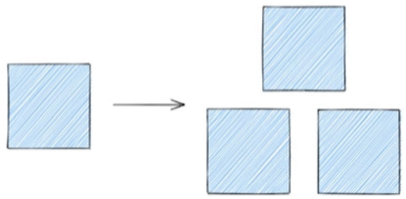


➢ Vertical Scaling



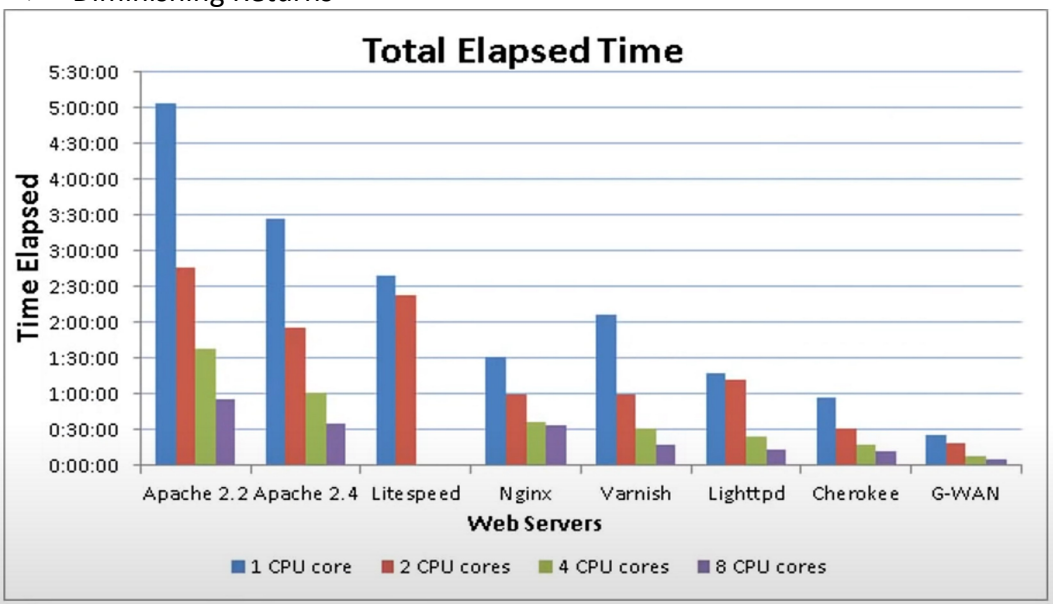
  - Upgrading the capacity of single machine
  - Moving to a new machine with more power/capacity.
  - Increase can be in:
    - Better Processor
    - Increased RAM
    - Other increasing adjustments
  - Referred to as '**scaled-up**' approach.
  - Doesn't require partitioning of data
  - Easy implementation.
  - Application compatibility is maintained.
  - Used in small/mid sized companies.
  - Eg.: MySQL and Amazon RDS .

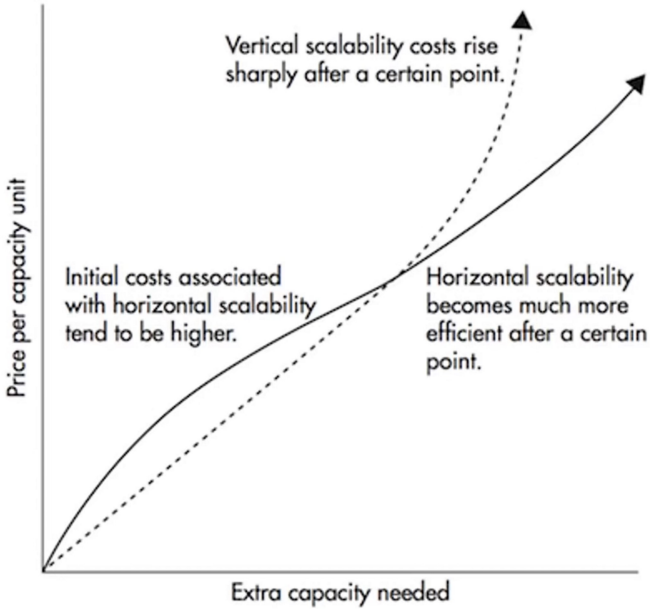
  - Drawbacks:
    - Limited Scaling.
    - Limited increase in performance.
    - Replacing needs downtime
    - Greater risk of hardware failures.
    - Expensive.

- Horizontal Scaling
  - 
  - Performance enhanced by
    - adding more machines to the network.
    - Sharing the processing and memory across multiple devices.
  - No replacing of previous server.
  - No downtime required.
  - Referred to as '**scaled-out**' approach.
  - Widely used because it includes
    - Increasing I/O concurrency
    - Reducing the load on existing nodes
    - Increasing disk capacity.
  - Can be achieved by
    - Distributed file system
    - Clustering
    - Load balancing
  - Traffic can be managed effectively.
  - Easy to upgrade
  - Cheaper
  - Eg.: Cassandra and MongoDB
  - More complexity upfront but more efficient long term

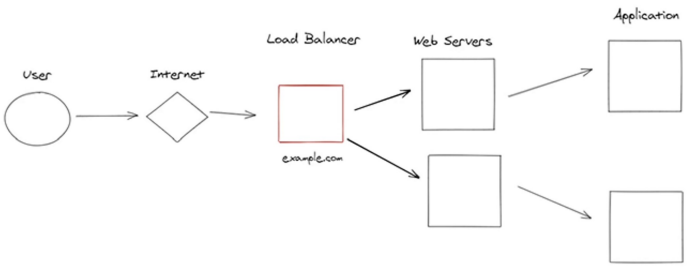
  - Drawbacks
    - Complicated architectural design
    - High licensing fees
    - High utility costs(cooling and electricity)
    - Extra networking equipment required.

- Diminishing Returns





- **<u>System Design Components</u>**

- Load Balancers
  - Balance incoming traffic to multiple servers.
  - Software or hardware based
  - Used to improve reliability and scalability of application
  - Software based : Nginx, HAProxy,
  - Hardware based: F5, Citrix
  - 

- Load Balancers Routing Methods

  - Round Robin
    - Simplest
    - LB has a list of servers available and it will cycle through them and send the next request.
    - May result in uneven traffic as some request can have heavy DB query whereas some may be just very light.

  - Least Connections
    - Routes based on the no. of client connections to server.
    - Useful for chat or streaming applications

  - Least Response time
    - Routes based on how quickly servers respond.
    - The servers with more connections/traffic will respond slower and vice-versa

  - IP Hash

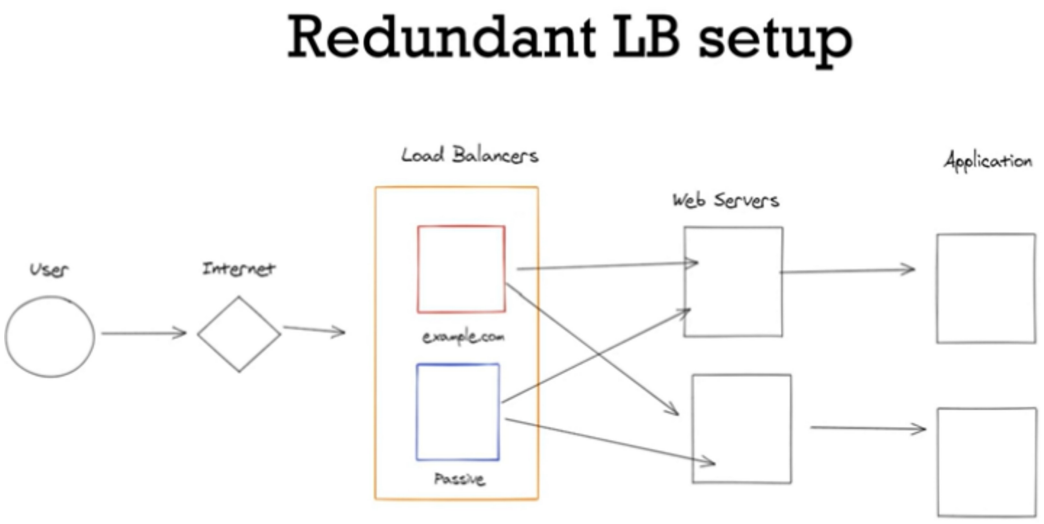
- Routes client to server based on IP
- Useful for stateful sessions.

- Types of Load Balancers

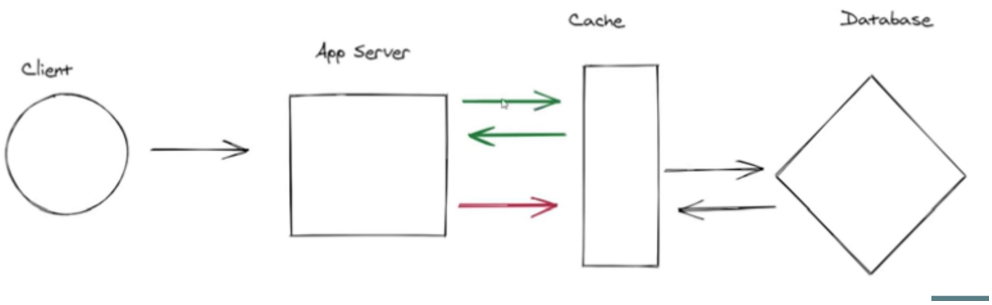
  L4 v/s L7 load balancers

  - Layer 4 load balancers
    - Only has access to TCP and UDP data.
    - Faster
    - Lack of information can lead to uneven traffic because it does not have access to full request itself.
    - Benefit of that is that it's good on the edge of your data centre or of your network because it can look at the IP address and for ex. If you are getting a DOS attack ,you can just toss that request at the edge.

  - Layer 7 load balancers
    - Has full access to HTTP protocol and data
    - SSL termination.
    - Check for authentication
    - Smarter routing options
    - More CPU intensive but not a big factor.

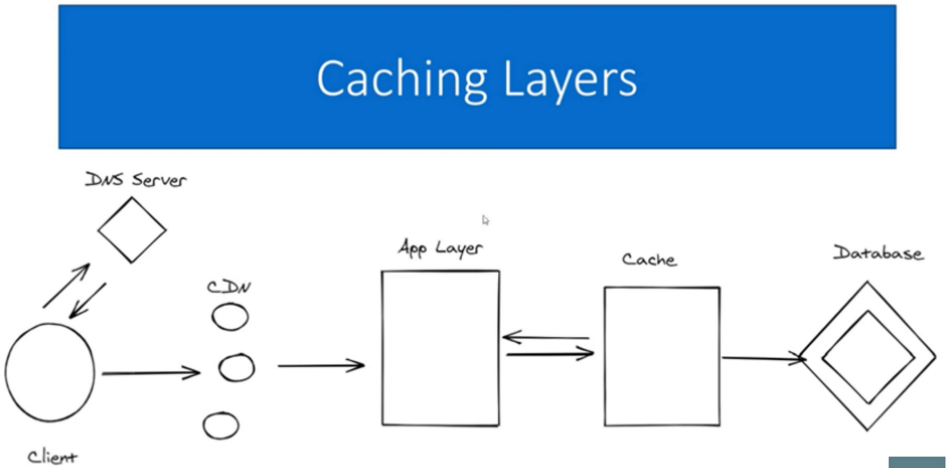


- Caching
  - Why?
    - Saves time
    - Saves Money



- Speed and performance

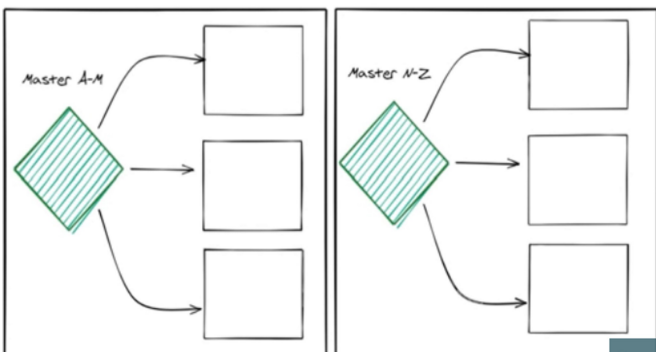
- Read/Write operation up to 50-200x faster
- Can serve same amount of traffic with fewer resources
- Pre-calculate data



- Time to Live (TTL)
  - Set a time period before a cache entry is deleted
  - Used to prevent stale data.

- Last Recently Used (LRU)
  - Once cache is full, remove last accessed key and add new key
- Last Frequently Used (LFU)
  - Track no. of times key is used
  - Drop least used when cache is full

- Caching Strategies
  - Cache Aside
  - Read Through
  - Write Through
  - Write Back

- Sharding
  - Horizontal Partitioning of DB
  - Schema remains same but split across multiple DBs
  - 

- Vertical Partitioning of DB
  - Schema is divided

➢ **Designing a You Tube Clone**

✓ Requirements
- Video upload
- View video upload
- Search for videos
- Track stats of videos
- Comments on videos

✓ Capacity estimates
- Storage
- Bandwidth

✓ Key data
- 500 hours of video uploaded per minute
- 1 Billion hours of video watched per day

○ Storage estimate
- 500 hours x 60 x 24 x 50MB //1000,000 = 2160 TB per day
- 2.16 PB per day

○ Bandwidth estimate
- 1 Billion hours x 3GB per hour // 1000 = 3,000,000 TB = 3,000 PB

-