Stateless Datacenter Load Balancing with Beamer

Vladimir Olteanu, Alexandru Agache, Andrei Voinescu, Costin Raiciu
University Politehnica of Bucharest

Thanks to SSICLOPS
Datacenter load balancing
Datacenter load balancing today
Strawman approach
Strawman approach

- Adding/removing servers breaks connection affinity
Load balancers use state to ensure connection affinity

**MUX**

<table>
<thead>
<tr>
<th>Flow</th>
<th>DIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Server₁**

**Server₂**
Load balancers use state to ensure connection affinity

- Only new connections are hashed
Load balancers use state to ensure connection affinity.
Load balancers use state to ensure connection affinity

• Scaling mux pool may reset some connections
Load balancers use state to ensure connection affinity.
SYN floods use up state memory

<table>
<thead>
<tr>
<th>Flow</th>
<th>DIP</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

MUX

Server\(_1\)

Server\(_2\)
SYN floods use up state memory
SYN floods use up state memory

- Back to the straw man approach
Stateful designs don’t guarantee connection affinity
Beamer: **stateless** load balancing

Beamer muxes do not keep per-connection state; each packet is forwarded independently.

When the target server changes, connections may break.

Beamer uses state stored in servers to redirect stray packets.
Beamer daisy chaining

- Used when reassigning traffic
Beamer daisy chaining

• Used when reassigning traffic

Want to POWER OFF
• Daisy-chained connections die off in time
### Balancing packets in Beamer

Which hashing algorithm is best?

<table>
<thead>
<tr>
<th></th>
<th>Low churn</th>
<th>Good load balancing</th>
<th>Few rules in dataplane</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ECMP</strong></td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Consistent Hashing</strong></td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td><strong>Maglev Hashing</strong></td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
</tr>
</tbody>
</table>
Beamer hashing

Indirection layer
Pick number of buckets B > N, number of servers

Mux dataplane:
• Assign each bucket to one server
• Bucket-to-server mappings known by all muxes
• Maintained by a centralized controller

Mux algorithm:
• Hash each packet modulo B
• Send to corresponding server
Beamer at work

<table>
<thead>
<tr>
<th>Bkt</th>
<th>DIP</th>
<th>PDIP</th>
<th>TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

MUX

Server₁

Server₂
### Beamer at work

<table>
<thead>
<tr>
<th>Bkt</th>
<th>DIP</th>
<th>PDIP</th>
<th>TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>t</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>1</td>
<td>t</td>
</tr>
</tbody>
</table>

- Packets contain previous server and time of reassignment
Beamer at work

- New connections are handled locally
Daisy chained connections die off in time
Benefits of Beamer muxes

Less memory usage and cache thrashing

Implementable in hardware: P4

Interchangeable

Resilient to SYN flood attacks

Cost: 16B encapsulation overhead per packet
Beamer mux performance

• Software implementation on top of netmap
• Machine: Xeon E5-2697 v2 @ 2.70GHz, Intel 82599 NIC

• Compared against:
  – Stateful – similar performance to Google’s Maglev [NSDI’16]
Single mux performance

![Graph showing Single mux performance](image)
Realistic traffic

HTTP traffic from recent MAWI trace
• Packets replayed back-to-back

36Gbps of upstream traffic on 7 cores
• 15 times more downstream traffic: 540Gbps

Rough estimate: 50-500 servers/mux
• Assuming servers source 1-10Gbps of traffic
Testbed evaluation

• 20 machines
  – 10Gbps NICs

• IBM RackSwitch 8264 as border router

• Software muxes
  – P4 reference implementation also used
Adding and removing muxes

- Mux failures and churn are handled smoothly
Adding servers

- Beamer spreads traffic evenly across servers
Connection affinity under SYN flood attacks

1Mpps SYN flood
2 muxes, 8 servers, 700 running connections
Drain servers during SYN flood

<table>
<thead>
<tr>
<th>DIPs Drained</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stateful</td>
<td>0</td>
<td>87±2</td>
<td>148±8</td>
<td>351±21</td>
</tr>
<tr>
<td>Beamer</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Control plane

- A centralized fault-tolerant **controller** manages the dataplane
Control plane

- ZooKeeper Cluster
- Controller
- MUX v1
- MUX v1
- MUX v1
- MUX v1
- DIP
- PDIP
- TS

<table>
<thead>
<tr>
<th>DIP</th>
<th>PDIP</th>
<th>TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>D3</td>
<td>TS1</td>
</tr>
<tr>
<td>D2</td>
<td>D4</td>
<td>TS2</td>
</tr>
</tbody>
</table>
|     |      |...

...
Control plane

- Muxes download update
- Daisy chaining allows for temporarily stale muxes
Control plane experiments

- Tested on Amazon EC2
- 3 ZooKeeper daemons, 100 muxes
- Large simulated service: 64K servers, 6.4M buckets
- Stress-tested controller
Control plane experiments

When adding 32,000 servers:

• Controller takes 1-10s to update ZooKeeper

• Muxes take 0.5-6s to get new dataplane information

• Total control traffic: 1GB (10MB/mux)
Please see paper for:

• MPTCP support in Beamer

• Minimizing # of rules required in muxes
  – 1 rule / server, rather than 1 rule / bucket

• Avoiding reset connections in corner cases
Conclusions

• Stateless load balancing using daisy chaining

• 36Gbps of HTTP traffic on 7 cores
  – 540Gbps of downlink traffic

• Scalable, fault tolerant control plane

• Beamer is open-source: https://github.com/Beamer-LB