# 1. Scale from zero to millions of users

## 1.1 Single server setup

-Single server setup where everything is running on one server: web app, database, cache…

A diagram of a web server

AI-generated content may be incorrect.

-Request flow:

A diagram of a web browser

AI-generated content may be incorrect.

A close-up of a document

AI-generated content may be incorrect.

-Traffic source:

A close-up of a text

AI-generated content may be incorrect.

## 1.2 Database

-Multiple servers: web/mobile traffic (web tier) and database (data tier) servers allows them to be scaled independently.

A diagram of a computer network

AI-generated content may be incorrect.

-Which database to use:

+Relational database (RDBMS or SQL database) represent and store data in tables and rows. You can perform join operations.

+Non-Relational databases (NoSQL database) are grouped into 4 categories: key-value stores, graph stores, column stores, document stores. Join operations are not supported.

+Use non-relational databases if:

A close up of black text

AI-generated content may be incorrect.

## 1.3 Vertical scaling vs horizontal scaling

-**Vertical scaling** (scale up): process of adding more power (CPU, RAM…)

-**Horizontal scaling** (scale-out): adding more servers into your pool of resources.

-When traffic is low, vertical scaling is great option but comes with serious limitations:

A white background with black text

AI-generated content may be incorrect.

## 1.4 Load balancer

-Load balancer distributes incoming traffic among web servers in a load-balanced set.

A diagram of a network

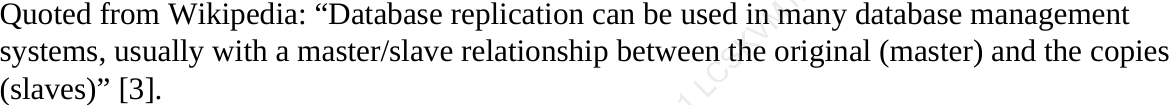
AI-generated content may be incorrect.

A close up of text

AI-generated content may be incorrect.

->solved no failover issue and improved availability of web tier.

## 1.5 Database replication



-A **master database** only supports write operations. A **slave database** gets copies of the data from master database and only supports read operations.

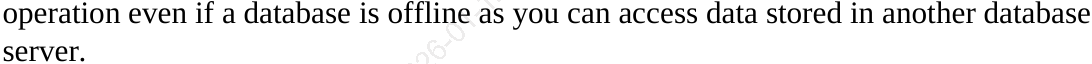
A diagram of a web server

AI-generated content may be incorrect.

-Advantages:

A close up of text

AI-generated content may be incorrect.



-What if one of databases goes offline:

A close-up of a text

AI-generated content may be incorrect.

-Add the load balancer and database replication

A diagram of a computer network

AI-generated content may be incorrect.

A close-up of a text

AI-generated content may be incorrect.

## 1.6 Cache

-**Cache**: a temporary storage area that stores the result of expensive responses or frequently accessed data in memory so that subsequent requests are served more quickly.

1.6.1 Cache tier

-**Cache tier**: a temporary data store layer, much faster than the database. It includes better system performance, ability to reduce database workloads and the ability to scale the cache tier independently.

A blue square with black text

AI-generated content may be incorrect.

-**read-through cache**: After receiving a request, a web server 1st checks if the cache has the available response. If it has, it sends data back to client. If not, it queries the database, stores the response in cache, and sends it back to client.

-Most cache servers provides APIs for common programming languages. Example: Memcached APIs

A close up of a text

AI-generated content may be incorrect.

1.6.2 Considerations for using cache

A close-up of a text

AI-generated content may be incorrect.

A close-up of a text

AI-generated content may be incorrect.

A diagram of a computer network

AI-generated content may be incorrect.

A close up of text

AI-generated content may be incorrect.

## 1.7 Content delivery network (CDN)

-**CDN**: a network of geographically dispersed servers used to deliver static content. CDN servers cache static contents (images, videos, CSS, JS files…)

-**Dynamic content caching**: caching HTML pages that are based on request path, query strings, cookies, request headers.

-CDN workflow:

A diagram of a cloud with arrows

AI-generated content may be incorrect.

A close-up of a page

AI-generated content may be incorrect.

A white background with black text

AI-generated content may be incorrect.

-Considerations of using a CDN

A close up of a text

AI-generated content may be incorrect.

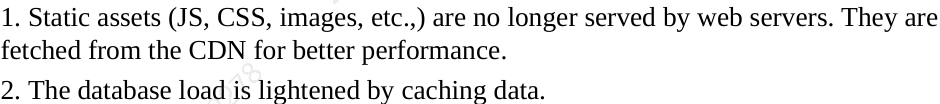
A white paper with black text

AI-generated content may be incorrect.

-The design after adding CDN and cache

A diagram of a computer network

AI-generated content may be incorrect.



## 1.8 Stateless web tier

1.8.1 Stateful architecture

-A stateful server remembers client data (state) from one request to the next.

A diagram of a computer server

AI-generated content may be incorrect.

-The issue is that every request from the same client must be routed to the same server. This can be done with sticky sessions in most load balancers, but this adds the overhead. Adding or removing servers is difficult. It’s also challenging to handle server failures.

1.8.2 Stateless architecture

A diagram of a server

AI-generated content may be incorrect.

-**Stateless architecture**: A stateless server keeps no state information. HTTP request from users can be sent to any web server, which fetch state data from a shared data store. State data is stored in a shared data store and kept out of web servers. A stateless system is simpler, more robust, and scalable.

-Design with a stateless web tier:

A diagram of a computer server

AI-generated content may be incorrect.

**+**We move the session data out of web tier and store them in persistent data store: relational database, Memcached/Redis, NoSQL… Autoscaling (ad or remove web servers automatically based on traffic load) is easily archived.

## 1.9 Data centers

-**geoDNS**: a DNS service that allows domain names to be resolved to IP addresses based on the location of a user.

A diagram of a computer network

AI-generated content may be incorrect.

-In the event of any significant data center outage, we direct all traffic to a healthy data center.

A diagram of a computer network

AI-generated content may be incorrect.

-Technical challenges must be resolved to achieve multi-data center setup:

A close-up of a text

AI-generated content may be incorrect.

## 1.10 Message Queue

-**Message queue**: a durable component, stored in memory, that supports asynchronous communication. It serves as a buffer and distributes asynchronous requests.

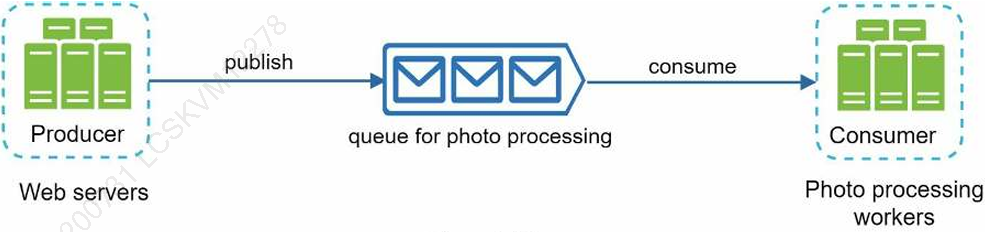
-The **basic architecture**: input services (producers/publishers) create messages, and publish them to a message queue. Other services or servers (consumers/subscribers) connect to the queue, and perform actions defined by messages.

A blue and white line of mail icons

AI-generated content may be incorrect.

+The producer can post a message to the queue when consumer is unavailable to process it. The consumer can read messages from the queue even when the producer is unavailable.

-Example: photo customization



## 1.11 Logging, metrics, automation

-**Logging**: monitoring error logs is important because it helps to identify errors and problems. You can monitor error logs at per server level or use tools to aggregate them to a centralized service for easy search and viewing.

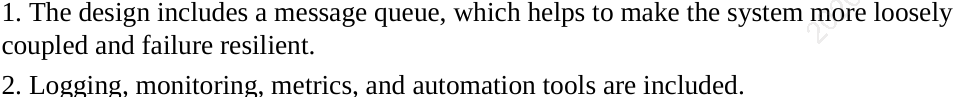
-**Metrics**: Collecting different types of metrics help us to gain business insights and understand the health status of the system. Some useful metrics:

A close up of black text

AI-generated content may be incorrect.

-**Automation**: build or leverage automation tools (build, test, deploy process) to improve productivity.

-Adding message queues and different tools:



A diagram of a computer network

AI-generated content may be incorrect.

## 1.12 Database scaling

-**Vertical scaling**:

+Add more power (CPU, RAM, DISK) to existing machine.

+Some drawbacks:

A close up of text

AI-generated content may be incorrect.

-**Horizontal scaling**:

+Add more server:

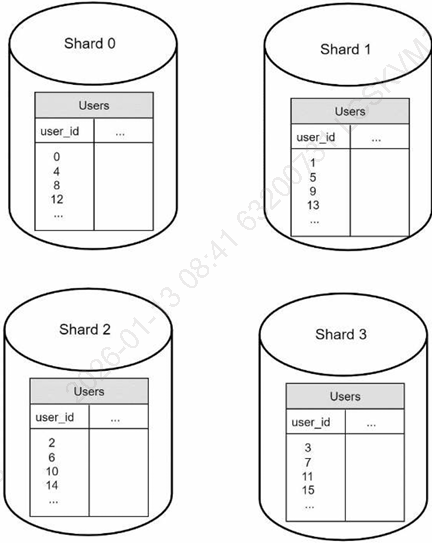
A diagram of a comparison between a horizontal scaling and a horizontal scaling

AI-generated content may be incorrect.

+Horizontal scaling (Sharding) separates databases into smaller **shards**. Each shard shares the same schema, though the actual data on each shard is unique to the shard.

+Example: When you access data, a hash function is used to find the corresponding shard. We use user\_id%4 as the hash function. If = 0, shard 0 is used to store and fetch data…

A diagram of a diagram of a user

AI-generated content may be incorrect.

+The most important thing to implement sharding is the choice of **sharding key** (partition key). It consists of one or more columns that determine how data is distributed (example: user\_id). It helps you retrieve and modify data efficiently by routing database queries to correct database.

+Challenges:

A close-up of a text

AI-generated content may be incorrect.

-Use shard databases to increase data traffic.

A diagram of a computer network

AI-generated content may be incorrect.

## 1.13 Millions of users and beyond

-More fine-tuning and new strategies: optimize your system, decouple system to even smaller services.

-Summary of how we scale system to support millions of users:

A white text with black text

AI-generated content may be incorrect.

## 1.14 Reference materials

-Hypertext Transfer Protocol: https://en.wikipedia.org/wiki/Hypertext\_Transfer\_Protocol

-Should you go Beyond Relational Databases?: <https://blog.teamtreehouse.com/should-you-go-beyond-relational-databases>

-Replication: https://en.wikipedia.org/wiki/Replication\_(computing) -Multi-master replication: <https://en.wikipedia.org/wiki/Multi-master_replication>

-NDB Cluster Replication: Multi-Master and Circular Replication: https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster-replication-multi-master.html

-Caching Strategies and How to Choose the Right One: https://codeahoy.com/2017/08/11/caching-strategies-and-how-to-choose-the-right-one/

-R. Nishtala, "Facebook, Scaling Memcache at," 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI ’13).

-Single point of failure: <https://en.wikipedia.org/wiki/Single_point_of_failure>

-Amazon CloudFront Dynamic Content Delivery: https://aws.amazon.com/cloudfront/dynamic-content/

-Configure Sticky Sessions for Your Classic Load Balancer: https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-sticky-sessions.html

-Active-Active for Multi-Regional Resiliency: https://netflixtechblog.com/active-active-for-multi-regional-resiliency-c47719f6685b -

-Amazon EC2 High Memory Instances: <https://aws.amazon.com/ec2/instance-types/high-memory/>

-What it takes to run Stack Overflow: http://nickcraver.com/blog/2013/11/22/what-it-takes-to-run-stack-overflow

-What The Heck Are You Actually Using NoSQL For: [http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql for.html](http://highscalability.com/blog/2010/12/6/what-the-heck-are-you-actually-using-nosql%20for.html)

# 2. Back-of-the-envelope Estimation

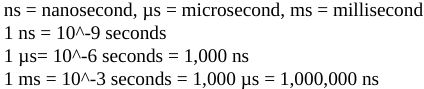
## 2.1 Power of 2

A table with numbers and words

AI-generated content may be incorrect.

## 2.2 Latency numbers every programmer should know

A white rectangular table with black text

AI-generated content may be incorrect.

-Conclusions:

A white background with black text

AI-generated content may be incorrect.

## 2.3 Availability numbers

-The ability of a system to be continuously operational for a long period of time. Most services falls between 99% and 100%.

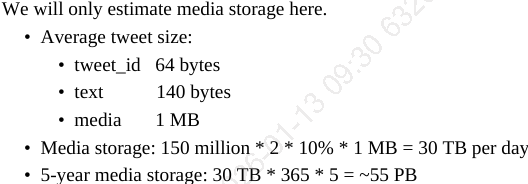
-**SLA** (service level agreement) is the agreement between service provider and customer to defines the level of uptime your service will deliver.

A white rectangular table with black text

AI-generated content may be incorrect.

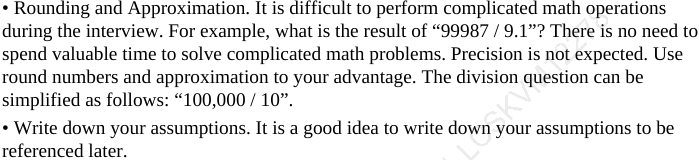
## 2.4 Example: Estimate Twitter QPS and storage requirements

A white paper with black text

AI-generated content may be incorrect.

## 2.5 Tips

-Solving the problem is more important than obtaining results. Interviewers may test problem-solving skills.



A white background with black text

AI-generated content may be incorrect.

## 2.6 Reference materials

-J. Dean.Google Pro Tip: Use Back-Of-The-Envelope-Calculations To Choose The Best Design: http://highscalability.com/blog/2011/1/26/google-pro-tip-use-back-of-the-envelope-calculations-to-choo.html

-System design primer: https://github.com/donnemartin/system-design-primer

-Latency Numbers Every Programmer Should Know: <https://colin-scott.github.io/personal_website/research/interactive_latency.html>

-Amazon Compute Service Level Agreement: https://aws.amazon.com/compute/sla/

-Compute Engine Service Level Agreement (SLA): <https://cloud.google.com/compute/sla>

-SLA summary for Azure services: https://azure.microsoft.com/en-us/support/legal/sla/summary/