Load-Balancing
Introduction (with examples...)

For AFNOG 2014
By Laban Mwangi
Lmwangi_ta_gmail.com
(Rework of slides from Joel Jaeggli)
What is Load-balancing

- The act of dividing a workload between \( N > 1 \) devices capable for performing a task.
- Multiple contexts in internet services where this concept occurs.
  - DNS
  - MX records
  - Multiple links (L2 trunks, L3 ECMP)
  - Multiple servers
Goals

- Greater scalability
  - Horizontal scaling. Just add more switches/servers...
- Higher availability
  - Don't care about single device failure. Route around failures automatically!
- Reduced cost
  - Cheaper to use commodity hardware and architecture for failure. Examples: AWS/GCE...
amaze..
Quick Survey

• L2
  • LACP (Switches)

• L3
  • L3 ECMP (Switches, Routers, OS kernel)

• L4
  • HAProxy (OS userland)

• L4+
  • NGINX (OS userland)
  • HAProxy (OS userland)
  • F5, A10, Netscaler... (Hardware..)
Examples: L2 – Link aggregation

- Widespread support for **LACP** (Link Aggregation Control Protocol)
- Bond two physical layer 2 channels into one logical one.
  - Resilience against single port/channel failure.
  - L2 Bandwidth scaling
- Balancing and dynamic behaviour is important!
Examples: L3 - Equal-cost multi-path routing (ECMP)

- Packets are forwarded to the next hop over links having an equal routing cost.
- Stateless mode breaks TCP (PMTU)
  - Different hops may have different MTU settings
  - TCP sensitive to re-ordering

- We need a way to make flows stateful.....
Examples: L4 - Equal-cost multi-path routing (ECMP) + hashing

- If packets in a TCP session take the same path...
  - Path MTU issues would be fixed
  - Re-ordering would be fixed
- Different TCP sessions can take different paths.
- We need a way to uniquely identify L4 sessions.
- What attributes do you think would identify a TCP session?
Flow identification (5-tuple)

- XOR hash of fields to generate a flow id.
- Hash src & dest ip addresses, protocol number from the IP header and ....

<table>
<thead>
<tr>
<th>Offsets</th>
<th>Octet</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Version</td>
<td>IHL</td>
<td>DSCP</td>
<td>ECN</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>32</td>
<td>Identification</td>
<td>Flags</td>
<td>Fragment Offset</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>Time To Live</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>96</td>
<td>Source IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>128</td>
<td>Destination IP Address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>160</td>
<td>Options (if IHL &gt; 5)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5-tuple continued

• … hash of port numbers.

• How?
  • Example: CRC32(src_ip, dst_ip, pr_no, src_port, dst_port) % count of links
What does an L4 load Balancer do?

- Looks and the Destination IP and Port to determine which endpoint to send a packet/flow to in a pool of servers.
- Forwards the incoming connection to one pool member on the basis of policy (liveness, load).
- May keep the connection pinned to the particular pool member by tracking the connection.
- But... This breaks scaling!
  - Existing flows won't be remapped dynamically!
  - An LB/server failure would break a session!
What does an L7 load balancer do?

• An L7 load balancer answers incoming connection requests.

• It understands the protocol being spoken across the connection (e.g. HTTP IMAP FTP etc).

• On the basis of either 5-tuple hash or some higher layer value, (example a URI or a cookie or both) the request is directed to a member of the appropriate pool.

• L7 is another word for proxy or ALG (Application Layer Gateway).
Isn't L7 going to be slower than L4?

- Probably but not always.
- Importantly there are optimizations that can reduce the expense.
  - TCP syn-cookies
  - Connection pooling
  - Consider 3-way handshake
Applications - Cont

• Open source
  • Apache mod_proxy_balance
  • Squid
  • Haproxy
  • NGNIX
  • LVS
Applications Commercial

• Commercial
  • F5
  • Netscalar
  • A10
• Benefits of a commercial approach
  • Coordination of supporting elements
    – Routing
    – DNS
    – Complex health checks
    – HA
• Can have ASIC based acceleration.
High Availability Approaches

- Active-Passive
  - VRRP
  - State replication
- Active-Active
  - State-replication considerations
- Horizontally scaled
  - GTM – DNS based approach
  - L3ECMP (routed)
HA – active/passive
HA – active/passive - failover

• Connections terminated:
  • Stateless secondary
    – Secondary won't know which server to send packets to
    – TCP sessions will timeout and a new session initiated
    – Failover in the order of seconds (Thumb suck: 3-20s)
HA – active/passive failover with replication

• Connections work:
  • Secondary knows the hash state
  • Packets lost retransmitted
Active / Passive

- Active-passive failover requires a mechanism
- Could use:
  - VRRP (Virtual Router Redundancy Protocol)
  - CARP (Common Address Redundancy Protocol)
- If failover is not coordinated with load-balancer-health, a failed load-balancer may remain active (coordination problem).
- If state is not replicated between load balancers, failover will not account for existing connections (not a problem for short-lived connections with no affinity)
Active / Passive Cont

- Affinity can be preserved with a Cookie
- LVS (linux virtual server) can do state-replication (using a kernel module)
- State-replication doesn't help with scaling performance-wise (at all)
Active/Active – How?

• Need a mechanism to distribute requests to multiple front end load-balancers. In effect, a load balancer for your load balancers.

• HOW?
  
  • DNS e.g. each LB has a separate ip address associated with resources it's load-balancing
    - Return one or more resource records either randomly or on some externally instrumented basis.
    - Fail load balancers in or out using health check or manually
  
  • L2 or L3 stateless plus sticky mechanism.
Turtles all the way...

• When do we stop?
Active/Active – Stateful vs Not

- Stateful is typically done by clusters of commercial load-balancers. State replication can be expensive and imperfect.
  - At scale, can be extremely expensive
  - Memory on cluster members and bandwidth/cpu for replication is the limiting factor for state and connections per section.

- Stateless
  - In the DNS case resource records for a failed LB have to time out of caches before that LB stops being used.
  - In the L3-ECMP case a failure will cause some fraction of connections to rehash across other load-balancers anywhere from a quarter to half (they will then be rendered out of state and lost).
Our Exercise - HAPerxy

- We're going to deploy HAPerxy to load-balance connections to two http servers.
- HAPerxy can do L4 (any TCP) or L7 (HTTP) load balancing
- We're going to do L7, this allows us to access http related features, including for example including a cookie.
HAProxy vs NGINX

- **L4 vs L7**
  - HAProxy can load balance anything over TCP or do L7.
  - NGINX is L7 only (HTTP(s) and IMAP/POP3).

- **SSL**
  - HAProxy doesn't support (can't only treat as TCP)
  - NGINX does, so cookies for example can be parsed, can be used for SSL offload etc.

- **Model**
  - HAProxy is threaded, effectively allowing it to engage multiple cpus in the activity.
  - NGINX uses an event driven single threaded model.
  - Both have merit, HAProxy is probably more scalable.
Goals

1) Install and perform a basic configuration of HAProxy.

2) Configure two additional webserver instances on alternate ports.

3) Demonstrate load-balanced-http connections between them.

4) Log X-Forwarded-For.

5) Bonus: use a cookie to pin a requesting host to one server or another.

6) Bonus: Remove failing servers from LB pool.
Bootstrap..

- `$ pkg install git`
- `$ git clone https://github.com/afnog/sse.git`
- `$ cd sse/loadbalancing/exercises/loadbalancer/`
- `$ Follow the trail by changing dirs and reading readmes`
  - `$ cd 01...`
  - `$ cat Readme.md`
- `Or browse github: https://github.com/afnog/sse/tree/loadbalancing/loadbalancer`
Bibliography

- HAPerxy - http://haproxy.1wt.eu/
- NGNIX - http://wiki.nginx.org/Main