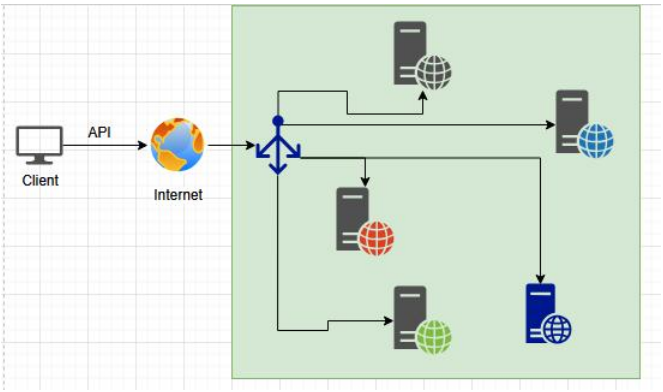
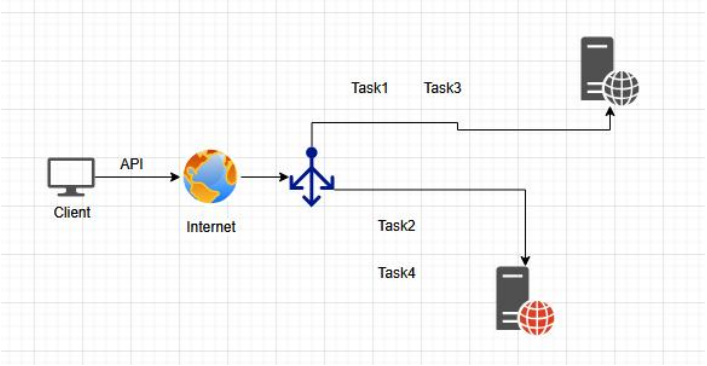
LoadBalancing

Say we have multiple webservers, in the middle, we will have a LoadBalancer responsible for distributing user requests to webservers. Our focus is the right-hand side from load-balancer to webservers. What's load? User requests, traffic or concurrent users

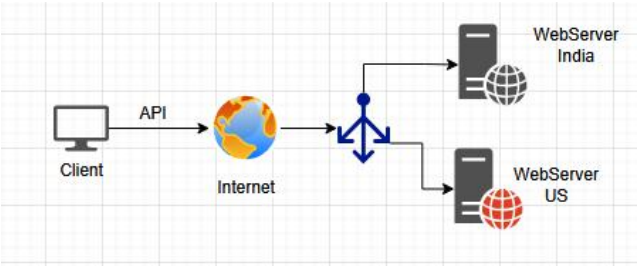


LoadBalancing algorithms
1. RoundRobin:



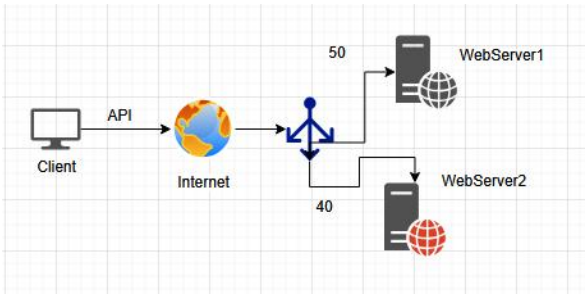
LoadBalancer will distribute Task1 to WebServer1, then Task2 to WebServer2, Task3 to WS1 and Task4 to WS2. RoundRobin sends requests one-by-one to webservers.
Benefits: Equal distribution, Simple, No need to manage state of any webserver (State means whether we get any response from ws or not, was the response ok or not), Lightweight loadbalancer

2. Geo-based:



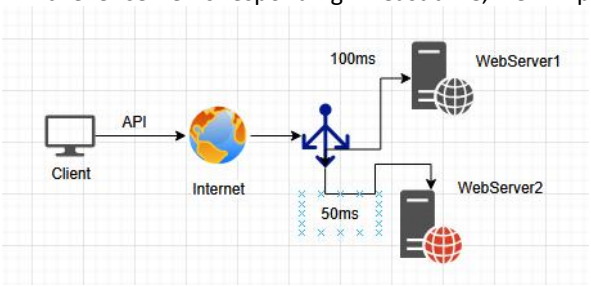
All requests from India will go to WebServer India and all requests from US will go to WebServer US. Is this is good strategy? Maybe not, because we are not considering the capacity of WebServers etc. Or say, we don't have many requests from India then India WebServer will sit idle for long.
Benefits: Fast, Simple

3. Least-Connections:



Lets say we get 40 requests to WebServer2 and 50 to WebServer1. The next request will eventually go to WebServer2 because of less load. Say if load drops to 25 from 50 for WebServer1 then the next new request will go to WS1 only < 40.
Loadbalancer is maintaining a state. In State, it is maintaining the number of connections each WebServer is handling.

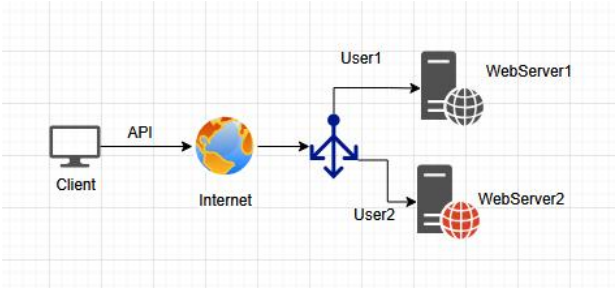
4. Least Time:
Whichever server is responding in least time, we will provide the next task to that server



Say WS1 is responding in 100ms and WS in 50ms, most of the responses will go to WS2 only. If WS2 response time increases to 110ms that time we shift the requests to WS1
LB will manage state (Average response time)

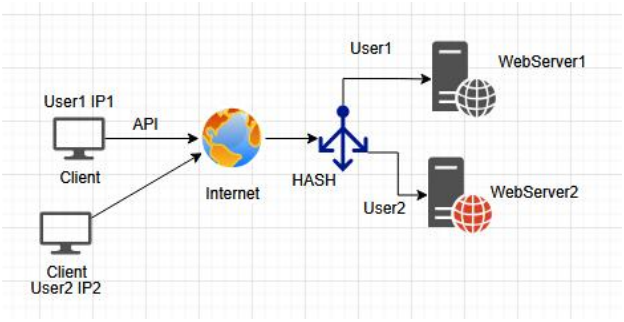
In real world scenario, we will implement a hybrid model like a combination of Geo-based and Least-connections etc.

5. Sticky session / Session affinity:



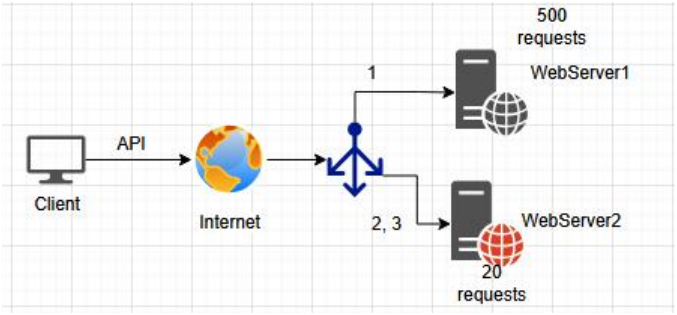
We will maintain User1 session with WebServer1. We are sticking users to webservers based on the session we have. Session information can be stored in JWT.
Here also, Webservers will maintain the state, user groups

6. IP Hash:
Users will have IPs and IPs are converted into Hash values using Hash function. Based on the Hash value, they are assigned to Webservers



Based on IP, we provide Hash function in LoadBalancer to convert IPs into Hash, accordingly it will navigate to Webservers. For a certain range of Hashes, it will be served by WS1 and for a range will be served by WS2.

7. Weighted RoundRobin:
Loadbalancer will have some or the metrics to check weight of webservers could be number of connections, size of webservers, response time or hybrid combinations



Lets say WS1 is currently handling 500 requests already and WS2 is handling only 20 requests, request 1 is heavy, request 2 is routed to WS2 as per RoundRobin, now request 3 must go to WS1 but we will break the pattern of regular RoundRobin and assign request 2 to WS2 itself because request 1 was heavy for WS1

Health checks:
Our loadbalancer is driving requests to webservers. Our loadbalancers are responsible for not burning our webservers. It cannot be like WS1 is taking a lot of load and WS2 is not taking that much load unevenly. Certain healthchecks we can have in our loadbalancer to make them more functional.
Two types of healthchecks: Active and Passive
Passive: It will observe all requests (all real calls) from client.
Active: It will generate mock/fake calls to webservers to check performance/response. We are creating fake loads for webserver to handle. Say we are going to scale our system, to perform sensitivity analysis, we will send a lot of mock calls.

HealthCheck parameters:
- Interval (How often do we check webservers? Every 5 sec, 5 min, 2 days?
- Timeout (How long do we wait for the response? Max 1 sec, that means our Loadbalancer will wait for max 1 sec.
- Unhealthy threshold: Markdown after N failures. Say we can set the threshold to 10 failures
- Healthy threshold: Markup after N successes. Say after 10 success requests, we are saying our system is again healthy.

Examples:
Amazon works on DynamoDB sessions / ELB to maintain loadbalancers
Facebook / Twitter X work on Memcache + Redis
Netflix works on Stateful JWT + Regis
Shopify works on Redis

Homework: How does Google calender handle loadbalancer?