Graphite@Scale:

How to store millions of metrics per second

Booking.com

Vladimir Smirnov
System Administrator

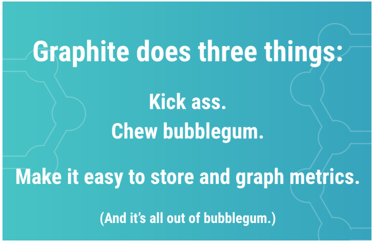
GrafanaCon EU 2018
1 March 2017

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.)

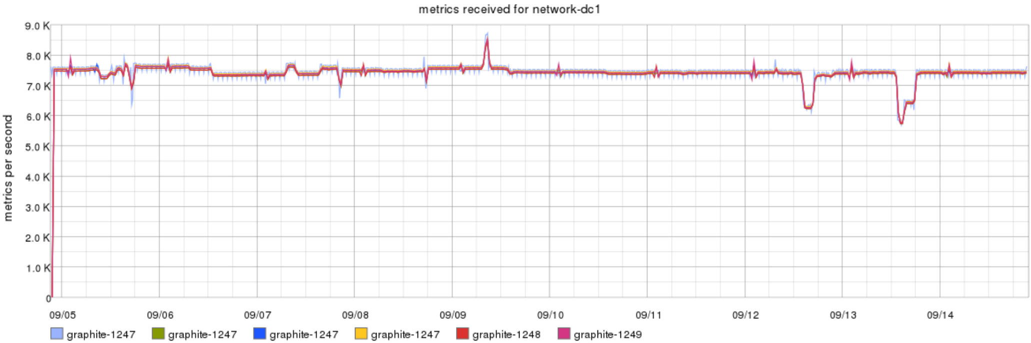
From the graphiteapp.org

- Allows to store time-series data
- Easy to use — text protocol and HTTP API

```
echo "metric.name 1.234 $(date +%s)" | nc host 2003
```

- Modular — you can replace any part of it

Graphite: Example



```
https://host/render?target=aliasByNode(carbon.*.metricsRecevied,1)
```

Our current setup

- **O(100)** Storage servers in multiple DCs
- **O(10)** of Frontend Servers
- **O(100)** TB of data in total
- **O(100 M)** unique metrics
- **O(10 M)** unique points per second
- **O(10 k)** RPS on Frontend
- **O(10 k)** of Individual Metric Requests per second
- **O(10 M)** points fetched from storage every second.

Our current setup

- **O(100)** Storage servers in multiple DCs
- **O(10)** of Frontend Servers
- **O(100)** TB of data in total
- **O(100 M)** unique metrics
- **O(10 M)** unique points per second
- **O(10 k)** RPS on Frontend
- **O(10 k)** of Individual Metric Requests per second
- **O(10 M)** points fetched from storage every second.

Our current setup

- **O(100)** Storage servers in multiple DCs
- **O(10)** of Frontend Servers
- **O(100)** TB of data in total
- **O(100 M)** unique metrics
- **O(10 M)** unique points per second
- **O(10 k)** RPS on Frontend
- **O(10 k)** of Individual Metric Requests per second
- **O(10 M)** points fetched from storage every second.

Our current setup

- **O(100)** Storage servers in multiple DCs
- **O(10)** of Frontend Servers
- **O(100)** TB of data in total
- **O(100 M)** unique metrics
- **O(10 M)** unique points per second
- **O(10 k)** RPS on Frontend
- **O(10 k)** of Individual Metric Requests per second
- **O(10 M)** points fetched from storage every second.

Our current setup

- **O(100)** Storage servers in multiple DCs
- **O(10)** of Frontend Servers
- **O(100)** TB of data in total
- **O(100 M)** unique metrics
- **O(10 M)** unique points per second
- **O(10 k)** RPS on Frontend
- **O(10 k)** of Individual Metric Requests per second
- **O(10 M)** points fetched from storage every second.

Our current setup

- **O(100)** Storage servers in multiple DCs
- **O(10)** of Frontend Servers
- **O(100)** TB of data in total
- **O(100 M)** unique metrics
- **O(10 M)** unique points per second
- **O(10 k)** RPS on Frontend
- **O(10 k)** of Individual Metric Requests per second
- **O(10 M)** points fetched from storage every second.

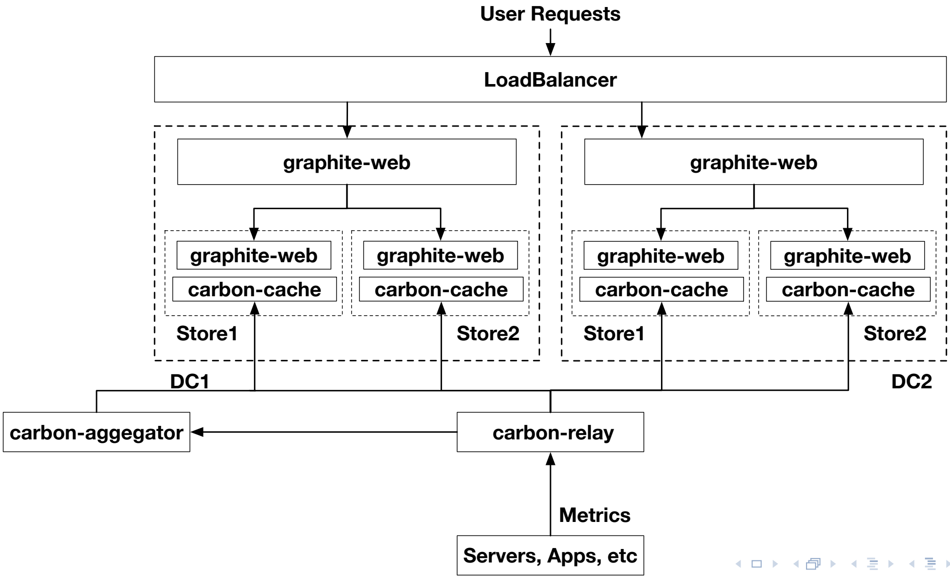
Our current setup

- **O(100)** Storage servers in multiple DCs
- **O(10)** of Frontend Servers
- **O(100)** TB of data in total
- **O(100 M)** unique metrics
- **O(10 M)** unique points per second
- **O(10 k)** RPS on Frontend
- **O(10 k)** of Individual Metric Requests per second
- **O(10 M)** points fetched from storage every second.

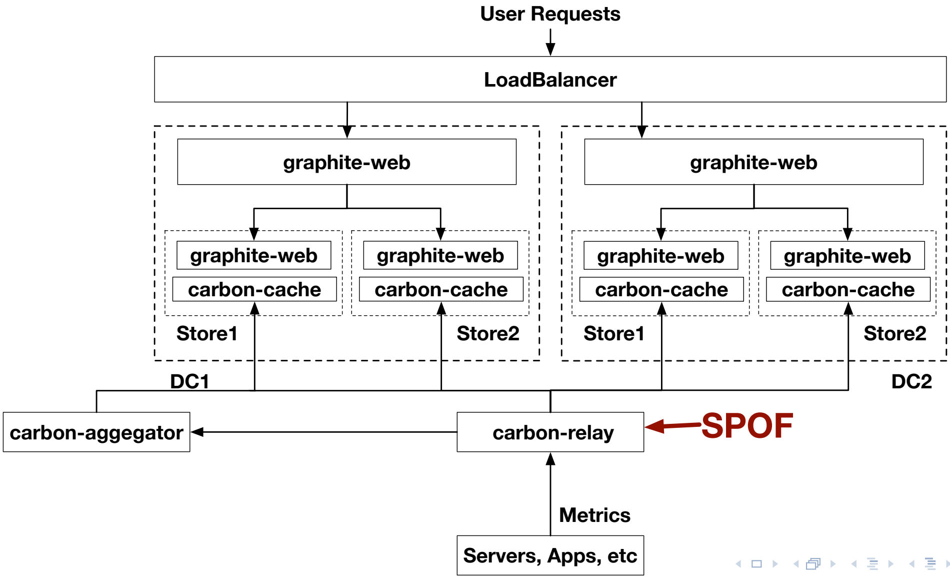
Our current setup

- **O(100)** Storage servers in multiple DCs
- **O(10)** of Frontend Servers
- **O(100)** TB of data in total
- **O(100 M)** unique metrics
- **O(10 M)** unique points per second
- **O(10 k)** RPS on Frontend
- **O(10 k)** of Individual Metric Requests per second
- **O(10 M)** points fetched from storage every second.

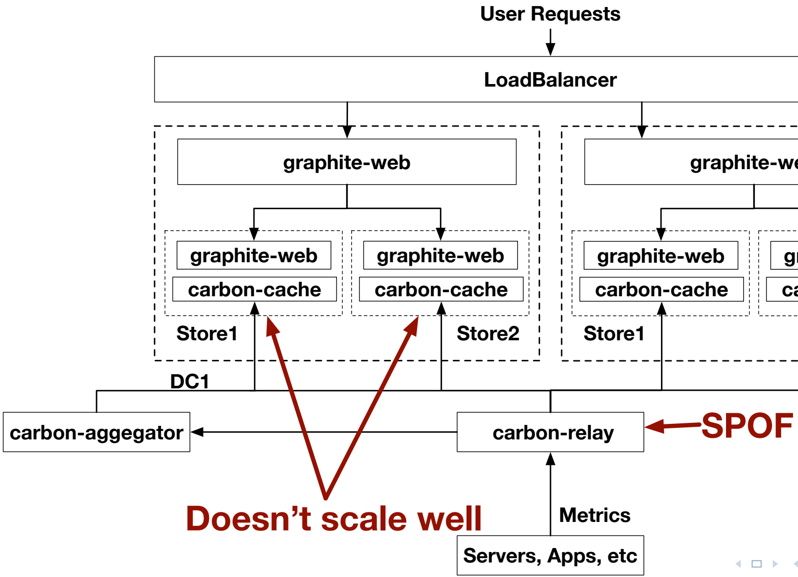
Original stack



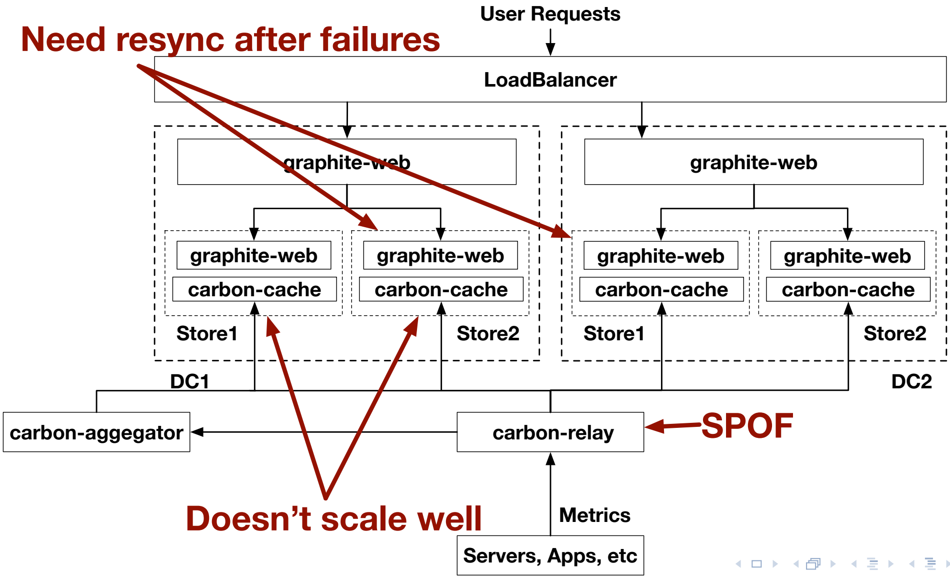
Problems: Relay



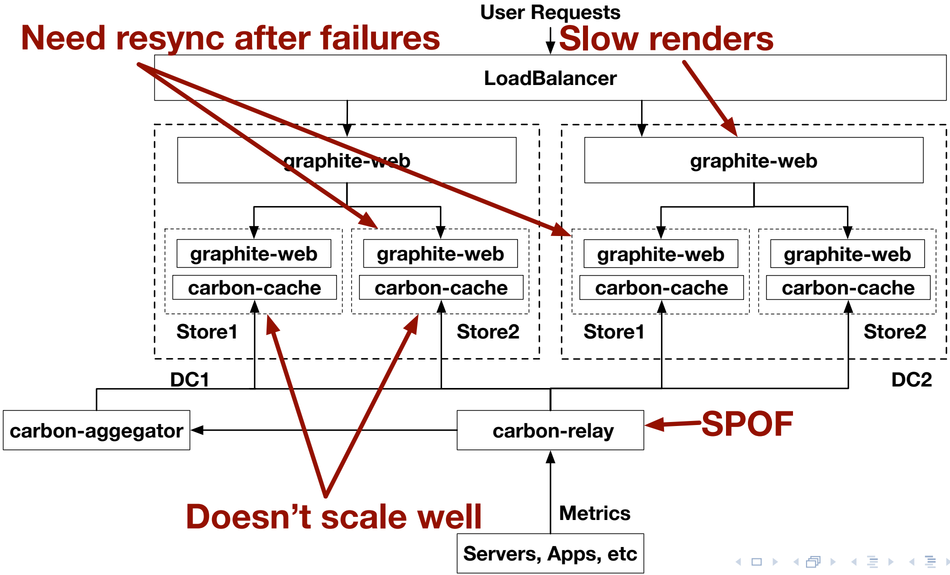
Problems: Scalability



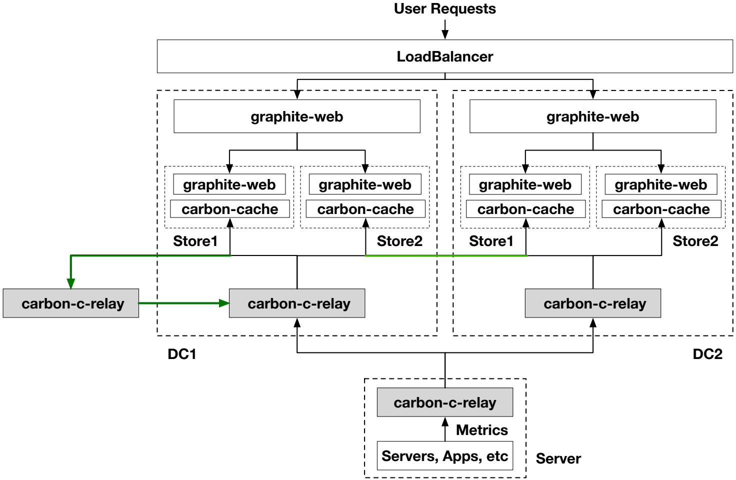
Problems: Consistency



Problems: Render time



Replacing carbon-relay



Replacing carbon-relay

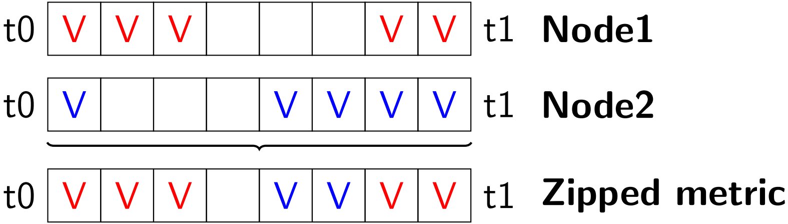
carbon-c-relay:

- 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

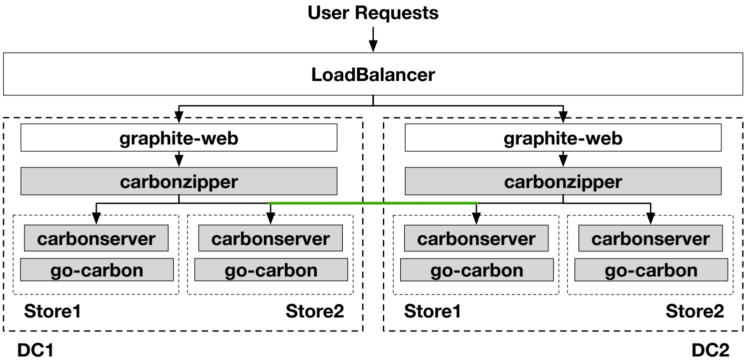
Zipper stack: Solution

Query: target=sys.server.cpu.user

Result:


t0 V V V _ _ _ V V t1 Node1
t0 V _ _ _ V V V V t1 Node2
t0 V V V _ V V V V t1 Zipped metric

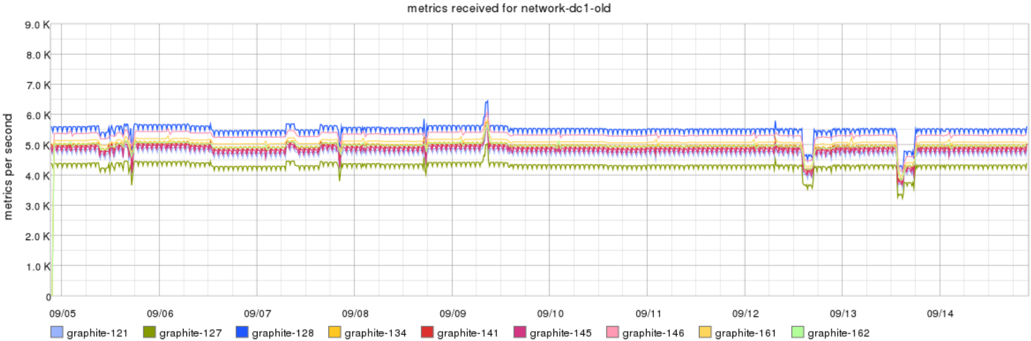
Zipper stack: architecture



Zipper stack: results

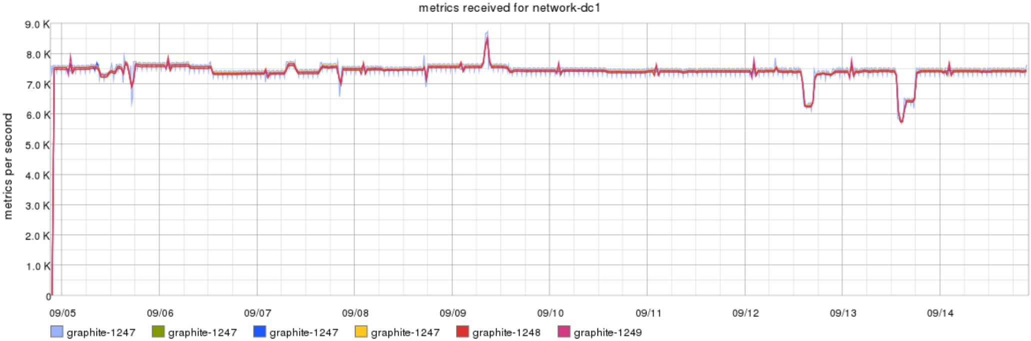
- Written in **Go**
- Can query store servers in **parallel**
- Can "Zip" the data
- carbonzipper ⇔ carbonserver — **2700** RPS
  graphite-web ⇔ carbon-cache — **80** RPS.
- carbonserver is now part of go-carbon (since December 2016)

Metric distribution: how it works



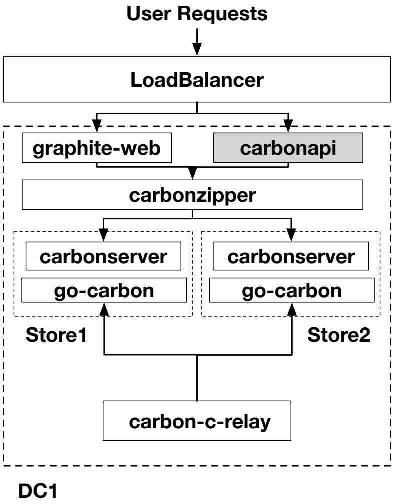
Up to **20%** difference in worst case

Metric distribution: jump hash

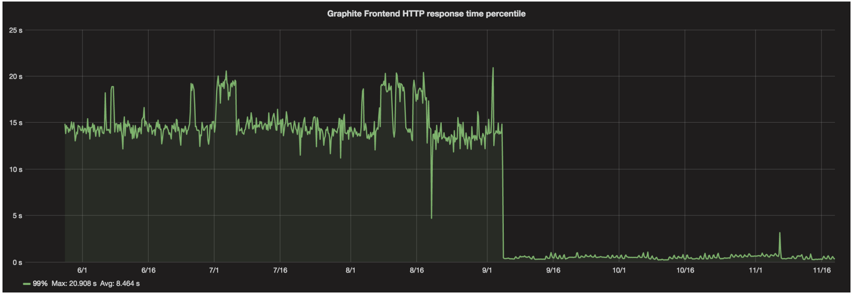


arxiv.org/pdf/1406.2294v1.pdf

Rewriting Frontend in Go: carbonapi

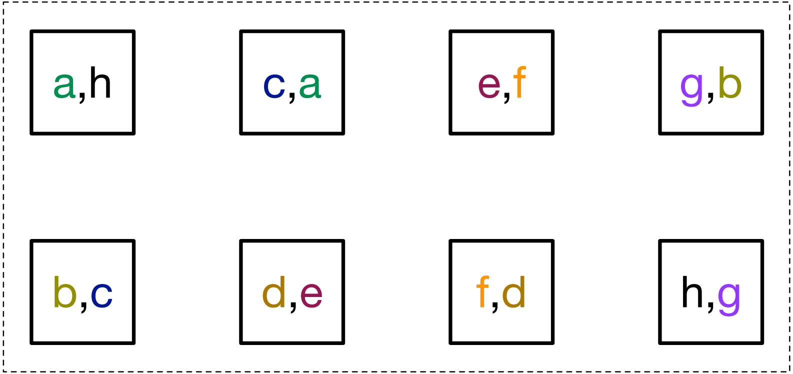


Rewriting Frontend in Go: result



- Significantly reduced response time for users (**15s ⇒ 0.8s**)
- Allows more complex queries because it's faster
- Easier to implement new heavy math functions
- Parsing and functions are available as separate libraries.

Replication techniques and their pros and cons



a,h

c,a

e,f

g,b

b,c

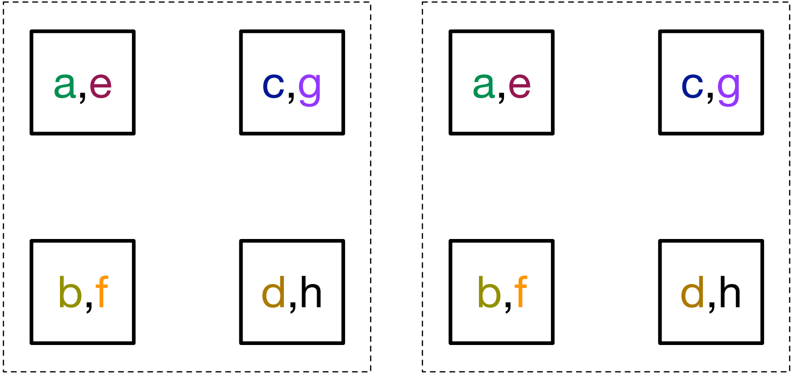
d,e

f,d

h,g

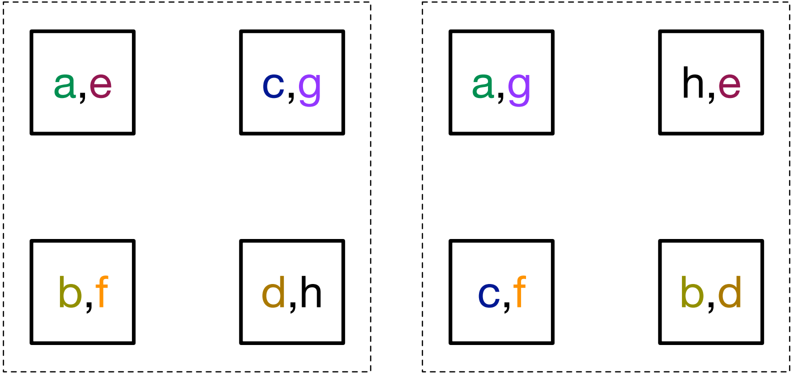
Replication Factor 2

Replication techniques and their pros and cons



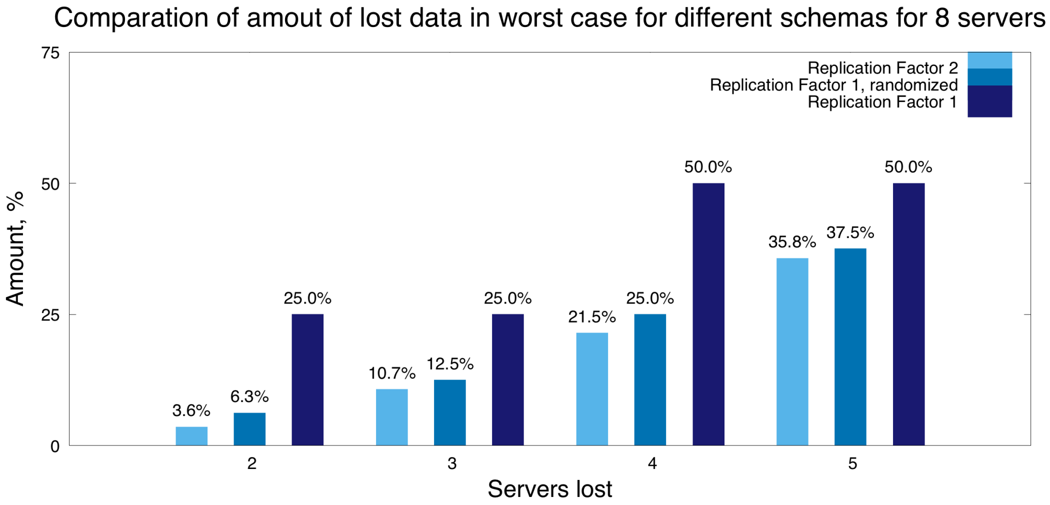
Replication Factor 1

Replication techniques and their pros and cons

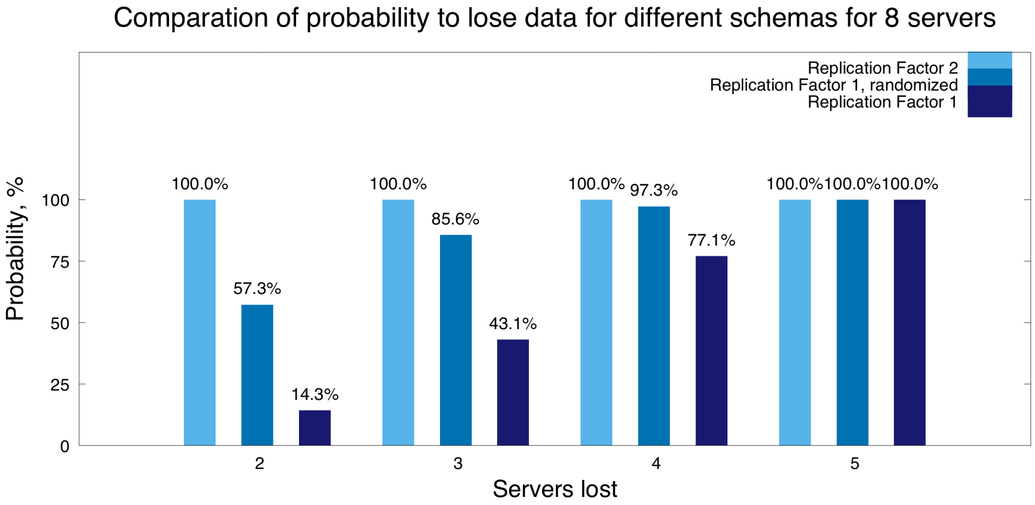


Replication Factor 1, randomized

Replication techniques and their pros and cons



Replication techniques and their pros and cons



Adding simple tags

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

- Separated tags stream and storage
- No history
- No negative match support (yet)
- Only ”and” syntax

What’s next?

- Find a replacement for Whisper (in progress)
- Replace graphite line protocol between components (in progress)
- Migrate to streaming protocol between backends (in progress).
- Implement differential flamegraphs
- Continue to work on collecting traces

It's all Open Source!

- carbon-c-relay — github.com/grobian/carbon-c-relay
- carbonzipper — github.com/go-graphite/carbonzipper
- go-carbon — github.com/lomik/go-carbon
- carbonapi — github.com/go-graphite/carbonapi
- carbonsearch — github.com/kanatohodets/carbonsearch
- gorelka — github.com/go-graphite/gorelka
- flamegraphs — github.com/Civil/ch-flamegraphs
- replication factor test — github.com/Civil/graphite-rf-test

Several major users: Booking.com, eBay Classifieds Group and Slack

Questions?

vladimir.smirnov@booking.com

civil.over@gmail.com

Twitter: @Civilus

Facebook: civilus

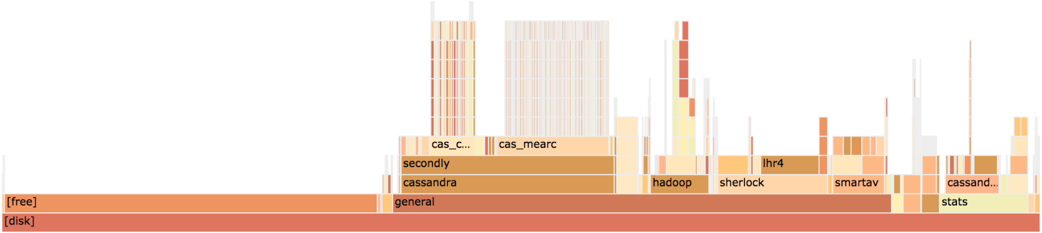
Telegram: Civiloid

LinkedIn: vladsmirnov

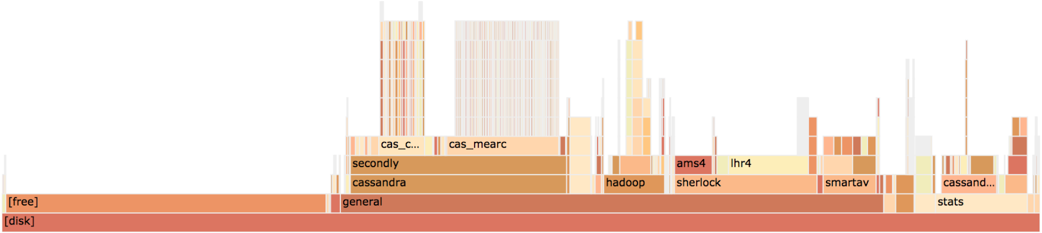
Thanks!

We are hiring SREs in Amsterdam!
https://workingatbooking.com

Bonus: Instrumenting: FlameGraphs: Before



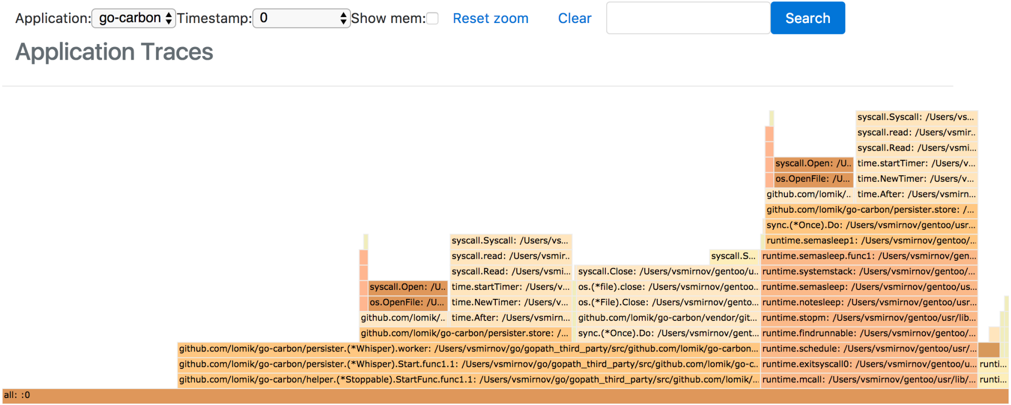
Bonus: Instrumenting: FlameGraphs: After



Bonus: Instrumenting: Conclusion

- Collect and Store information about every metric
- Database: Clickhouse
- Stores raw data about each metric: name, size, mtime, access time, etc.

Bonus: Instrumenting: Profiling stack



Application: go-carbon
Timestamp: 0
Show mem: □
Reset zoom
Clear
Search
Application Traces
syscall.Syscall: /Users/vs...
syscall.read: /Users/vsmir..
syscall.Read: /Users/vsmi...
syscall.Open: /U..
time.startTimer: /Users/v...
os.OpenFile: /U...
time.NewTimer: /Users/v...
github.com/lomik/..
time.After: /Users/vsmirn...
github.com/lomik/go-carbon/persister.store: /...
sync.(*Once).Do: /Users/vsmirnov/gentoo/usr...
syscall.Syscall: /Users/vs...
runtime.semasleep1: /Users/vsmirnov/gentoo/...
syscall.read: /Users/vsmir..
syscall.S...
runtime.semasleep.func1: /Users/vsmirnov/gen...
syscall.Read: /Users/vsmi...
syscall.Close: /Users/vsmirnov/gentoo/u..
runtime.systemstack: /Users/vsmirnov/gentoo/...
syscall.Open: /U..
time.startTimer: /Users/v...
os.(*file).close: /Users/vsmirnov/gentoo..
runtime.semasleep: /Users/vsmirnov/gentoo/us...
os.OpenFile: /U...
time.NewTimer: /Users/v...
os.(*File).Close: /Users/vsmirnov/gento...
runtime.notesleep: /Users/vsmirnov/gentoo/usr...
github.com/lomik/..
time.After: /Users/vsmirn...
github.com/lomik/go-carbon/vendor/git...
runtime.stopm: /Users/vsmirnov/gentoo/usr/lib...
github.com/lomik/go-carbon/persister.store: /...
sync.(*Once).Do: /Users/vsmirnov/gent...
runtime.findrunnable: /Users/vsmirnov/gentoo/...
github.com/lomik/go-carbon/persister.(*Whisper).worker: /Users/vsmirnov/go/gopath_third_party/src/github.com/lomik/go-carbon...
runtime.schedule: /Users/vsmirnov/gentoo/usr/...
github.com/lomik/go-carbon/persister.(*Whisper).Start.func1.1: /Users/vsmirnov/go/gopath_third_party/src/github.com/lomik/go-c..
runtime.exitsyscall0: /Users/vsmirnov/gentoo/u...
runti...
github.com/lomik/go-carbon/helper.(*Stoppable).StartFunc.func1.1: /Users/vsmirnov/go/gopath_third_party/src/github.com/lomik/...
runtime.mcall: /Users/vsmirnov/gentoo/usr/lib/...
runti...
all: :0