CHEETAH: A High-Speed Load-Balancer Design with Guaranteed Per-Connection-Consistency

Tom Barbette, Chen Tang, Haoran Yao, Gerald Q. Maguire Jr, Dejan Kostic, Panagiotis Papadimitratos & Marco Chiesa
Load-balancer
Load-balancer

Per-Connection-Consistency (PCC)
Load-balancer

Per-Connection-Consistency (PCC)
Load-balancer

Per-Connection-Consistency (PCC)

This site can’t be reached

The connection was reset.

Try:
- Checking the connection
- Checking the proxy and firewall
- Running Windows Networking Diagnostics

ERR_CONNECTION_RESET

This webpage is not available

ERR_CONNECTION_TIMED_OUT

Details
Load-balancer

Per-Connection-Consistency (PCC)

Uniform Load Balancing

LB
Load-balancer

Per-Connection-Consistency (PCC)

Efficient

Uniform Load Balancing

LB
Load-balancer

- Per-Connection-Consistency (PCC)
- Efficient
- Dynamicity

Uniform Load Balancing
The challenge: Ensuring PCC

For each packet of an existing connection, the LB asks itself:

« Which server is handling this connection? »

Per-Connection-Consistency
Today’s solutions cannot ensure all requirements at the same time.
Stateless solutions
ECMP, WCMP [EuroSys’14]
Stateless solutions

Consistent hashing [STOC’97], Beamer [NSDI’18], Faild [NSDI’18]
Stateful solutions
Silkroad [SIGCOMM’17], Ananta [SIGCOMM’13], Maglev [NSDI’16], Katran

Uniform

Efficient

PCC

Dynamic
CHEETAH

PCC « Which server is handling this connection? »

CHEETAH: ask the user to remember for us
key idea: store information about the load balancing decisions into a cookie

support any realizable load-balancing logic

high resilience

amenable to simple and fast implementation
CHEETAH IMPLEMENTATIONS

- 100G with 4 cores
  FastClick

- 12.8 Tb/s
  P4_Tofino

- P4_16

→ **5x faster** software processing time **while guaranteeing** PCC

→ **twice** better tail latency compared to hashing thanks to more uniform load spreading

2020-03-09
CHEETAH
CHEETAH Overview

Users

Load-Balancer

LB-logic

selected server 2

Servers
Cheetah Overview

Load-Balancer

Users

Servers

LB-logic

set cookie

get server id

selected server 2

2020-03-09
**CHEETAH Overview**

**Where to store the cookie?**
- TCP timestamp
- QUIC connection ID
- IPv6 address

---

*Flowchart*

1. **LB-logic**
   - set cookie
   - get server id
   - extract server id
   - selected server 2

2. **Users**
   - 2

3. **Load-Balancer**
   - 2

4. **Servers**
   - Server 2

---

**Overview**

- 2020-03-09
Allows attackers to target a server...

Users

Load-Balancer

Selected server 2

Get server id

Set cookie

Extract server id

Was server 2

Servers

Overview

2020-03-09
Stateless Cheetah

Set cookie:
cookie = id XOR hash(4-tuple)

Get server id:

Extract server id:
id = cookie XOR hash (4-tuple)

Load-Balancer

Selected server

Previous server

Servers

Users

KTH

Load-Balancer
Stateless CHEETAH

Load-Balancer

- Set cookie: $\text{cookie} = \text{id} \ XOR \ \text{hash(4-tuple)}$
- Extract server id: $\text{id} = \text{cookie} \ XOR \ \text{hash(4-tuple)}$

LB-logic

- Selected server
- Get server id

Users

- AE7B

Servers

- AE7B

2020-03-09
**Stateless CHEETAH**

- **Users**
  - **LB-logic**:
    - set cookie:
      - `cookie = id XOR hash(4-tuple)`
    - extract server id:
      - `id = cookie XOR hash(4-tuple)`

- **Load-Balancer**:
  - selected server

- **Servers**
  - previous server
Stateless Cheetah

The server can directly set up the cookie to enable Direct Server Return (DSR).

Users

Load-Balancer

Servers

LB-logic

set cookie
cookie = id XOR hash(4-tuple)

get server id

The server can directly set up the cookie to enable Direct Server Return (DSR).

Load-Balancer

LB-logic

selected server

set cookie
cookie = id XOR hash(4-tuple)

get server id

The server can directly set up the cookie to enable Direct Server Return (DSR).

Load-Balancer

LB-logic

selected server

set cookie
cookie = id XOR hash(4-tuple)

get server id

The server can directly set up the cookie to enable Direct Server Return (DSR).
There are two CHEETAHS

Stateless CHEETAH

Stateful CHEETAH

→ Keep per-connection state on the LB

NAT
Statistics
Rate limiter
...

https://www.goodfreephotos.com
Stateful CHEETAH

Flow index

O(1) lookup

Per-connection state table

<table>
<thead>
<tr>
<th>Server ID</th>
<th>NAT</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIP_3</td>
<td>1.2.3.4:5678</td>
<td>9 packets</td>
</tr>
<tr>
<td>DIP_6</td>
<td>9.0.1.2:3456</td>
<td>90 packets</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Stateful CHEETAH

Flow index

Fast in software

Entirely doable from hardware dataplane

O(1) lookup

Per-connection state table

<table>
<thead>
<tr>
<th>Server ID</th>
<th>NAT</th>
<th>Statistics</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIP_3</td>
<td>1.2.3.4:5678</td>
<td>9 packets</td>
</tr>
<tr>
<td>DIP_6</td>
<td>9.0.1.2:3456</td>
<td>90 packets</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Stack of Empty indexes

1AB2
39F0
...

O(1) insertion

O(1) deletion

cookie
EVALUATION

Stateful at the price of stateless
Packet processing performance analysis

Requests for a 8K file to 64 servers using HTTP, 640 000 req/s
Requests for a 8K file to 64 servers using HTTP, 640,000 req/s
Packet processing performance analysis

We have the advantages of stateful classification at the price of stateless:

- PCC
- Dynamicity
- Uniformity

2020-03-09

Requests for a 8K file to 64 servers using HTTP, 640 000 req/s
Packet processing performance analysis

2 to 3 times faster classification than a cuckoo hash table

Requests for a 8K file to 64 servers using HTTP, 640 000 req/s
EVALUATION

Reducing load imbalances with arbitrary LB mechanisms
Uniform workload:
Variance across server load with hashing

20-30% variance with medium to high load

In line with Maglev [NSDI’16] which shows up to a ~30% overprovision

Requests for a 8K file using HTTP to 64 servers, at 100 to 200 req/s per servers
Uniform workload:
Variance across server load with Round-Robin (RR)

By using round-robin, CHEETAH can bring down the variance to reach a near-uniform load balancing.
Uniform workload:
Tail latency

**CHEETAH** lowers the tail latency by 2 to 3x

Requests for a 8K file using HTTP to 64 servers, at 100 to 200 req/s per servers
Bimodal workload:

Round-Robin (RR) does not help

100req/s per servers with 10% of heavy request and 90% of small requests
Bimodal workload:
Least loaded

100req/s per servers with 10% of heavy request and 90% of small requests
Bimodal workload:
Average Weighted Round-Robin and Power-of-2-choice

CHEETAH allows to use your preferred advanced server selection techniques

a factor of ~2X

100req/s per servers with 10% of heavy request and 90% of small requests
In the paper

- Large-scale simulations
  - 468 servers, motivation, # broken connections when ensuring uniformity with hashing

- More experiments
  - PCC w/ dynamicity, comparison with Beamer [NSDI’18]

- Details about the cookie encoding and limitations
  - Other possible cookie encodings

- Multi-tier load-balancing considerations
  - Stateful load balancers still break connections at scale!
 Exploited a network cookie to:

- Guarantee PCC and support any implementable LB mechanism
- Present a fast design to allow $O(1)$ flow insertion and lookup from the dataplane

Cookie as a standard?

Implemented on both software switches and programmable Tofino ASIC

Thank you!
Backup slides
TESTBED
Uniform workload experiment

64 servers

Machine 1
Machine 2
Machine 3
Machine 4
Machine 5
CPU 1
CPU 2
CPU 3
CPU 4
CPU 5
CPU 6
CPU 7
CPU 8
CPU 9
CPU 10
CPU 11
CPU 12
CPU 13
CPU 14
CPU 15
CPU 16

LB

SRIOV

8K HTTP requests

Machine 6
Machine 7
Machine 8

2020-03-09
FULL STATEFUL DESIGN
**CHEETAH stateful load balancer**

Client-side

- IP src, VIP
- TCP src, dst
- payload

Server-side

- LB-logic
- hash_s(c)
- ConnStack[0]
  - cookie
    - 1D15
    - ...
    - ...
  - hash_s(c) % m
- ConnStack[m-1]
  - cookie
    - 1A2B
    - 39EF
    - ...
- ConnTable[0]
  - id
  - hash
  - DIP
  - ...
  - 39EF FEFE DIP_6
  - 39F0
  - 39F1 A7A7 DIP_4
  - ...
  - ...

- ConnTable[m-1]
  - id
  - hash
  - DIP
  - ...
  - ...
  - 39EF - DIP_1
  - 39F0 3434 DIP_3
  - 39F1 5656 DIP_3
  - ...
  - ...

Client-side logic:
- IP src, VIP
- TCP src, dst
- payload

Server-side logic:
- LB-logic
- hash_s(c)
CHEETAH stateful load balancer

**ConnTable[0]**
- id
- hash
- DIP
- ...
- ...
- ...
- 39EF
- FFEF
- DIP_6
- 39F0
- 11D7
- DIP_3
- 39F1
- A7A7
- DIP_4
- ...
- ...
- ...

**ConnTable[m-1]**
- id
- hash
- DIP
- ...
- ...
- ...
- 39EF
- -
- -
- 39F0
- 3434
- DIP_1
- 39F1
- 5656
- DIP_3
- ...
- ...
- ...

**ConnStack[0]**
- cookie
- 1D15
- ...
- ...
- ...

**ConnStack[m-1]**
- cookie
- 1A2B
- 39EF
- ...

**LB-logic**
- DIP_3
- 11D7

**hash₃(c)**
- 39F0

**hash₃(c) % m**
- ...

IP src, VIP
TCP src, dst
payload

IP src, DIP_3
TCP src, dst
cookie q
payload

client-side

server-side
TS PARSING
Implementation: storing the cookie

Where to store the cookie?

- **TCP timestamp**, QUIC connection-id, IPv6 addresses
- quickly extracting the TCP timestamp is key to high performance

<table>
<thead>
<tr>
<th>SYN packets</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MSS SACKOK Timestamp [NOP WScale]</td>
<td>49.86%</td>
</tr>
<tr>
<td>MSS NOP WScale NOP NOP Timestamp [SACK EOL]</td>
<td>44.49%</td>
</tr>
<tr>
<td>MSS NOP WScale SACKOK Timestamp</td>
<td>4.53%</td>
</tr>
<tr>
<td>Slow path</td>
<td>1.12%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SYN-ACK packets</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MSS SACKOK Timestamp [NOP WScale]</td>
<td>76.85%</td>
</tr>
<tr>
<td>MSS NOP WScale SACKOK Timestamp</td>
<td>18.79%</td>
</tr>
<tr>
<td>MSS NOP NOP Timestamp [SACK EOL]</td>
<td>1.69%</td>
</tr>
<tr>
<td>MSS NOP WScale NOP NOP Timestamp [SACK EOL]</td>
<td>1.55%</td>
</tr>
<tr>
<td>Slow path</td>
<td>1.12%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Other packets</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>NOP NOP Timestamp</td>
<td>98.46%</td>
</tr>
<tr>
<td>NOP NOP Timestamp [NOP NOP SACK]</td>
<td>1.49%</td>
</tr>
<tr>
<td>Slow path</td>
<td>0.05%</td>
</tr>
</tbody>
</table>

Based on a 1-hour KTH packet trace