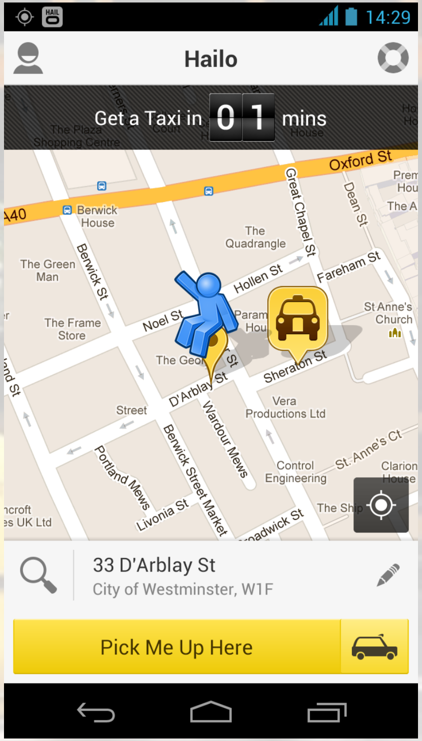
Running Hailo on Cassandra

The kung-fu of “medium-sized data”

What is Hailo?

- Taxi-app connecting passengers to drivers
- Operating in locations around the world
- Available round-the-clock
- We want our app to be usable anywhere
- We may need to scale up operations at any time



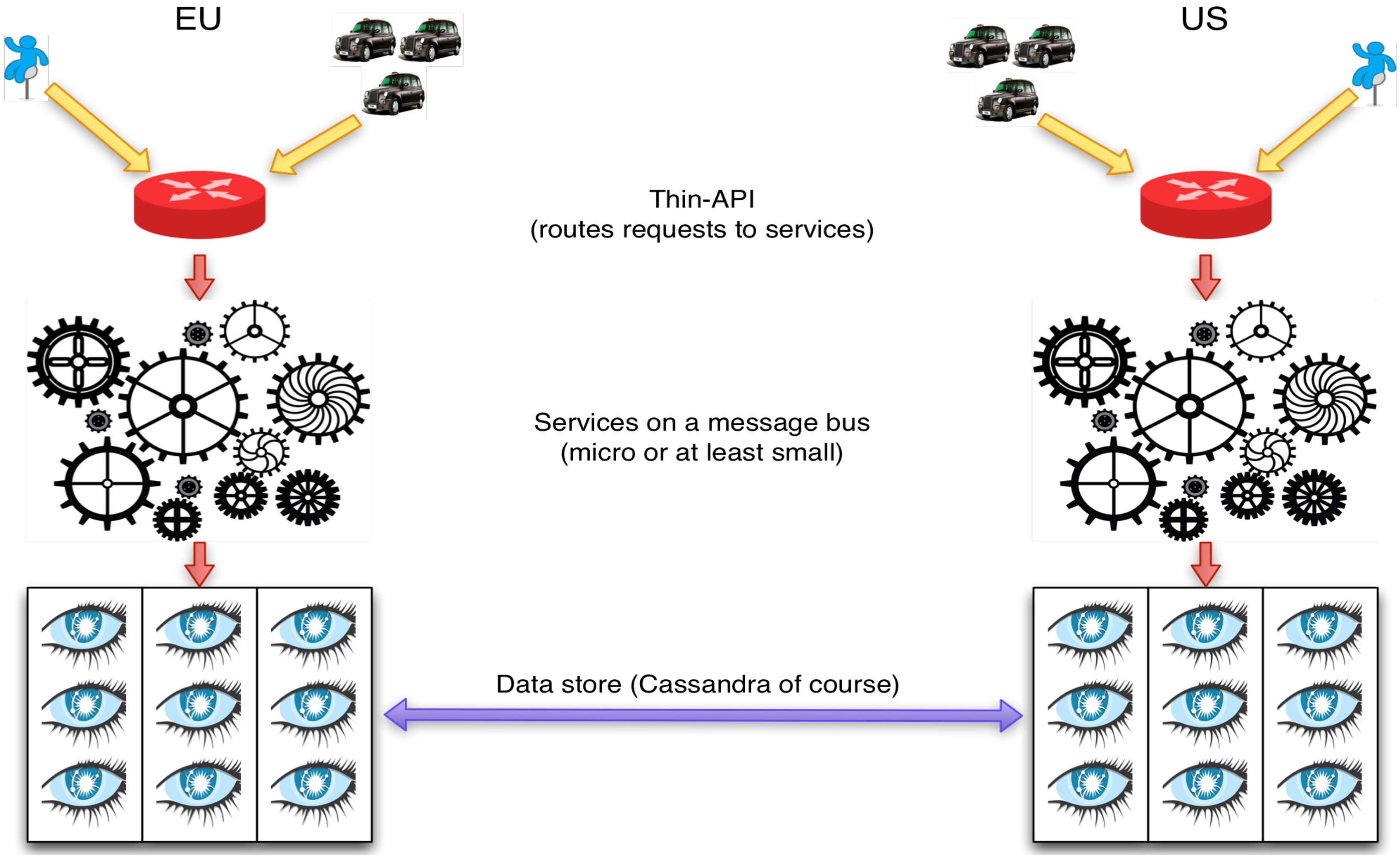
Why Cassandra?

- Elegant architecture (truly masterless topology)
- Linearly scalable (want more power then just add more nodes)
- Flexible (add new DCs on-the-fly)
- Fault-tolerant
- Not necessarily because we have big-data - just big requirements
- Simple!

How C* helped us grow

- C* made it easy for us to replicate our data across multiple DCs
- Capacity could be added as we needed it without interrupting live traffic
- Zero downtime upgrades meant that we could perform operations during business hours
- Fault-tolerance allowed us to sleep easily at night knowing that we would be free from outages
- Defeated the end-of-level bosses of scaling and stability

HAIL



EU

US

Thin-API
(routes requests to services)

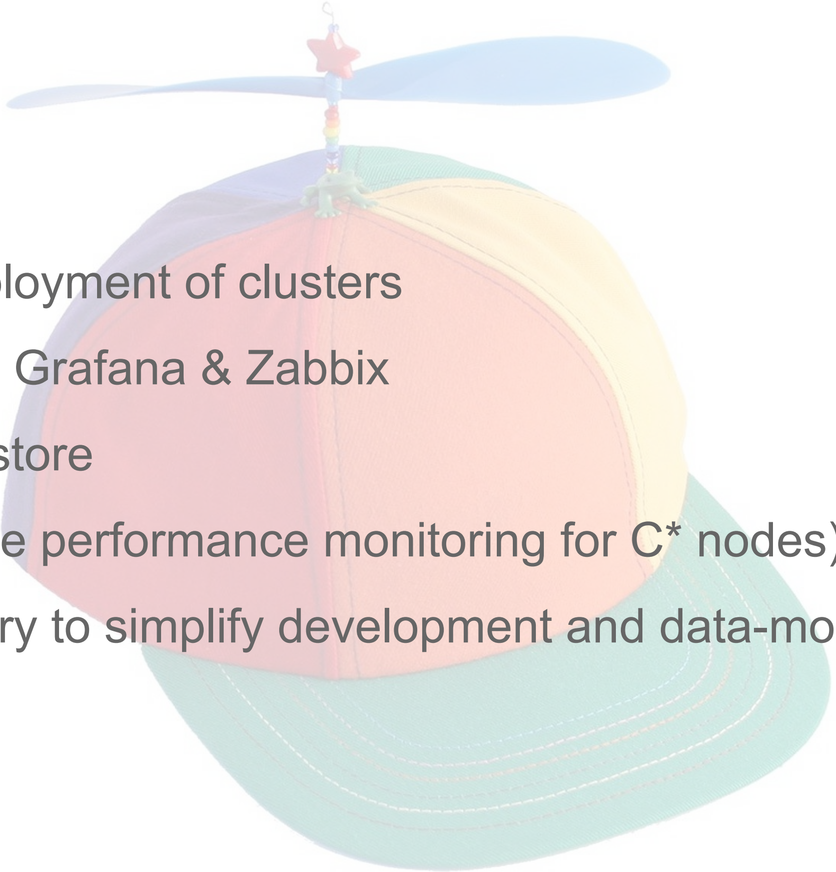
Services on a message bus
(micro or at least small)

Data store (Cassandra of course)

The Stack

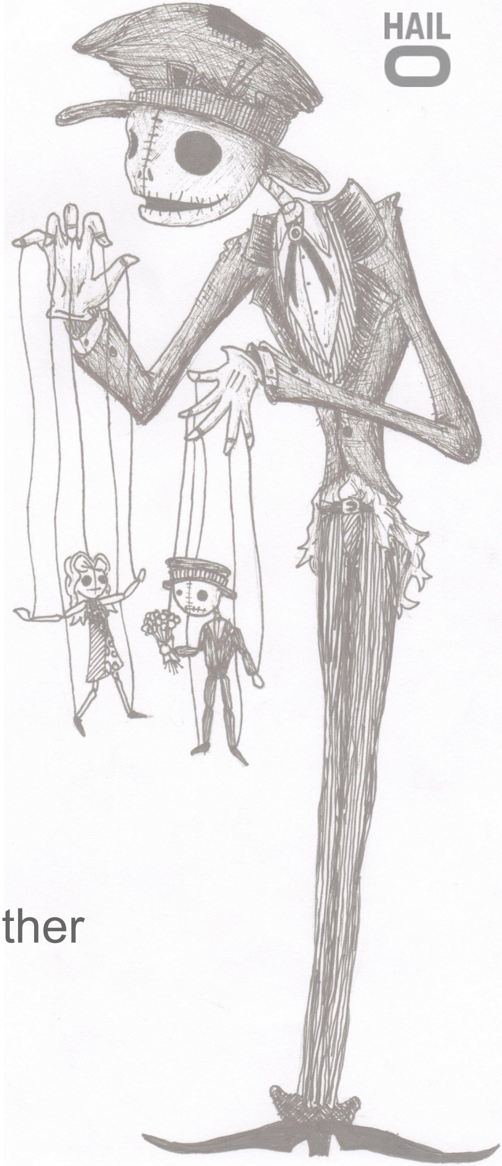
- Pure AWS
- DCs connected by OpenVPN
- C* nodes are either c3.2xlarge (premium) & m1.xlarge (economy)
- All storage is on striped-ephemeral disk (fast and cost-effective)
- In each DC we use 3 availability-zones
- Each cluster is scaled in multiples of 3
- Data is stored using RF=3 (one copy in each AZ with NTS)
- Most queries are local-quorum, often reads are relaxed to ONE
- Microservices in GO (performant self-contained binaries)
- NSQ & Rabbit as message buses
- Ubuntu Server

Things we had to develop

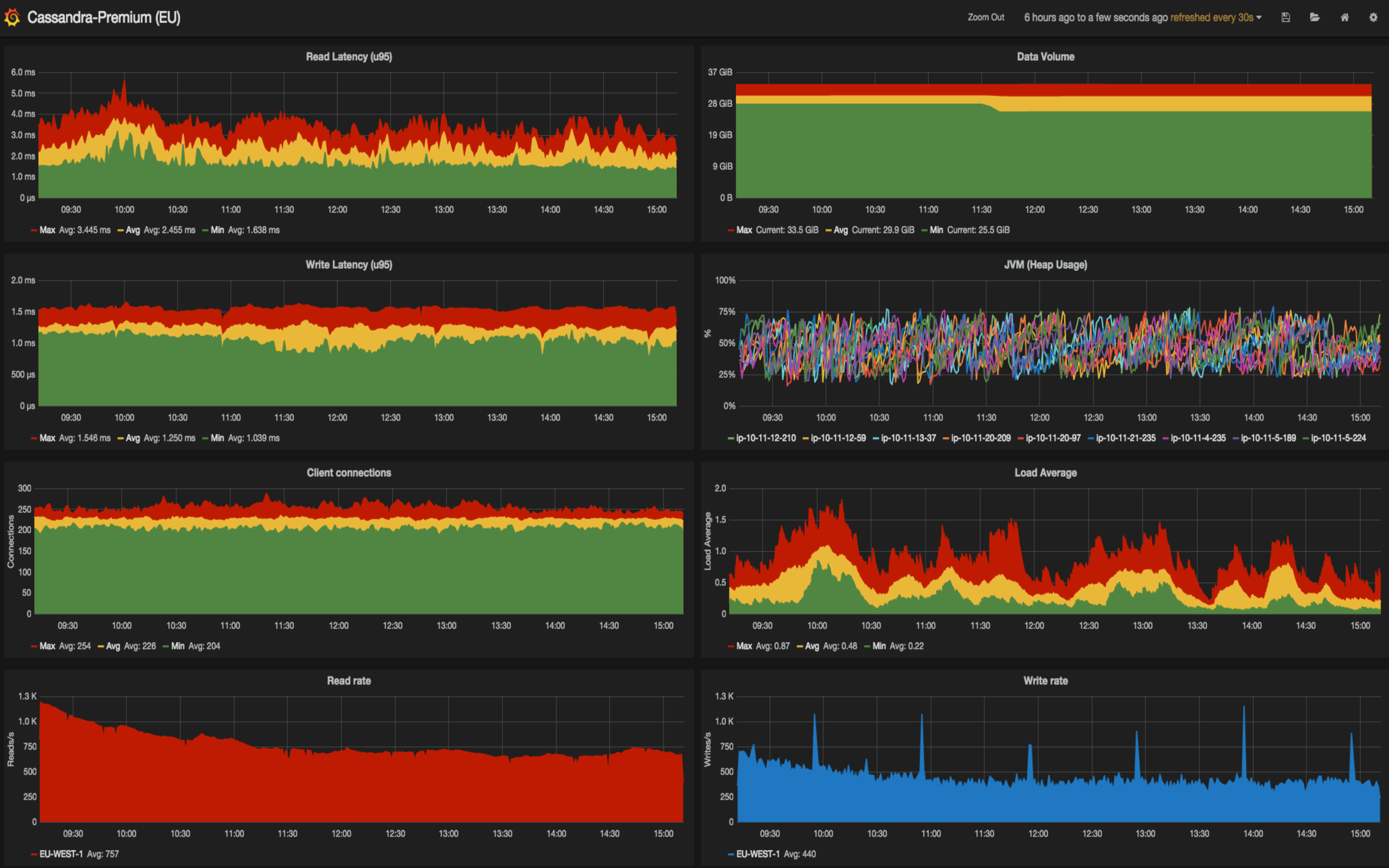


- Automated deployment of clusters
- Monitoring with Grafana & Zabbix
- Backup and restore
- CTOP (real-time performance monitoring for C* nodes)
- GoCassa (library to simplify development and data-modeling)

Automation



- Nodes born in static autoscaling groups from JSON templates
- Storage automatically striped / encrypted / mounted
- Joined to puppet using cloud-init (thanks Ubuntu)
- Clustered using custom AWS/Puppet plugins to locate seeds
- Options to automatically create schemas and load test-data
- Scripts to “migrate” data incrementally from one cluster to another



Cassandra-Premium (EU)

Read Latency (u95)

Max Avg: 3.445 ms Avg Avg: 2.455 ms Min Avg: 1.638 ms

Data Volume

Max Current: 33.5 GiB Avg Current: 29.9 GiB Min Current: 25.5 GiB

Write Latency (u95)

Max Avg: 1.546 ms Avg Avg: 1.250 ms Min Avg: 1.039 ms

JVM (Heap Usage)

ip-10-11-12-210 ip-10-11-12-59 ip-10-11-13-37 ip-10-11-20-209 ip-10-11-20-97 ip-10-11-21-235 ip-10-11-4-235 ip-10-11-5-189 ip-10-11-5-224

Client connections

Max Avg: 254 Avg Avg: 226 Min Avg: 204

Load Average

Max Avg: 0.87 Avg Avg: 0.48 Min Avg: 0.22

Read rate

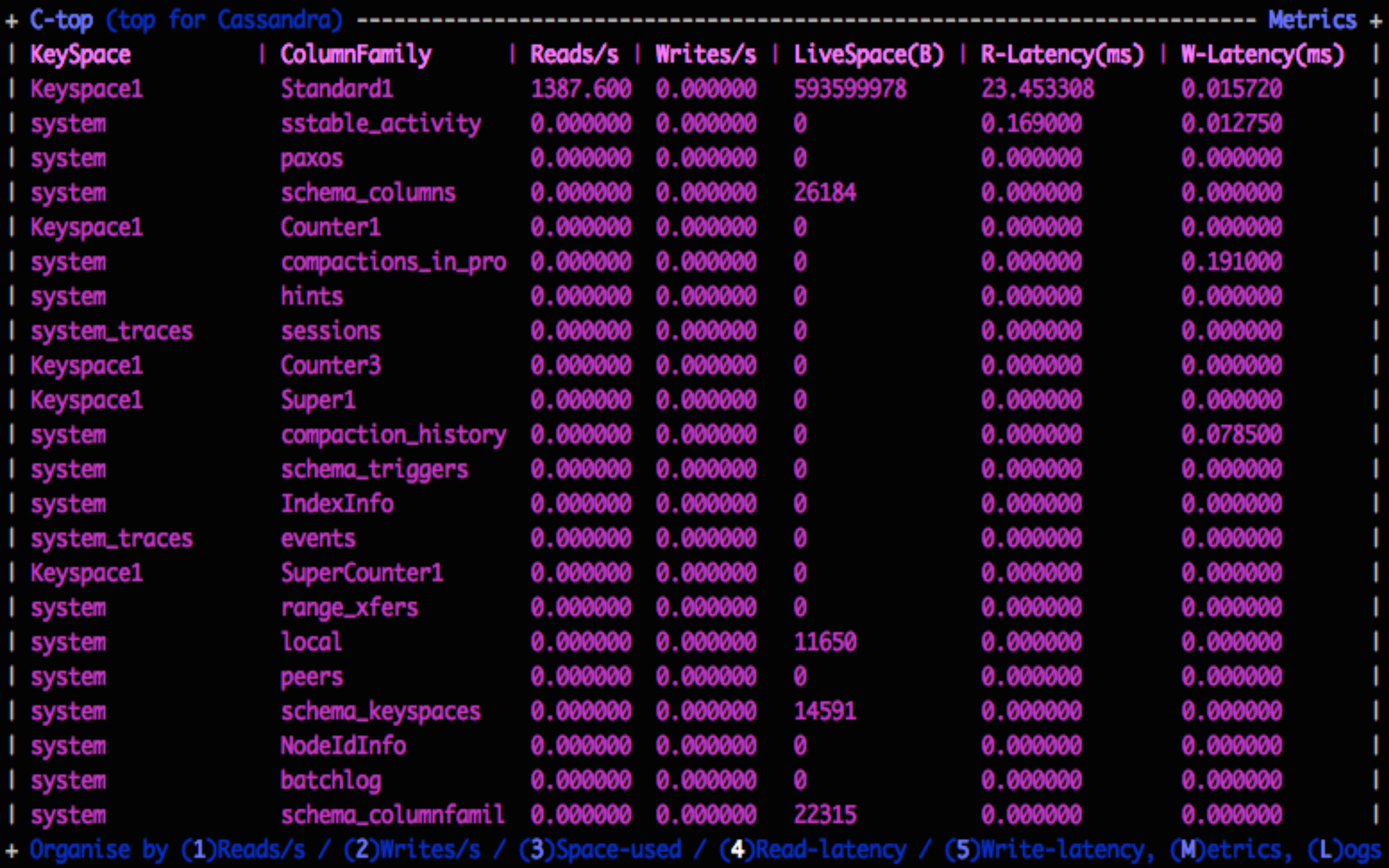
EU-WEST-1 Avg: 757

Write rate

EU-WEST-1 Avg: 440

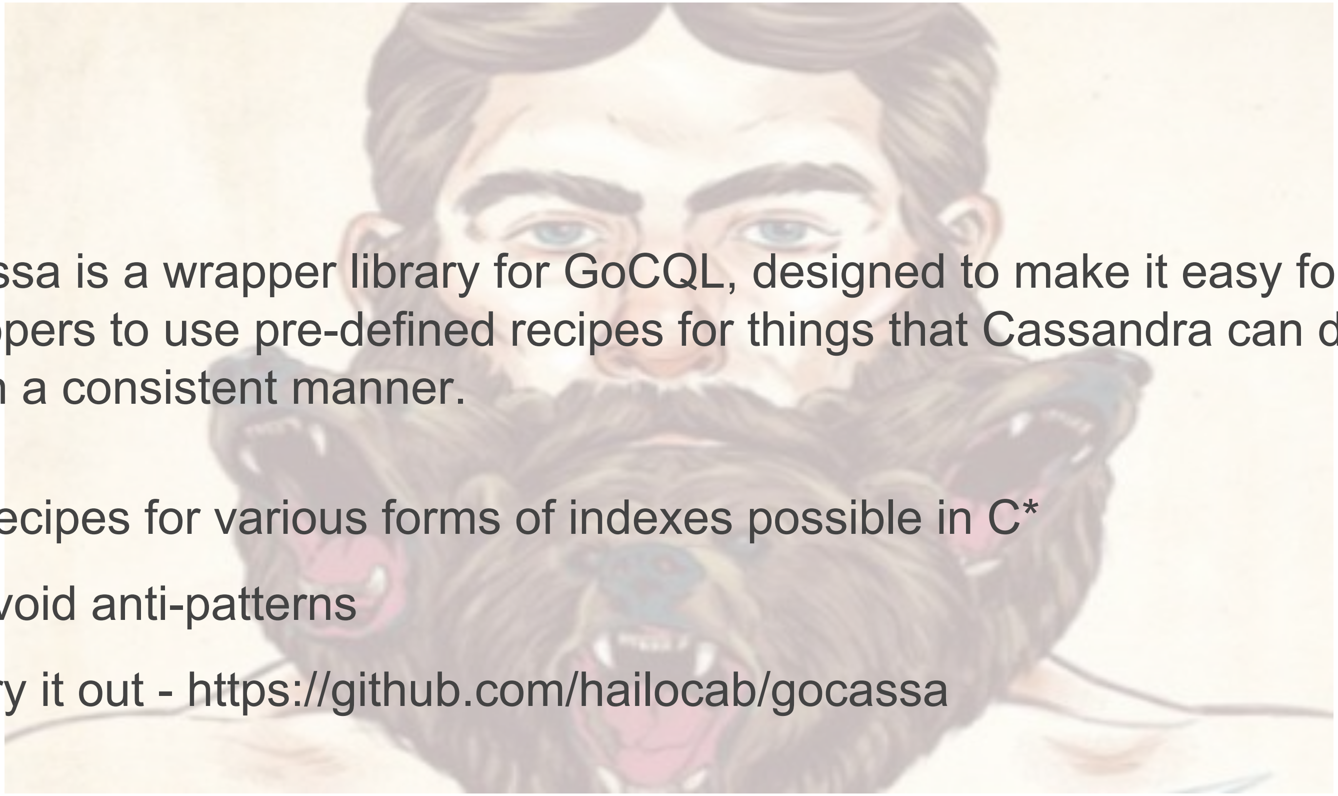
Backup & restore with S3

- Built on the legendary “s3cmd”
- Only transfers new SSTables (and --delete those that no longer exist)
- Disable the md5-check (sstables are immutable… name & time is fine)
- Encryption handled by AWS API using “SSE-C”
- Remember which files you had on each day for “point-in-time” restores
- One day you will say “thanks” to your past-self for doing this



```
+ C-top (top for Cassandra) ------------------------------------------------------------------------ Metrics +
| KeySpace          | ColumnFamily      | Reads/s | Writes/s | LiveSpace(B) | R-Latency(ms) | W-Latency(ms)  |
| Keyspace1           Standard1           1387.600  0.000000   593599978      23.453308       0.015720       |
| system              sstable_activity    0.000000  0.000000   0              0.169000        0.012750       |
| system              paxos               0.000000  0.000000   0              0.000000        0.000000       |
| system              schema_columns      0.000000  0.000000   26184          0.000000        0.000000       |
| Keyspace1           Counter1            0.000000  0.000000   0              0.000000        0.000000       |
| system              compactions_in_pro  0.000000  0.000000   0              0.000000        0.191000       |
| system              hints               0.000000  0.000000   0              0.000000        0.000000       |
| system_traces       sessions            0.000000  0.000000   0              0.000000        0.000000       |
| Keyspace1           Counter3            0.000000  0.000000   0              0.000000        0.000000       |
| Keyspace1           Super1              0.000000  0.000000   0              0.000000        0.000000       |
| system              compaction_history  0.000000  0.000000   0              0.000000        0.078500       |
| system              schema_triggers     0.000000  0.000000   0              0.000000        0.000000       |
| system              IndexInfo           0.000000  0.000000   0              0.000000        0.000000       |
| system_traces       events              0.000000  0.000000   0              0.000000        0.000000       |
| Keyspace1           SuperCounter1       0.000000  0.000000   0              0.000000        0.000000       |
| system              range_xfers         0.000000  0.000000   0              0.000000        0.000000       |
| system              local               0.000000  0.000000   11650          0.000000        0.000000       |
| system              peers               0.000000  0.000000   0              0.000000        0.000000       |
| system              schema_keyspaces    0.000000  0.000000   14591          0.000000        0.000000       |
| system              NodeIdInfo          0.000000  0.000000   0              0.000000        0.000000       |
| system              batchlog            0.000000  0.000000   0              0.000000        0.000000       |
| system              schema_columnfamil  0.000000  0.000000   22315          0.000000        0.000000       |
+ Organise by (1)Reads/s / (2)Writes/s / (3)Space-used / (4)Read-latency / (5)Write-latency, (M)etrics, (L)ogs
```

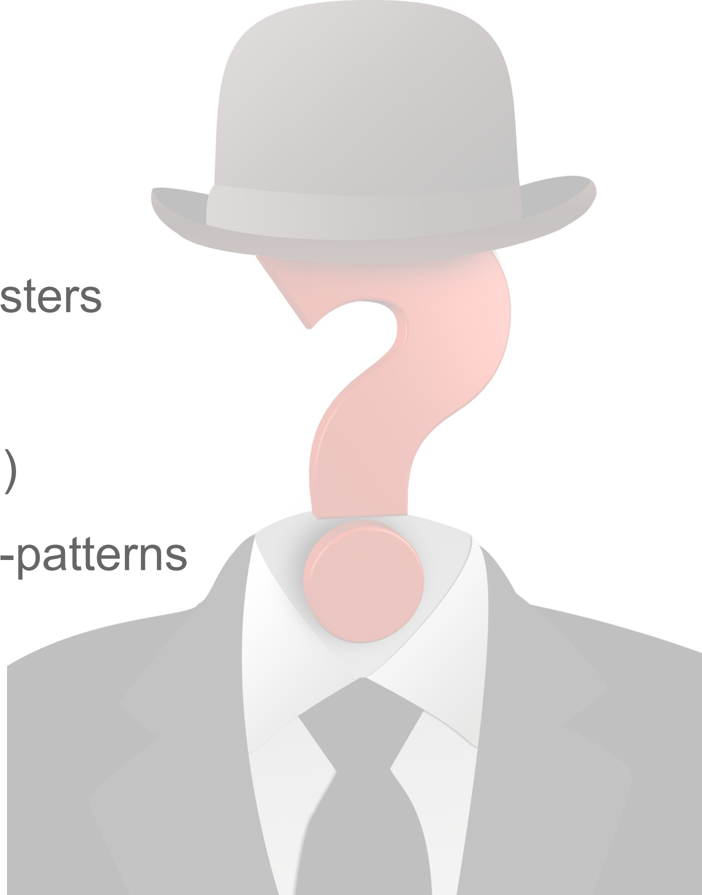
GoCassa



GoCassa is a wrapper library for GoCQL, designed to make it easy for developers to use pre-defined recipes for things that Cassandra can do well, in a consistent manner.

- Recipes for various forms of indexes possible in C*
- Avoid anti-patterns
- Try it out - https://github.com/hailocab/gocassa

Advice for first-time users



- Automate all the things from day one
- Understand how your queries affect your C* clusters
- Practise restoring data REGULARLY
- Don’t overload your nodes with data (or queries)
- Invest time in designing data-models and query-patterns
- Advocate Cassandra and lead by example

- Continue to develop our tools to allow swarms of micro-services to share Cassandra clusters
- Continue the development of the GoCassa library to make it easy to get the best out of Cassandra
- Keep investigating new ways to model data for Cassandra
- Keep enjoying the performance and stability!

Thanks for listening!

- → http://jobs.hailocab.com/
- → https://github.com/hailocab/ctop
- → https://github.com/hailocab/gocassa

chris.hoolihan@hailocab.com