Scalability part2 Cheat Sheet

by saran sudha via cheatography.com/202593/cs/43102/

Reverse proxy

- Distribute incoming traffic across multiple backend servers
- Ensures that no single server becomes overwhelmed with requests
- andles routing traffic to the available servers, making it simple to scale horizo-ntally
- Can easily scale application by adding or removing backend servers as needed
- Compress responses before sending them to clients, reducing the amount of data transmitted
- Offloading SSL termination to the reverse proxy alleviates the resource-intensive task of encryption/decryption from backend servers

Load Balaner

- Distributes incoming traffic
- Ensures optimal resource utilization, fault tolerance, and scalability as the application grows

Types:

Hardware Load Balance

- Supports L4* and L7*
- Higher cost
- Limited flexibility and agility compared

Software Load Balancer

- Supports only L7*
- Lower cost compared to HLB
- Greater flexibility and agility in deployment and management.

L4 - Transport Layer of the OSI includes UDP,TCP, SCTP

L7 - Application Layer of the OSI which is highest layer includes HTTP, HTTPS, SFTP etc

DNS as load balancer

- Involves leveraging DNS to distribute incoming client requests across multiple backend servers or resources.
- Returns single IP in round robin fashion
- can be configured along with health checks
- DNS records can be updated dynamically based on server availability and load condit-ions.
- it's essential to consider the limitations and potential latency issues associated with DNS caching and propagation

Drawback

- Indefinite caching
- Low or zero TTLs
- Very high load on DNS

SAGA Pattern

- SAGA breaks down complex transactions into smaller, loosely coupled units of work called "sagas."
- In a microservices architecture each microservice can represent a step in the saga, allowing them to be scaled indepe-ndently based on demand
- Asynchronous nature can be leveraged to improve scalability by decoupling services and allowing them to scale independently
- the SAGA pattern's transaction decomposi-tion, asynchronous communication, and distributed coordination can aid in constr-ucting scalable distributed systems.