Routing Performance Testing and Measurements

Shash Reddy (Pivotal) & Swetha Repakula (IBM)
Agenda

Overview of Cloudfoundry
Motivation & Goal
Analyzing Performance
Improving Performance
Key Takeaways
Q & A
Overview of Cloud Foundry
Simplified Architecture

cf push test-app

LB

GOROUTER

CC API

BBS

ROUTE Emitter

NATS

CELL

AUCTIONEER
Gorouter passes the request to CC
Simplified Architecture

CC tells Diego to Deploy the App

BBS tells the scheduler to place the work on a cell
Simplified Architecture

- **Route emitter** listens for changes from BBS and notifies Gorouter of new routes.

**Diagram:**
- LB
- Gorouter
- CC API
- BBS
- Auctioneer
- NATs
- CELL
- Route Emitter
Simplified Control Plane view

GET

\[ \text{test-app.cf-app.com} \]
Simplified Control Plane view

Gorouter passes the request to the corresponding cell
Simplified Control Plane view

Response is sent back to the Gorouter
Simplified Control Plane view

Gorouter passes the request back to the client through the load balancer
Why did we start this work?

- Gorouter is in the critical path for all requests to applications.
- SAP submitted a Github issue to point out a performance regression introduced by a commit.
What should be the outcome?

1. Better inform operators how to scale routers
2. Measure and Improve the Gorouter Performance
3. Catch Regressions

Me trying to reach my life goals
Analyzing Router Performance
Agile feedback loop

Learn → Implement → Measure
Basic Performance testing

- How to test Router?
- What type of load for test?
- What measurements should we focus on?
Router in Cloudfoundry

Client → LB

APP Container

Cell
Basic Performance testing
Improved Performance Testing

- Altered our load generating client to give us more information (start and end times of each request)
- Wanted to understand performance of router under different loads.
- Started monitoring CPU during load ramp up.
- Cockroft’s Headroom plot
Headroom plot
Sample perf report

Headroom plot

Throughput (requests/sec)

Latency (sec)

Throughput

Mean CPU percentage over time

CPU percentage

Time since benchmark started (seconds)
Reformed Performance Improvement Goal

- Reduce the inflection point.
- Improve throughput by 3x.
Improving Gorouter Performance
Where to start?

- Analyze the resource usage to identify any bottlenecks
- How to interpret the results?
Where to start?

- Analyze the resource usage to identify any bottlenecks

How to interpret the results?

```go
go tool pprof --seconds=60 http://localhost:8080/debug/pprof/profile
```

Fetching profile from http://localhost:8080/debug/pprof/profile?seconds=60

Please wait... (1m0s)

Entering interactive mode (type "help" for commands)

(pprof) top

41.50s of 85.59s total (48.49%)
Dropped 668 nodes (cum <= 0.43s)
Showing top 10 nodes out of 266 (cum >= 1.18s)

<table>
<thead>
<tr>
<th>flat</th>
<th>flat%</th>
<th>sum%</th>
<th>cum</th>
<th>cum%</th>
</tr>
</thead>
<tbody>
<tr>
<td>12.38s</td>
<td>14.46%</td>
<td>14.46%</td>
<td>19.84s</td>
<td>23.18%</td>
</tr>
<tr>
<td>6.42s</td>
<td>7.50%</td>
<td>21.97%</td>
<td>6.78s</td>
<td>7.92%</td>
</tr>
<tr>
<td>5.67s</td>
<td>6.62%</td>
<td>28.59%</td>
<td>6.21s</td>
<td>7.26%</td>
</tr>
<tr>
<td>3.91s</td>
<td>4.57%</td>
<td>33.16%</td>
<td>3.91s</td>
<td>4.57%</td>
</tr>
<tr>
<td>3.45s</td>
<td>4.03%</td>
<td>37.19%</td>
<td>3.45s</td>
<td>4.03%</td>
</tr>
<tr>
<td>2.53s</td>
<td>2.96%</td>
<td>40.14%</td>
<td>2.53s</td>
<td>2.96%</td>
</tr>
<tr>
<td>2.50s</td>
<td>2.92%</td>
<td>43.07%</td>
<td>18.77s</td>
<td>21.93%</td>
</tr>
<tr>
<td>2.23s</td>
<td>2.61%</td>
<td>45.67%</td>
<td>3.63s</td>
<td>4.24%</td>
</tr>
<tr>
<td>1.24s</td>
<td>1.45%</td>
<td>47.12%</td>
<td>1.24s</td>
<td>1.45%</td>
</tr>
<tr>
<td>1.17s</td>
<td>1.37%</td>
<td>48.49%</td>
<td>1.18s</td>
<td>1.38%</td>
</tr>
</tbody>
</table>
Analyzing Memory Allocation

- Large temporal object creations in critical path
- Buffer Pools
Breaking It Down

- Core Gorouter
- Logging
- Metrics
- Syslog Streaming
Other Performance Improvements

- Go 1.7
- Refactoring & Breaking down monolithic structure of packages.
- Keep-alive connections
Did we meet our goals?
1. Better inform operators how to scale routers
1. Better inform operators how to scale routers
2. **Measure and Improve the Gorouter Performance**
   a. *We wanted to reduce the inflection point with high throughput without increasing latency*
   b. *Narrow down the performance scope to improving throughput by 3x*
1. Better inform operators how to scale routers
2. Measure and Improve the Gorouter Performance
3. Catch Regressions
THINK OF PERFORMANCE AS A FIRST CLASS CITIZEN
Q & A
References

- https://github.com/rakyll/hey [hey load generator client]
- https://github.com/adrianco/headroom-plot [headroom plot]