Load Balancing

Arnav Gupta

December 7, 2024

Contents

1 Load Balancers

Single point of contact for a service that forwards requests to a cluster of workers and remembers request source. Load balancer can relay requests for 10s-100s of applications servers.

One IP address appears like a big machine, but is actually a cluster.

Load balancer provides the same interface as a single server.

Individual servers can be replaced without affecting overall service. Proxy can monitor health of servers by periodically sending health check, and if request fails, server must be crashed (stop relaying requests there).

2 types of local load balancers:

- **Network Address Translation**
  - works at TCP/IP layer (layer 4 load balancer)
  - forwards packets one-by-one but remembers which server was assigned to each client
  - changes IP addresses and ports of packets in both directions (maintains mappings)

        - more efficient since need not implement TCP, HTTP, or store full requests/responses
    - compatible with any type of service, not just HTTP
    - more scalable
- **Reverse Proxy**
    - works at HTTP layer (application layer load balancer)
    - stores full requests/response before forwarding
    - can also do SSL termination, caching, and compression
        - TLS/SSL certificates stored just on proxy (internal communication unencrypted)
        - proxy might cache responses, but limits scalability
    - less scalable

Cloud-based load balancers can do either type of load balancing.

Load balancer is single point of failure and has limited throughput.

2 Domain Name Service

Distributed directory that maps hostnames to IP addresses. Uses hierarchical caching architecture for scalability.

Local DNS resolver has cached copies of recent answers. If no answer, ask up the hierarchy (go to nameserver).

DNS allows multiple answers to be given for a query (client chooses or DNS server can give different responses to different users). Each IP address is a different reverse-proxying load balancer in front of many app servers (scaling by DNS).

DNS can also connect user to closes replica of service (geographically) using IP address geolocation of requester.

3 Content Delivery Network (CDN)

Globally distributed web servers that cache responses for local clients.

Distributed caching HTTP reverse proxy that uses DNS to geographically load-balance.

Origin server: original, central web server (sets `Cache Control` HTTP header in response)

Edge servers are **caching proxies** and ask origin server if no cached response. This is where CDNs are.

8.8.8.8 is where Google's DNS is.

IP Anycast load balancing implemented with BGP. Traffic destined for 8.8.8.8 is sent to whichever of these Google routers are closest to the customer. Violates principle that IP address is a particular destination, but for DNS this doesn't matter since UDP and stateless.

DNS vs IP Anycast:

- DNS TTL has slow changes (minutes to hours) but IP Anycast uses BGP convergence
- DNS required advanced DNS software/config but IP Anycast requires operating own Autonomous System (ISP)

For global load balancing, DNS is the most common choice.

Tech giants can use IP Anycast.

For local load balancing:

- routing done by NAT or HTTP proxy
- scale limited by speed of one machine
- changes can be made in milliseconds
- simple deployment using off-the-shelf software and hardware

For global load balancing:

- routing done by DNS or BGP
- scale limited by Internet itself
- changes can be made in minutes to hours
- deployment requires advanced DNS software/config

Most large services load balance locally and at DNS:

- local provides continuous operation and scale within a data center (health checks, rolling updates)
- DNS for global scalability and low latency