

Graphite@Scale:

How to store million metrics per second

Booking.com

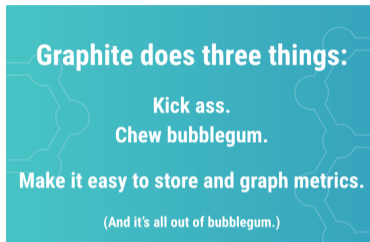
Vladimir Smirnov
System Administrator

Why you might need to store your metrics?

Most common cases:

- ▶ Capacity planning
- ▶ Troubleshooting and Postmortems
- ▶ Visualization of business data
- ▶ And more...

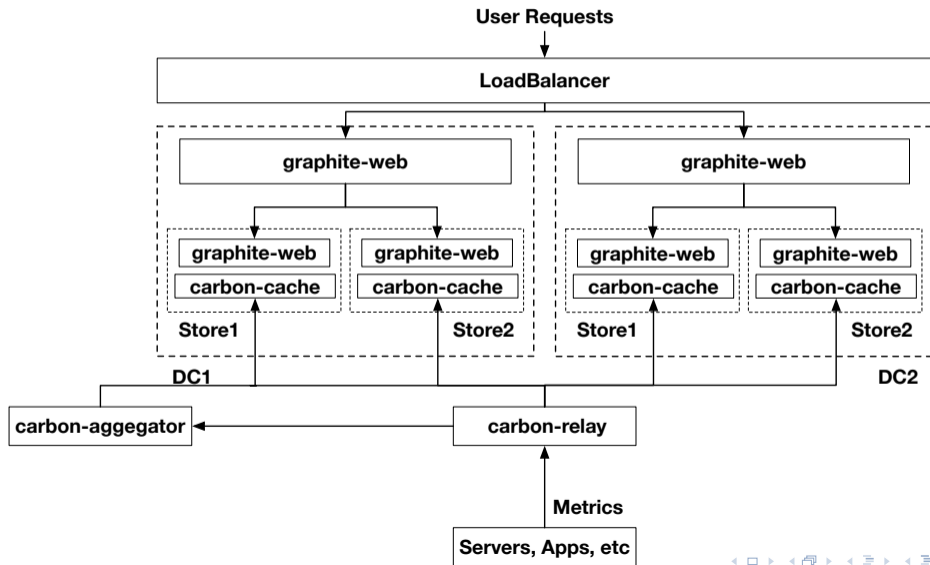
Graphite and its modular architecture



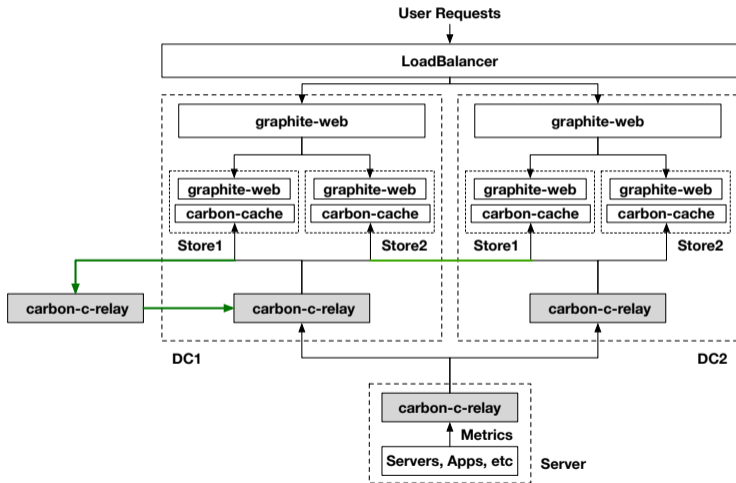
From the graphiteapp.org

- ▶ 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

Open Source stack



Replacing carbon-relay



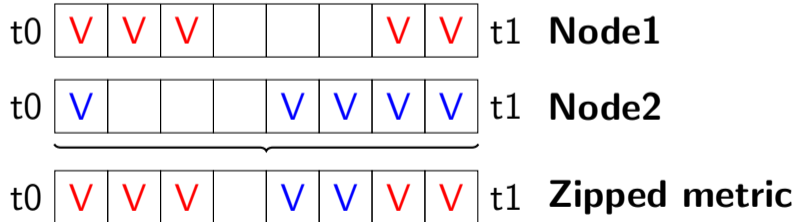
carbon-c-relay:

- ▶ 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

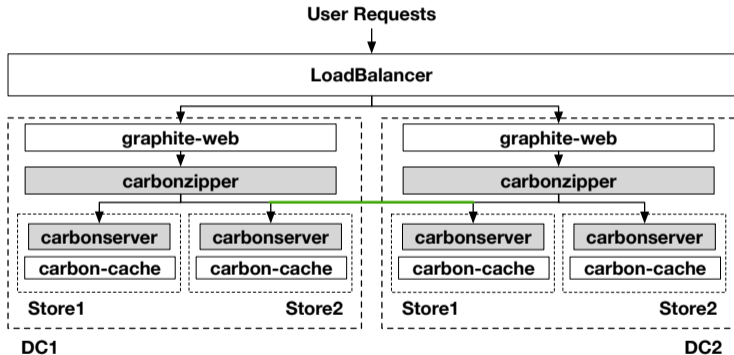
Zipper stack: Solution

Query: target=sys.server.cpu.user

Result:



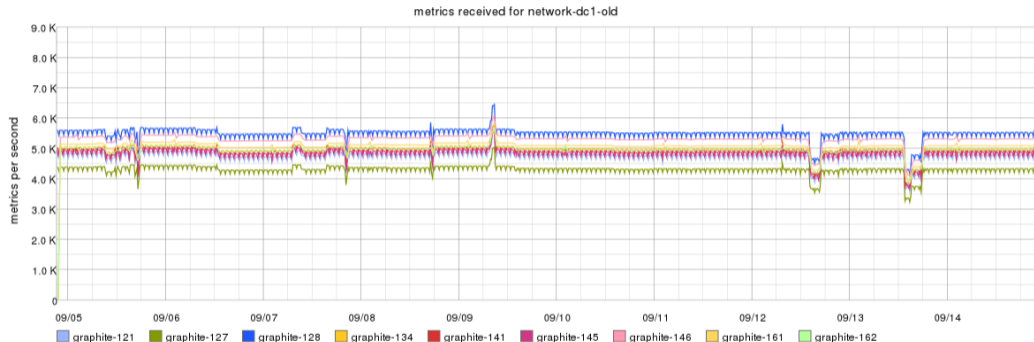
Zipper stack: architecture



Zipper stack: results

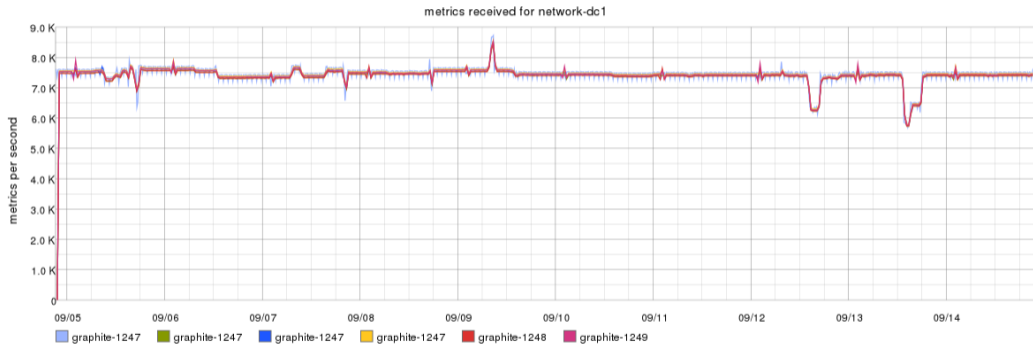
- ▶ Written in **Go**
- ▶ Can query store servers in **parallel**
- ▶ Can "Zip" the data
- ▶ carbonzipper \Leftrightarrow carbonserver — **2700** RPS
graphite-web \Leftrightarrow carbon-cache — **80** RPS.

Metric distribution: how it works

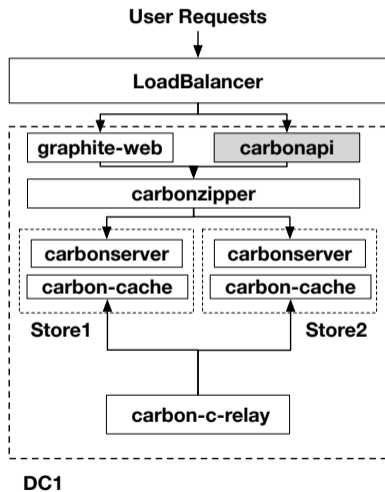


Up to **20%** difference in worst case

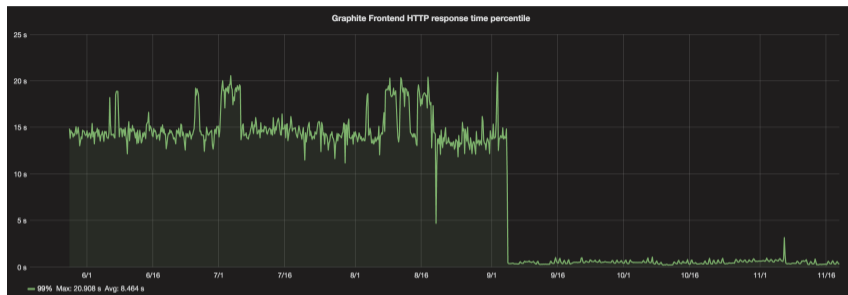
Metric distribution: jump hash



Rewriting Frontend in Go: carbonapi

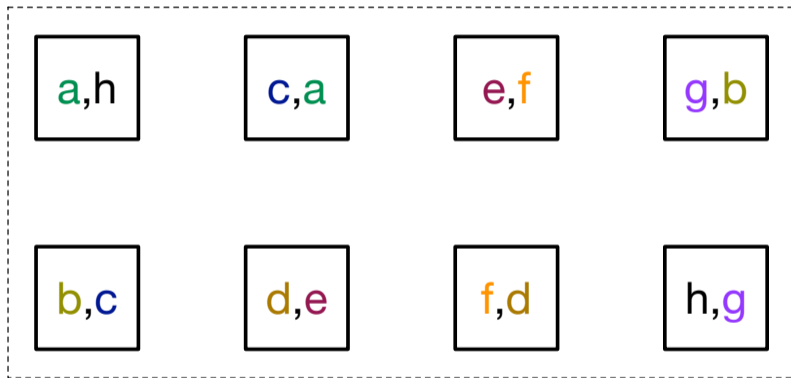


Rewriting Frontend in Go: result



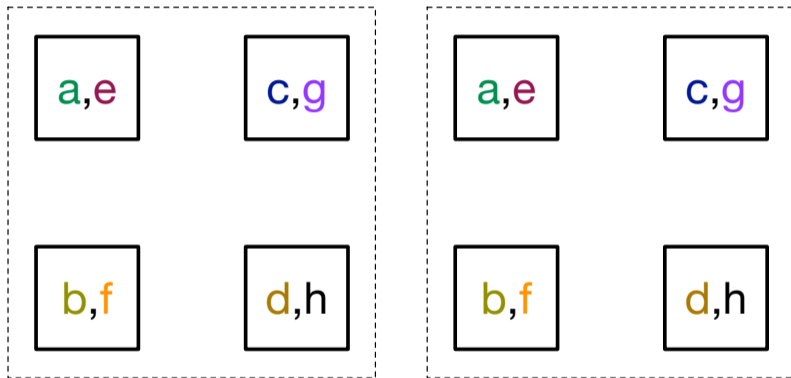
- ▶ Significantly reduced response time for users (**15s** \Rightarrow **0.8s**)
- ▶ Allows more complex queries because it's faster
- ▶ Easier to implement new heavy math functions
- ▶ Also available as Go library

Replication techniques and their pros and cons



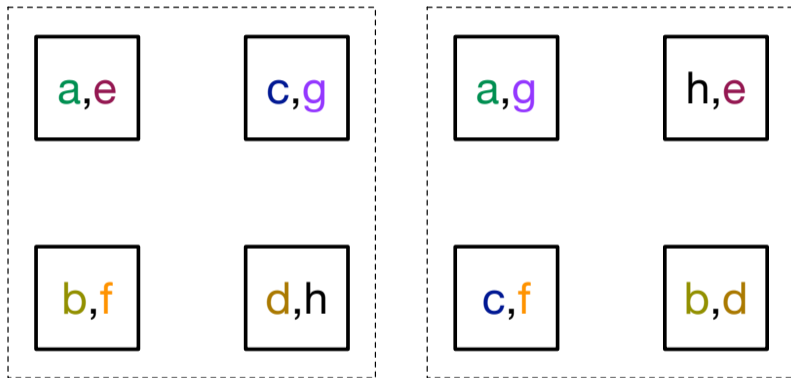
Replication Factor 2

Replication techniques and their pros and cons



Replication Factor 1

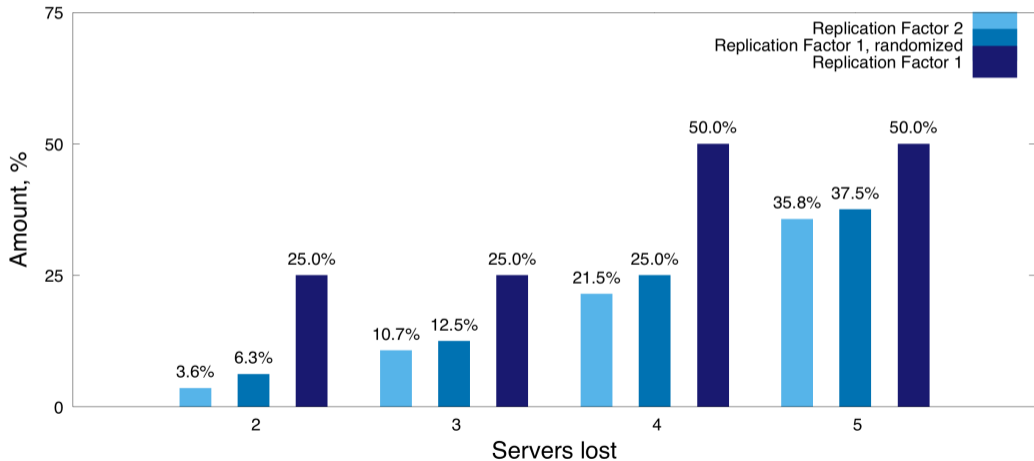
Replication techniques and their pros and cons



Replication Factor 1, randomized

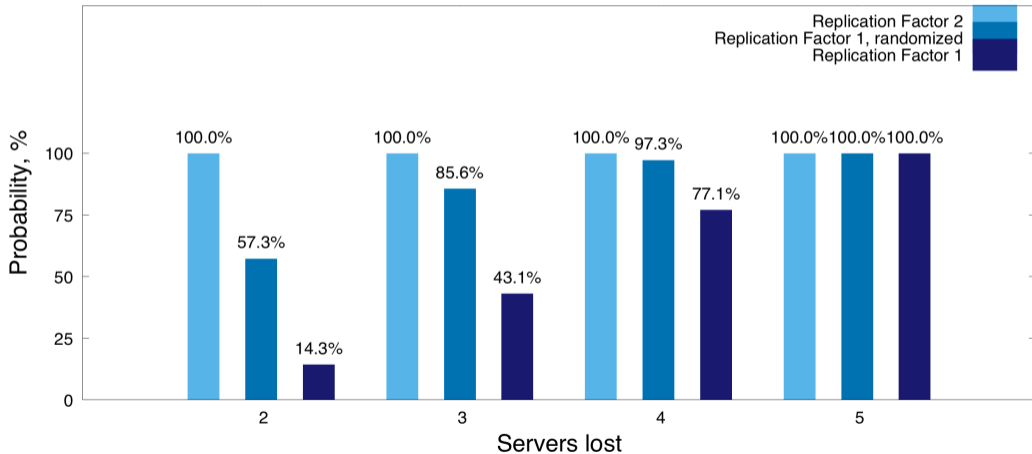
Replication techniques and their pros and cons

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



Our current setup

- ▶ **32** Frontend Servers
- ▶ **200** RPS on Frontend
- ▶ **30k** Metric Requests per second
- ▶ **11 Gbps** traffic on the backend
- ▶ **200** Store servers in 2 DCs
- ▶ **2M** unique metrics per second (**8M** hitting stores)
- ▶ **130 TB** of Metrics in total
- ▶ Replaced **all** the components*

* — except for carbon-cache

What's next?

- ▶ Metadata search (in progress)
- ▶ Solve problems with missing Cache (in progress)
- ▶ Find a replacement for Whisper
- ▶ Improve aggregators
- ▶ Replace graphite line protocol between components

It's all Open Source!

- ▶ carbonzipper — github.com/dgryski/carbonzipper
- ▶ carbonserver — github.com/grobian/carbonserver
- ▶ 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

Thanks!

We are hiring!
<https://workingatbooking.com>