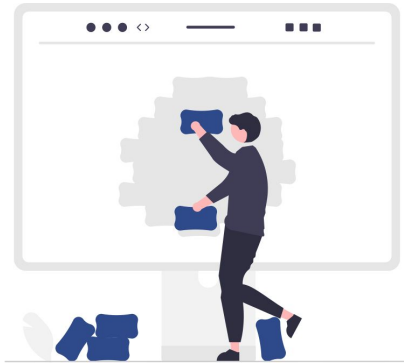
Episode 1

System Design for Noobs

July 2, 2022



Core topics covered in this slide:

1. DNS
2. CDN
3. Load Balancer

Some FAQ

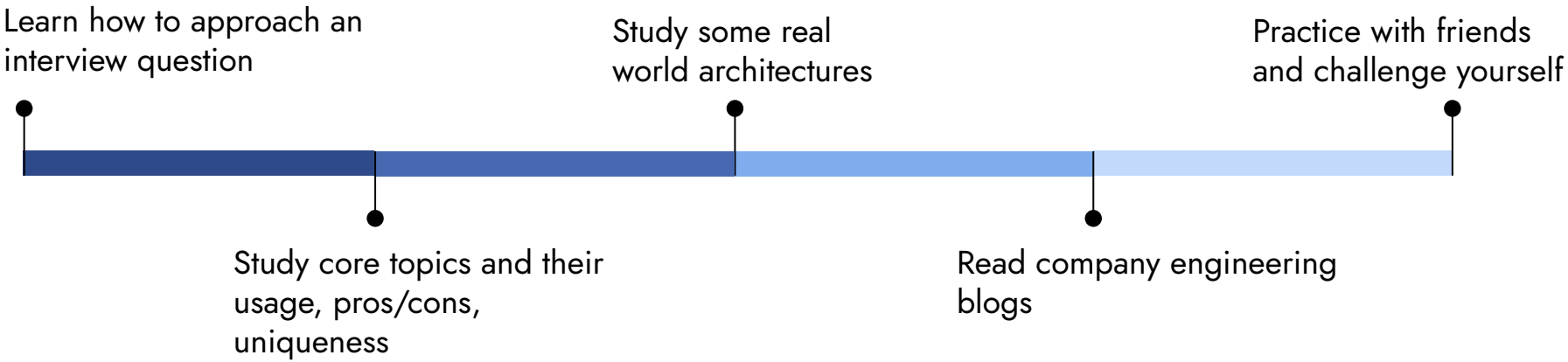
Do I need to know everything?

Absolutely f*cking **NOT!**

System Design questions mainly depends on some variables:

- Your experience
- Your technical background
- Applied company
- Applied position
- Luck!

The Plan


Learn how to approach an interview question
Study core topics and their usage, pros/cons, uniqueness
Study some real world architectures
Read company engineering blogs
Practice with friends and challenge yourself

The Approach

Step 1: Outline use cases, constraints, and assumptions

- Who is going to use it?
- How are they going to use it?
- How many users are there?
- What does the system do?
- What are the inputs and outputs of the system?
- How much data do we expect to handle?
- How many requests per second do we expect?
- What is the expected read to write ratio?

Step 2: Create a HLD (High Level Design)

- Draw the **core components** and their connections
- Question and justify your selections
- Specify the read/write scenarios

Step 3: Design the core components

No fixed formula.

Example:

- API design
- Database schemas
- Encryption

Step 4: Scale the system

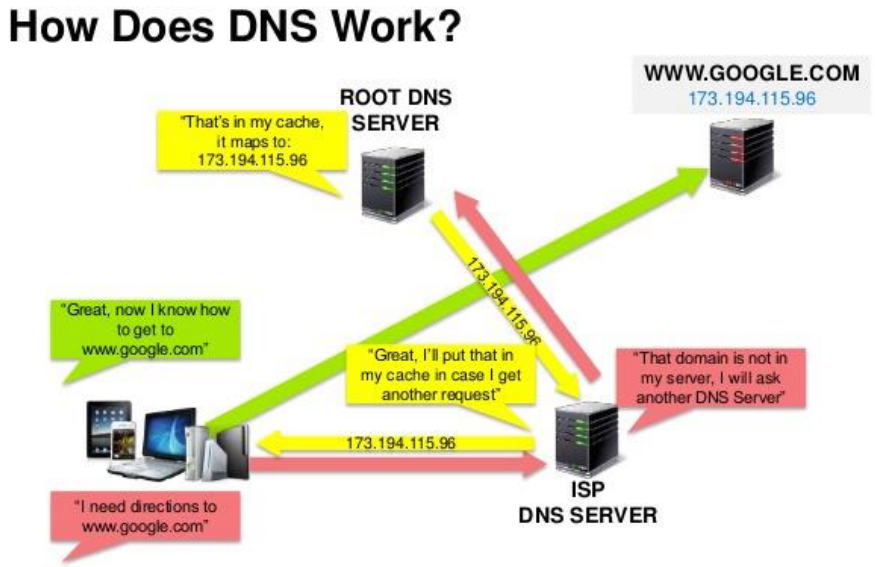
- Load balancers
- Horizontal/Vertical scaling
- Caching
- Reverse proxy?
- DB replication
- Discuss trade-offs

A very good template:

https://leetcode.com/discuss/career/229177/My-System-Design-Template

DNS

A Domain Name System (DNS) translates a domain name such as www.example.com to an IP address.



http://www.slideshare.net/srikrupa5/dns-security-presentation-issa

- **NS record** (name server) - Specifies the DNS servers for your domain/subdomain.
- **MX record** (mail exchange) - Specifies the mail servers for accepting messages.
- **A record** (address) - Points a name to an IP address.
- **CNAME** (canonical) - Points a name to another name or CNAME (example.com to www.example.com) or to an A record.

DNS Pros:

- Can do traffic routing based on various methods:
  - Weighted round robin
  - Latency based
  - Geolocation based

DNS Cons:

- Accessing DNS can cause slight delay. (Use caching to mitigate this)
- DNS server management is complex.
- DDoS attack is a possibility.

CDN

A content delivery network (CDN) is a globally distributed network of **proxy servers**, serving content from locations closer to the user.

https://www.creative-artworks.eu/why-use-a-content-delivery-network-cdn/

CDN Pros:

- Improves performance for serving contents.
- Users receive content from data centers close to them.
- Your servers do not have to serve requests that the CDN fulfills.

CDN Cons:

- CDN costs could be significant depending on traffic.
- Content might be stale if it is updated before the TTL expires it.
- CDNs require changing URLs for static content to point to the CDN.

Push CDN

Push CDN receives new content whenever changes occur on your server.

You take full responsibility of:

- Providing content
- Uploading directly to the CDN
- Rewriting URLs to point to the CDN
- Configuring when content expires and when it is updated.

When to Push CDN?

Sites with a small amount of traffic or sites with content that isn't often updated work well with push CDNs.

Note: Content is uploaded only when it is new or changed: **minimizing traffic**, but **maximizing storage**.

Pull CDN

Pull CDN grabs new content from your server when the first user requests the content.

Your responsibility:

- Determine the TTL (time-to-live) for how long content is cached on the CDN.

When to Pull CDN?

Sites with heavy traffic work well with pull CDNs, as traffic is spread out more evenly with only recently-requested content remaining on the CDN.

Note: Pull CDNs **minimize storage** space on the CDN, but can create **redundant traffic** if files expire and are pulled before they have actually changed.

Checkout these CDN services by yourself:

- CloudFlare
- Amazon CloudFront
- StackPath

Further study:

- http://www.travelblogadvice.com/technical/the-differences-between-push-and-pull-cdns/
- https://figshare.com/articles/journal_contribution/Globally_distributed_content_delivery/6605972

Load Balancer

Load balancers distribute incoming client requests to computing resources such as application servers and databases.

In each case, the load balancer returns the response from the computing resource to the appropriate client.

Why Load Balancers?

- Preventing requests from going to unhealthy servers.
- Preventing overloading resources.
- Eliminate a single point of failure.
- SSL termination:
  - Decrypt incoming requests and encrypt server responses so backend servers do not have to perform these potentially expensive operations
- Session persistence:
  - Issue cookies and route a specific client's requests to same instance if the web apps do not keep track of sessions
- Horizontal Scaling:
  - Improving performance and availability.

Load Balancer Cons

- Load Balancer is itself a single point of failure.
- Can become a performance bottleneck if not configured properly.

Load balancers can route traffic based on various metrics, including:

- Random
- Least loaded
- Session/cookies
- Round robin or weighted round robin
- Layer 4
- Layer 7

Layer 4 Load Balancing

Layer 4 load balancers look at info at the transport layer to decide how to distribute requests.

Generally, this involves the source, destination IP addresses, and ports in the header, but not the contents of the packet.

Layer 4 load balancers forward network packets to and from the upstream server, performing Network Address Translation (NAT).

Layer 7 Load Balancing

Layer 7 load balancers look at the application layer to decide how to distribute requests. This can involve contents of the header, message, and cookies.

Layer 7 load balancers terminate network traffic, reads the message, makes a load-balancing decision, then opens a connection to the selected server.

For example, a layer 7 load balancer can direct video traffic to servers that host videos while directing more sensitive user billing traffic to security-hardened servers.

Further study:

- https://www.nginx.com/blog/inside-nginx-how-we-designed-for-performance-scale/
- https://www.nginx.com/resources/glossary/layer-4-load-balancing/
- https://www.nginx.com/resources/glossary/layer-7-load-balancing/

Thank you.

Do you have any questions?

azmainadel47@gmail.com
Direct/WhatsApp: +880 1684 723252