Why you might need to store your metrics?

Most common cases:

- Capacity planning
- Troubleshooting and Postmortems
- Visualization of business data
- And more...
Graphite and its modular architecture

Graphite does three things:
- Kick ass.
- Chew bubblegum.
- Make it easy to store and graph metrics.
  (And it’s all out of bubblegum.)

From the graphiteapp.org

- Allows to store time-series data
- Easy to use — text protocol and HTTP API
- You can create any data flow you want
- Modular — you can replace any part of it
Breaking graphite: our problems at scale

What’s wrong with this schema?

- carbon-relay — SPOF
- Hard to scale
- Data is different after failures
- Render time increases with more servers
Replacing carbon-relay

User Requests

LoadBalancer

graphite-web

<table>
<thead>
<tr>
<th>carbon-cache</th>
<th>graphite-web</th>
</tr>
</thead>
<tbody>
<tr>
<td>Store1</td>
<td>Store2</td>
</tr>
</tbody>
</table>

DC1

carbon-c-relay

DC2

carbon-c-relay

carbon-c-relay

Metrics

Servers, Apps, etc

Server
Replacing carbon-relay

carbon-c-relay:

- Written in C
- Routes 1M data points per second using only 2 cores
- L7 LB for graphite line protocol (RR with sticking)
- Can do aggregations
- Buffers the data if upstream is unavailable
Query: target=sys.server.cpu.user

Result:

- Node 1:
  - t0: [V V V]
  - t1: [V V V]

- Node 2:
  - t0: [V V V V V]
  - t1: [V V V V V]

- Zipped metric:
  - t0: [V V V V V]
  - t1: [V V V V V]
Zipper stack: architecture
Zipper stack: results

- Written in **Go**
- Can query store servers in **parallel**
- Can ”Zip” the data
- `carbonzipper ⇔ carbonserver — 2700 RPS`
  `graphite-web ⇔ carbon-cache — 80 RPS.`
- `carbonserver` is now part of `go-carbon` (since December 2016)
Metric distribution: how it works

Up to **20%** difference in worst case
Metric distribution: jump hash

[Graph showing metrics received for network dc1]

Rewriting Frontend in Go: carbonapi
Rewriting Frontend in Go: result

- Significantly reduced response time for users (15s ⇒ 0.8s)
- Allows more complex queries because it’s faster
- Easier to implement new heavy math functions
- Also available as Go library
Replication techniques and their pros and cons

Replication Factor 2

a, h
b, c
c, a
d, e
e, f
f, d
g, b
h, g

Replication Factor 2
Replication techniques and their pros and cons

Replication Factor 1

a,e  

b,f  

c,g  

d,h  

a,e  

b,f  

c,g  

d,h
Replication techniques and their pros and cons

Replication Factor 1, randomized

- a, e
- c, g
- b, f
- d, h

- a, g
- h, e
- c, f
- b, d
Replication techniques and their pros and cons

Comparison of amount of lost data in worst case for different schemas for 8 servers
Replication techniques and their pros and cons

Comparison of probability to lose data for different schemas for 8 servers

<table>
<thead>
<tr>
<th>Servers lost</th>
<th>Replication Factor 2</th>
<th>Replication Factor 1, randomized</th>
<th>Replication Factor 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>100.0%</td>
<td>57.3%</td>
<td>14.3%</td>
</tr>
<tr>
<td>3</td>
<td>100.0%</td>
<td>85.6%</td>
<td>43.1%</td>
</tr>
<tr>
<td>4</td>
<td>100.0%</td>
<td>97.3%</td>
<td>77.1%</td>
</tr>
<tr>
<td>5</td>
<td>100.0% 100.0% 100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
</tr>
</tbody>
</table>
Our current setup

- **32** Frontend Servers
- **400** RPS on Frontend
- **40k** Metric Requests per second
- **11 Gbps** traffic on the backend
- **200** Store servers in 2 DCs
- **2.5M** unique metrics per second (**10M** hitting stores)
- **130 TB** of Metrics in total
- Replaced all the components
What’s next?

- Metadata search (in progress)
- Find a replacement for Whisper (in progress)
- Rethink aggregators
- Replace graphite line protocol between components
Example:

target = sum(virt.v1.*.dc:datacenter1.status:live.role:graphiteStore.text-match:metricsReceived)

- Separate tags stream and storage
- No history (yet)
- No negative match support (yet)
- Only ”and” syntax
- Just a few months old
Bonus 1: testing Clickhouse on a single server
It’s all Open Source!

- carbonzipper — github.com/dgryski/carbonzipper
- go-carbon — github.com/lomik/go-carbon
- carbonsearch — github.com/kanatohodets/carbonsearch
- carbonapi — github.com/dgryski/carbonapi
- carbon-c-relay — github.com/grobian/carbon-c-relay
- carbonmem — github.com/dgryski/carbonmem
- replication factor test — github.com/Civil/graphite-rf-test
Questions?

vladimir.smirnov@booking.com
What’s next?

Thanks!