Using a Load Balancer in the CORE Emulator

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

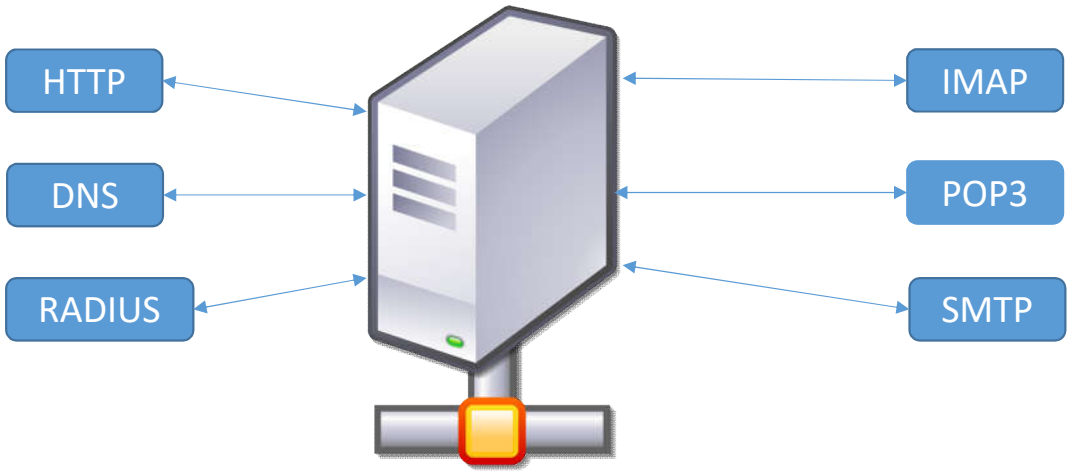
REDES DE COMPUTADORES – Prof. Tiago Ferreto

Aluno Március R. Neutzling Dias

PUCRS | ESCOLA POLITÉCNICA

Some historical perspective.

- Commercial Internet started in Brazil in 1995.
- First ISPs (Internet Service Providers) UOL, BOL, NutecNet/ZAZ/Terra.
- Mostly everybody follow the model "**one server to rule them all**".

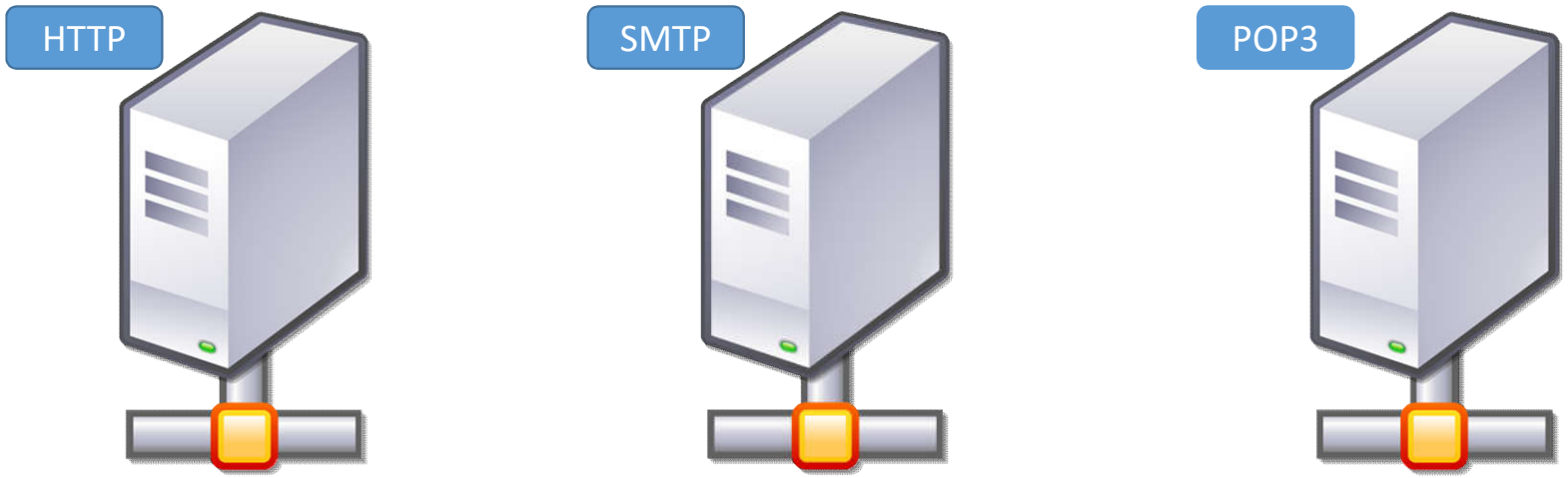


Fortunately the business grew up ☺.

- To scale services, upgrade the (only one) server.

- Switch to a new server after couple of months (not very good ☹).

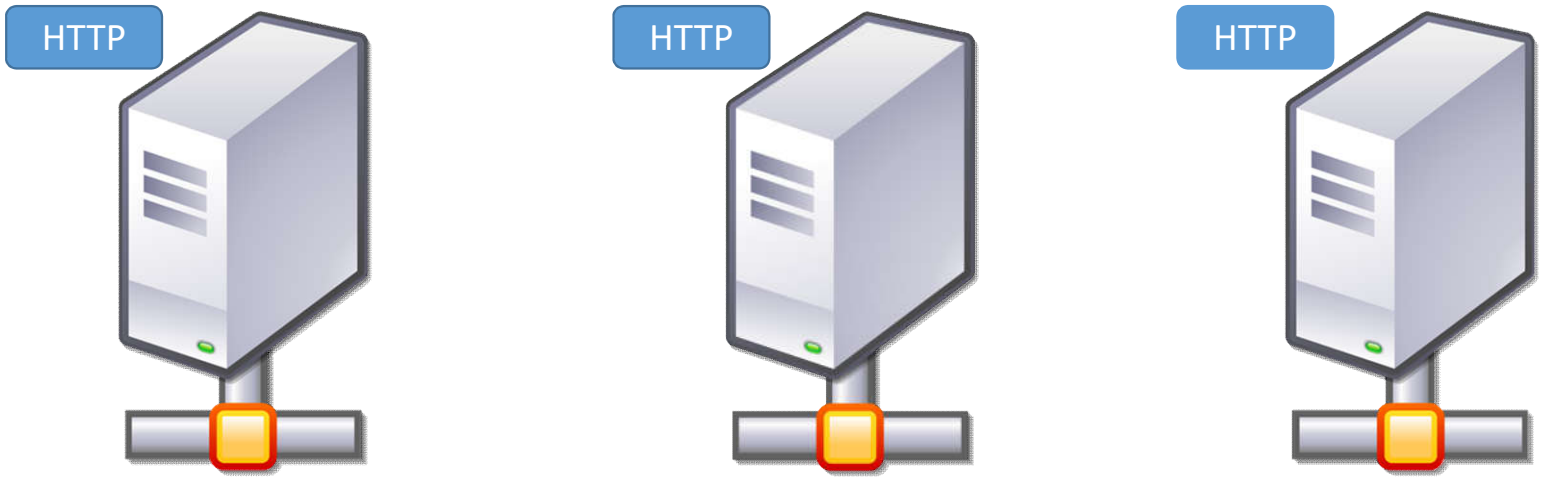
Big idea, split the services between servers!



Fortunately the business kept growing ☺☺.

- To scale services, upgrade the (only one) server.
- Switch to a new server after couple of months (not very good ☹☹).
- This sounds familiar…

Big idea, split the same service between many servers!



Sounds great, but how to do that ?

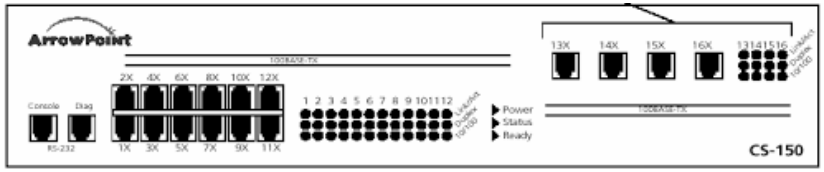
- Let’s use a load balancer. Problem, they don’t exist yet.

- **DNS Round-robin**! We can have multiple IP serving same service.

DNS Round-robin – drawbacks (few of them).

- Just does load distribution (and not very well).
- DNS caches (client side) can affect load distribution.
- If one server fail, users keep reaching the server.
- There is no persistence (we can’t have sessions).

Load balancers 1st generation (ArrowPoint).



Load balancers 1st gen, features.

- Just L4 TCP (no UDP).
- All servers must be physically connected to them.
- Not exactly the most stable platform (strange things happened in the LAN).

Load balancers 2st generation (Alteon)

Load balancers 2st gen, much better.

- We can balance udp based services (radius, DNS).
- Now we have L7 and it works!
- We still have a few issues, but much more stable.

Load balancers today (BIG-IP i15000 Series).

Today we have software based load balancers.

- Linux Virtual Server (L4)
- HAProxy (L7)
- Virtual/Cloud load balancers.

So what for load balancers are used to ?

- Horizontal escalability.

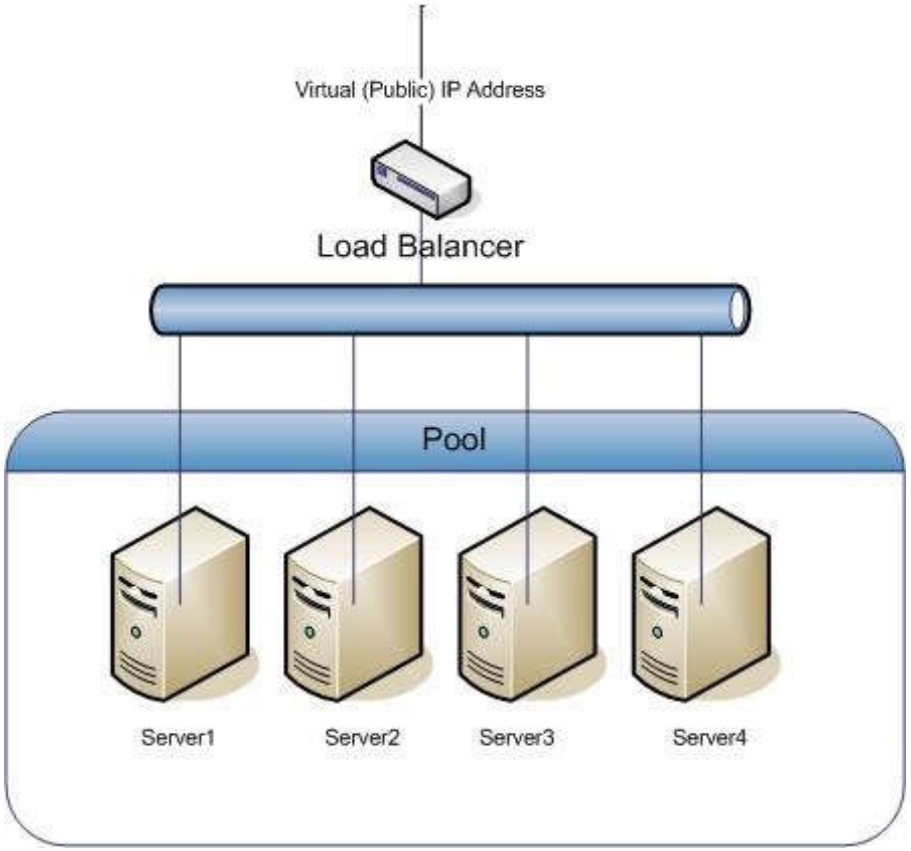
- High availability.

How does it Works ?

- Load balancer has a virtual public IP address (VIP).
- Redirect requests to backend servers running the service.
- Backend server responds to the load balancer.

How does it Works - schema

https://medium.com/@netscylla/secret-holes-behind-the-common-load-balancer-e7f70fcaa353



Why L7 it is relevant ?

- Pros
  - The content can be modified.
  - We can have rewrite rules.
  - Clients and servers are not required to use the same protocol (for example IPv4 vs IPv6, clear vs SSL).

- Cons
  - Much more cpu intensive.
  - There is no DRS (direct server response).

Load balancing – algorithms and estrategies

- Round-robin (for short connections, pick each server in turn).
- Leastconn (pick the least recently used of the servers with the lowest connection count).
- Least Response Time.
- Source (directly depends on the client's source address)
- Algorithms must support per-server weights so that it is possible to accommodate from different server sizes/generations in same farm.

Hashing and persistence

- Consistent hashing protects server farms against massive users redistribution when adding or removing servers in a farm.

- Hashing can apply to various elements such as client's source address, URL components, header field values, cookie.

- That's very important in large cache farms and it allows slow-start to be used to refill cold caches.

HAProxy sample conf file part 1.

```
# HAProxy configuration to run under Core enviroment.
global
    debug           # uncomment to enable debug mode for HAProxy

defaults
    #log /dev/log local0    # enable log on syslog (doesn't work under chroot)
    log localhost:514 local0        # enable log on syslog (to work under chroot)
    mode http               # enable http mode which gives of layer 7 filtering
    timeout connect 5000ms      # max time to wait for a connection attempt to a server to
succeed
    timeout client 50000ms      # max inactivity time on the client side
    timeout server 50000ms      # max inactivity time on the server side
```

HAProxy sample conf file part 2.

```
frontend http
    bind localhost:80
    bind 10.0.0.1:80
    default_backend apache_80

backend apache_80
    server node01 10.0.1.10:80 check fall 3 rise 2
    server node02 10.0.1.11:80 check fall 3 rise 2
    server node03 10.0.1.12:80 check fall 3 rise 2
```

server node01 10.0.1.10:80 check fall 3 rise 2

- Check This instructs HAProxy to perform health checks on the node.

- Fall 3 Instructs HAProxy to offline the node if 3 consecutive health check failures occur.

- Rise 2 If the node is marked offline due to failed health checks, this instructs HAProxy to not mark the node online unless it has two consecutive successful health checks.

It is demo time!

References.

- HAProxy Documentation http://cbonte.github.io/haproxy-dconv/1.9/intro.html
- Secret Holes Behind the Common Load-Balancer https://medium.com/@netscylla/secret-holes-behind-the-common-load-balancer-e7f70fcaa353
- How to Configure HAProxy Health Checks https://www.serverlab.ca/tutorials/linux/network-services/how-to-configure-haproxy-health-checks/