Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google

MAIN COMPONENTS OF SYSTEM DESIGN

01 DNS

02 CDN

03 LOAD BALANCERS

04 WEB / APPLICATION SERVERS

05 CACHE

06 MICROSERVICES

07 BLOB/OBJECT STORAGE

08 MESSAGE QUEUES

09 DATABASES

10 PROXY


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


DOMAIN NAME SYSTEM (DNS)

DNS is a fundamental technology used on the internet to translate human-friendly domain names (like www.example.com) into IP addresses (like 192.0.2.1), which computers use to identify each other on the network.

Here's how DNS works:

1. **User Request**: You type a website's URL (domain name) into your browser.
2. **DNS Lookup**: Your computer sends a request to a DNS server, asking for the IP address associated with that domain.
3. **DNS Resolution**: The DNS server looks up the domain in its database. If it has the IP address, it sends it back to your computer. If not, it queries other DNS servers until it finds the correct IP address.
4. **Connection**: Your computer uses the obtained IP address to connect to the web server hosting the website. The server then sends back the webpage content to be displayed in your browser.

Examples - Amazon Route 53, Google Cloud DNS, and Cloudflare DNS


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


CONTENT DELIVERY NETWORK (CDN)

CDN is a network of distributed servers strategically placed around the world, working together to deliver web content, such as images, videos, and other resources, to users from a server that is geographically closer to them.

Here's how CDN works:

1. **User Request:** You type a website's URL in your browser.
2. **CDN Routing:** Request goes to the nearest CDN server.
3. **Content Check:** CDN checks if the content is cached.
4. **Cached Content:** Cached content is sent if available.
5. **Retrieve if Needed:** If not cached, CDN fetches from the original server.
6. **Balancing and Speed:** CDN balances traffic, reducing latency.
7. **Quick Display:** Content loads faster in your browser.
8. **Enhanced Experience:** CDN ensures smoother browsing.

Examples - Cloudflare, Amazon CloudFront


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


LOAD BALANCER

A Load Balancer is a crucial component in distributing incoming network traffic across multiple servers to ensure optimal resource utilization, minimize response time, and prevent overload.

Here's how Load Balancer works:

1. **Traffic Distribution:** When a user sends a request to a website, it first reaches the load balancer.
2. **Server Selection:** The load balancer evaluates the current load on each server in the backend pool.
3. **Algorithm Logic:** Using a predefined algorithm (e.g., Round Robin, Least Connections, etc.), the load balancer selects an available server.
4. **Traffic Forwarding:** The load balancer forwards the user's request to the chosen server.
5. **Server Response:** The selected server processes the request, generates a response, and sends it back to the load balancer.
6. **Response to User:** The load balancer receives the response and delivers it to the user who made the initial request.

Examples - NGINX, HAProxy


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


WEB / APPLICATION SERVERS

A Web/Application Server is a specialized software designed to host and manage web applications, providing a platform for processing and delivering content over the internet.

Here's how it works:

1. **Client Request:** The User's browser requests a web page or app.
2. **Routing:** The web server sends a request to the right app server.
3. **App Processing:** The app server handles logic, and talks to databases.
4. **Data Retrieval:** Fetches needed data from databases.
5. Content Generation: Creates dynamic content (HTML, etc.).
6. **Response Generation:** Prepares a response with content.
7. **Response Delivery:** Sent back to a web server.
8. **Client Display:** The browser gets a response, and shows the page.

Examples - Apache Tomcat, Glassfish


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


CACHE

The cache is like a super-fast memory that stores frequently used data, making it quicker to access.

Here's how Cache works:

1. **Request**: When you request data, the system first checks the cache if it has it.
2. **Cache Hit**: If the data is in the cache (cache hit), it's fetched quickly.
3. **Cache Miss**: If not (cache miss), data is fetched from the original source and stored in the cache.
4. **Future Requests**: The next time you need the same data, it's already in the cache for quick access.

Examples - Redis, Memcached


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


MICROSERVICES

Microservices work by breaking down a software application into small, independent services that can be developed, deployed, and scaled individually. These services communicate with each other through well-defined APIs, and together they create a larger application.

Here's how microservices works:

1. **Decomposition:** Breaks down an app into small, independent services.
2. **Modularity:** Each service handles a specific function or feature.
3. **Communication:** Services communicate through APIs (Application Programming Interfaces).
4. **Isolation:** Services can be developed, deployed, and scaled independently.
5. **Scalability:** Scale individual services as needed for better performance.
6. **Resilience:** Failure in one service doesn't affect the entire app.
7. **Agility:** Enables rapid development and updates for each service.
8. **Flexibility:** Supports different tech stacks for different services.
9. **Complex Apps:** Combine multiple services to create a complete application.
10. **Maintenance:** Easier to manage and update specific services.


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


BLOB / OBJECT STORAGE

Storage is a type of data storage used to store and manage unstructured data, such as images, videos, documents, and more, in its native format.

Here's how storage works:

1. **Data Upload:** Users or applications upload files directly to the blob/object storage system.
2. **File Segmentation:** The uploaded files are divided into smaller chunks or objects. Each object is assigned a unique identifier and metadata.
3. **Distributed Storage:** Objects are distributed across multiple servers or nodes in a storage cluster. This ensures redundancy, fault tolerance, and improved performance.
4. **Metadata Storage:** Metadata, which contains information about the objects (e.g., file type, size, creation date), is stored alongside the objects.
5. **Access and Retrieval:** Users or applications can retrieve objects using their unique identifiers. The storage system locates and retrieves the required objects from the distributed nodes.
6. **Data Management:** Blob/object storage offers features like versioning, encryption, access control, and lifecycle policies for efficient data management.
7. **Data Access:** Objects can be accessed using APIs, enabling applications to integrate and interact with the storage service.
8. **Global Accessibility:** Many blob/object storage services offer global accessibility, allowing users to access their data from various geographical locations.

Examples - Amazon S3, Google Cloud Storage


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


MESSAGE QUEUES

A message queue is a communication tool that lets different software parts send and receive messages asynchronously. It helps components exchange data without being connected in real time, enhancing scalability and reliability in applications.

Here's how message queues works:

1. **Sending Messages**: When a component wants to send a message, it adds it to the message queue. This message can contain data, commands, or requests for other components.
2. **Queue Management:** The message queue acts as a buffer, holding the messages until the receiving component is ready to process them.
3. **Receiving Messages:** The receiving component checks the message queue for new messages at its own pace. It retrieves and processes messages one by one.
4. **Asynchronous Communication:** Since components communicate independently, they don't need to be active simultaneously. This enables more flexible and efficient interactions.
5. **Decoupling:** Message queues decouple components, allowing them to evolve independently without affecting each other. Changes in one component don't directly impact others.
6. **Message Acknowledgment:** Once a message is successfully processed, the receiving component can acknowledge its completion, and the message can be removed from the queue.

Examples - Apache Kafka, RabbitMQ, Apache ActiveMQ


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


DATABASES

A database is a structured collection of data organized for efficient storage, retrieval, and manipulation. It serves as a digital repository that stores and manages information, allowing users and applications to interact with data.

Here's how a database works:

1. **Data Structure:** Data is organized into tables, rows, and columns, forming a structured format that represents real-world entities and their relationships.
2. **Data Storage**: Databases store data on storage devices, such as hard drives or cloud storage. Data is saved in a structured manner to ensure efficient retrieval and updates.
3. **Data Retrieval:** Users and applications can query the database to retrieve specific data based on criteria. Queries are written using query languages like SQL (Structured Query Language).
4. **Data Manipulation:** Databases support various operations, such as adding, updating, or deleting data, ensuring data integrity and consistency.
5. **Indexing:** Indexes are created on specific columns to speed up data retrieval. They act like a table of contents, enabling the database to locate data faster.
6. **Data Relationships:** Databases allow defining relationships between tables, ensuring accurate representation of complex data scenarios.
7. **Data Security:** Databases implement security mechanisms to control access and protect sensitive information. Users are granted specific permissions to perform authorized actions.
8. **Concurrency Control:** Databases manage simultaneous user access to prevent conflicts and ensure data consistency during concurrent operations.
9. **Backup and Recovery:** Regular backups of the database are taken to prevent data loss. In case of failures, databases can be restored to a previous state.

Examples - Google Cloud Spanner, Firestore, Oracle, MySQL


Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google


PROXY

A proxy serves as an intermediary between a client and a destination server, enabling clients to make requests indirectly through the proxy server. Proxies offer various benefits, including security, privacy, and performance optimization.

Here's how a proxy works:

1. **Client Request:** The client (e.g., user's computer) sends a request to access a resource (e.g., a website).
2. **Proxy Server:** Instead of directly reaching the destination server, the client's request is intercepted by a proxy server.
3. **Forwarding Request:** The proxy server evaluates the request, and based on its configuration, it forwards the request to the destination server on behalf of the client.
4. **Destination Server Interaction:** The destination server processes the request as if it came directly from the client. It sends back the response to the proxy server.
5. **Proxy Server Response:** The proxy server receives the response from the destination server.
6. **Response to Client:** The proxy server then forwards the response to the client that made the initial request.

Examples - HAProxy, AWS Fargate

Hina Arora
Engineering Manager @ Jio

Dinesh Varyani
Engineer@ Google

IF YOU LIKE THE CONTENT DON'T FORGET TO FOLLOW US

Hina Arora

Dinesh Varyani