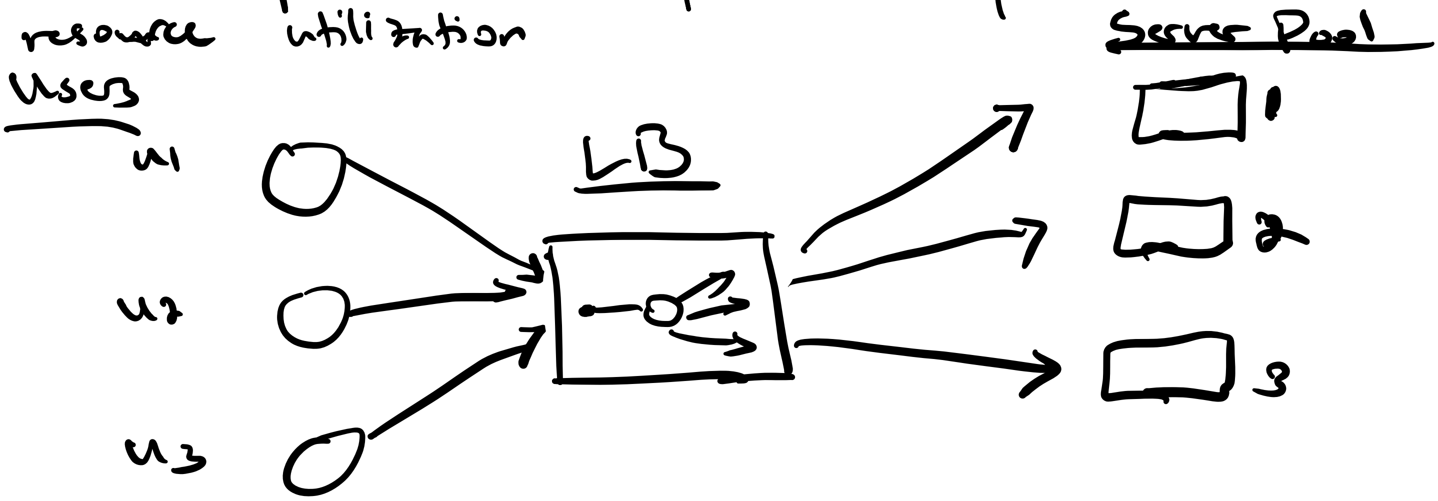
Load Balancers

- What is it?
  - To serve millions of incoming requests, thousands of servers (or even hundreds of thousands) work to share the load
  - They share the load via load balancers
  - The goal of a LB:
    - ↳ fairly divide all client's requests among the pool of available servers
      - done to avoid crashing the servers
  - You may not need an LB for a service that entertains only a few hundred → thousand requests
  - For those that need them, load balancers provide systems:
    - ↳ scalability: by adding servers, the capacity of the app can be ↑ seamlessly. They make upscaling or downscaling transparent to the end user
    - ↳ Availability: even if some servers go down, the system will still be available. LBs also are meant to hide faults & failures of the servers.
    - ↳ Performance: They quickly forward requests to servers w/a lesser load so the user gets a faster response. This improves both performance & resource utilization

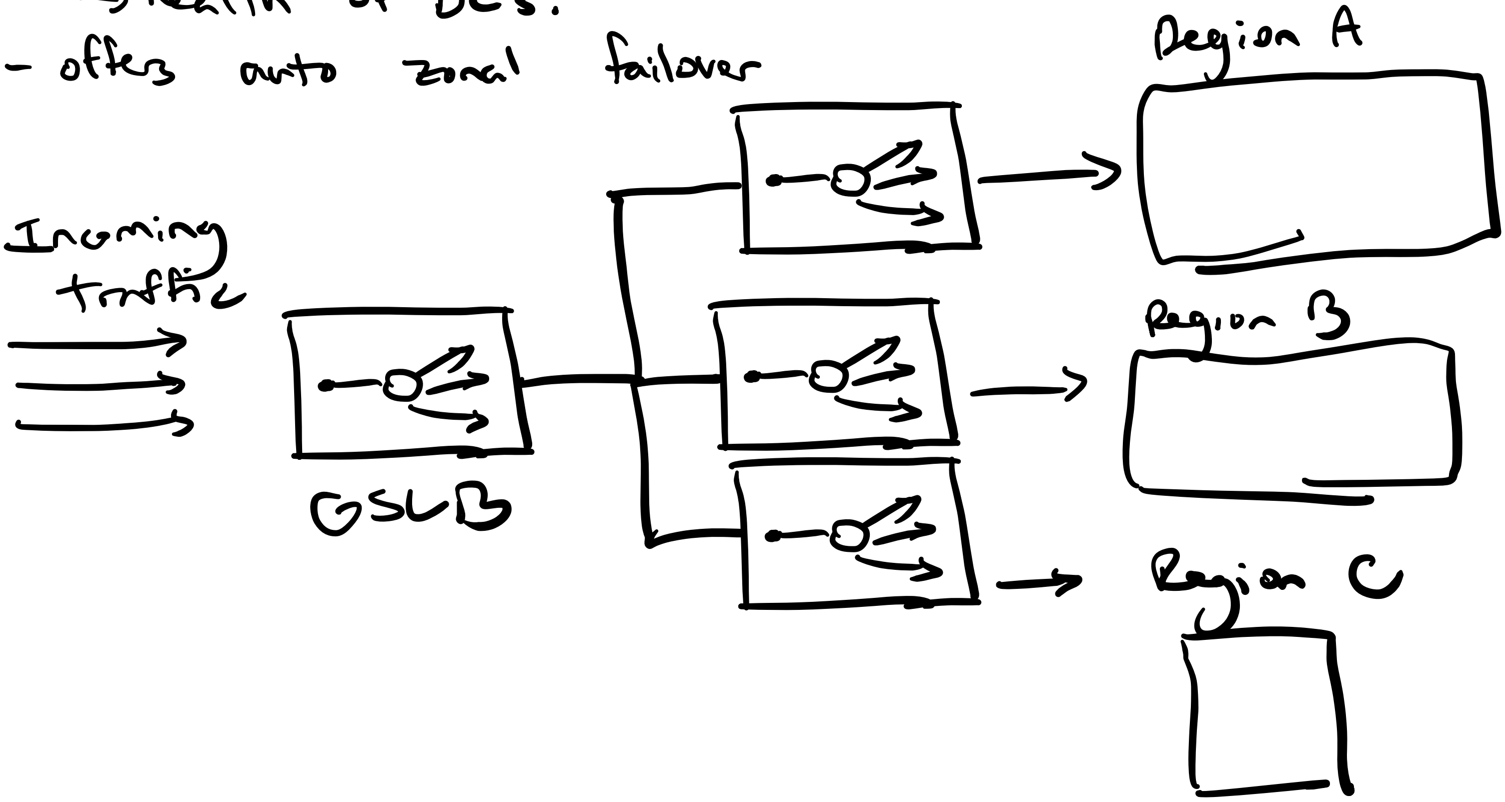


- Placing LB`s
  - Usually, LB`s sit b/t clients & servers
    ↳ Requests go thru to servers & back to clients via the LB layer
      - Not the only point that they're used
  - Place LB`s between:
    ↳ end users of the application & web servers/application getaway
    ↳ web servers & application servers that run the business/app logic
    ↳ b/t app servers & db servers

- LB services
  - Health checking: LBs use the heartbeat protocol to monitor the health (& thus, reliability) of end servers
  - TLS termination: They reduce the burden on end servers by handling TLS termination w/the client
  - Predictive analytics: Can also predict traffic patterns thru analytics performed over traffic passing thru or using stats of traffic over time
  - Service discovery: client's requests are forwarded to appropriate hosting servers by inquiring about service registry
  - Security: They can mitigate DOS attacks @ diff layers of the OSI model (layers 3, 4, 7)

- Global server load balancing (GSLB)
  - involves the distribution of traffic load across multiple geographical locations
  - ensures that globally arriving traffic load is intelligently forwarded to a data center (dc)

- Makes forwarding decisions based on:
  ↳ user's geo location
  ↳ num of hosting servers in diff locations
  ↳ health of DCs.
- offers auto zonal failover



- GSLB can forward requests to 3 diff data centers
- each LB layer w/in a DC will maintain info about the health of LBs & server farm
- GSLB uses this info to drive traffic decisions & forward traffic load based on each regions config & monitoring in

• Local load balancing
- LB-ing achieved w/in a data center
- this type of LB-ing focuses on improving efficieny & better resource use of the hosting servers in the data center
- They behave like a reverse proxy

- Advanced Details
  - Algorithms
    - ↳ round robin scheduling: requests forwarded to a server pool in repeated, sequential manner
    - ↳ weighted RR: servers w/ higher capability have ↑ weight
    - ↳ Least connections: Nodes w/ fewer connections get requests
    - ↳ Least response time
    - ↳ IP Hash: Server decided by IP Hash
    - ↳ URL Hash: Some services w/in the app are provided by specific servers only. URL Hash function is used

  - Static vs. Dynamic Algorithms
    - ↳ Static:
      - don't consider the changing state of the servers
      - task assignment done using existing knowledge on servers config
      - not complex
    - ↳ dynamic:
      - consider recent state of servers
      - maintain state thru communication (adds overhead)
      - require diff LBs to communicate
      - can be modular → more complexity, but better decisions

  - Stateful vs. Stateless
    - ↳ Stateful
      - involves maintaining a state of the sessions est. b/t clients & hosting servers
      - stateful LB includes state in its algo to perform LB-ing
    - ↳ stateless
      - maintains no state: faster, lightweight
      - use consistent hashing to make forwarding decisions

- not as resilient as stateful LBs: consistent hashing alone isn't enough to route a request to the correct app server
    - ↳ therefore a local state may also be required
- state managed across diff LB's: stateful
- state managed w/in a LB: stateless

- Types of LBs
  - Layer 4
    - ↳ LB-ing performed on the basis of transport protocols like TCP UDP
    - ↳ maintain connection/session w/clients
    - ↳ ensure TCP/UDP communication ends up being forwarded to the same backend server
    - ↳ TLS termination: some L4 LBs support it
  - Layer 7
    - ↳ based on data of app layer protocols
    - ↳ possible to make app aware forwarding decisions based on HTTP headers, URLs, cookies, & other app specific data
      - for example, User ID
    - ↳ TLS termination, rate limiting users, HTTP routing, header rewriting

- LB deployment
  - DNS's are tier 0 LBs
  - ECMPs (equal cost multipath) routers are tier 1 LBs
    - ↳ divides traffic based on IP or RR or weighted RR
    - ↳ Tier 1 LBs balance load across diff paths to higher tier LBs
    - ↳ play a vital role in horizontal scalability of higher tier LBs

↳ Tier 2 LBs: include layer 4 LBs

- Make sure that for any connection, all packets are forwarded to the same tier 3 LBs
- Glue b/t tier 1 & tier 3 LBs.

↳ Tier 3 LBs: In direct contact w/ backend servers

- perform health monitoring of servers @ HTTP level
- This tier enables scalability by evenly distributing requests among healthy back-end servers
- High availability by health-monitoring servers directly
- Idea: leave computation & data serving to app servers & effectively utilize LB commodity machines for trivial tasks