CASSANDRA SUMMIT 2016

Alex Petrov

Scalable time-series applications with Cassandra

Scalable Time Series with Cassandra

@ifesdjeen

Cyanite

past, present and future

Requirements

Throughput

Incoming data
Aggregates
Read queries

Scalability

Paths / metrics

Historical data

Readers / writers

Aggregated vs raw

Predefined list of reports to aggregate
Reducing amount of data points
Faster queries
Precision loss due to aggregation

```
“servers.*.workers.busyWorkers”: “10s:7d,1m:21d,15m:5y”
```

Graphite

Store numeric time-series data
Render graphs of this data on demand

https://graphiteapp.org/

Ecosystem

collectd

RIEMANN

Nagios Core™

logstash

Grafana

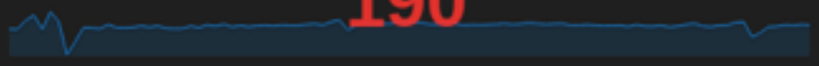
carbon: listens for time-series data
whisper: DB for storing TS data
graphite-web: UI & API for graphs

Website Overview

Zoom Out | Last 3 hours

Logins

190



Sign ups

269

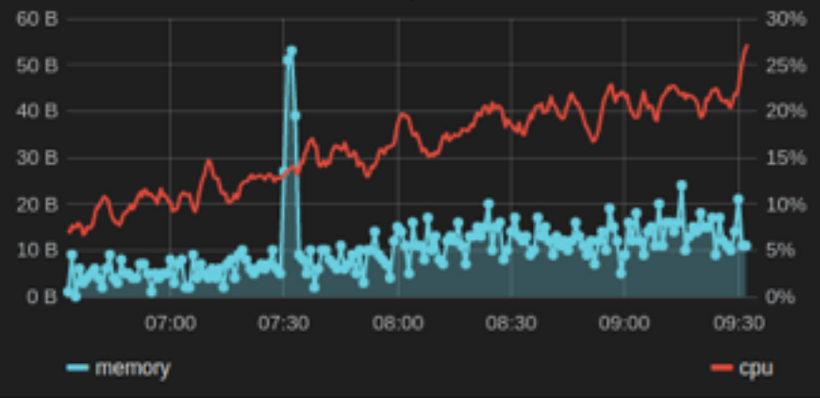
Sign outs

273

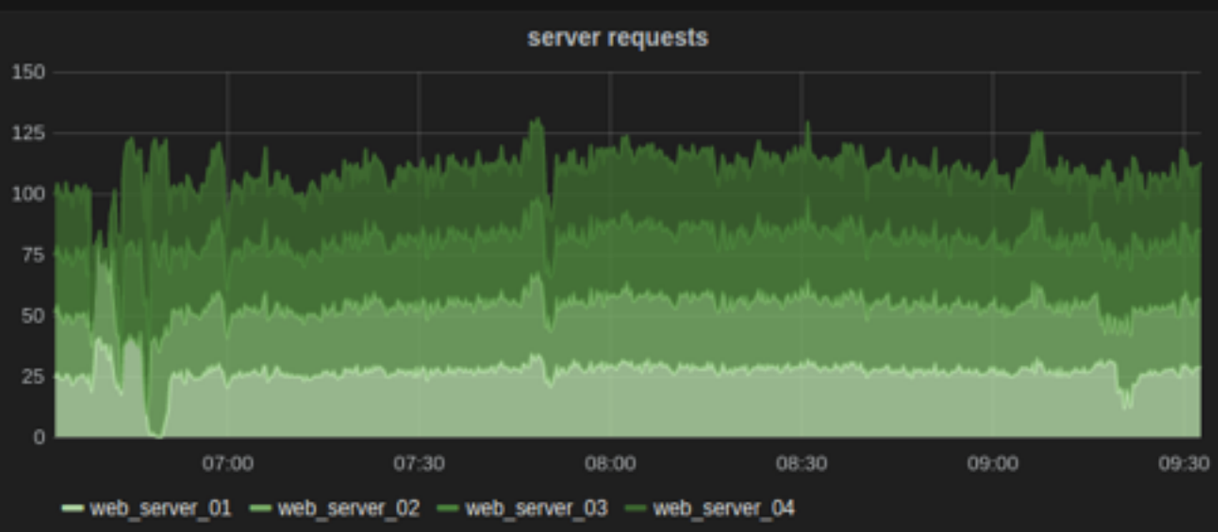
Memory / CPU

logins

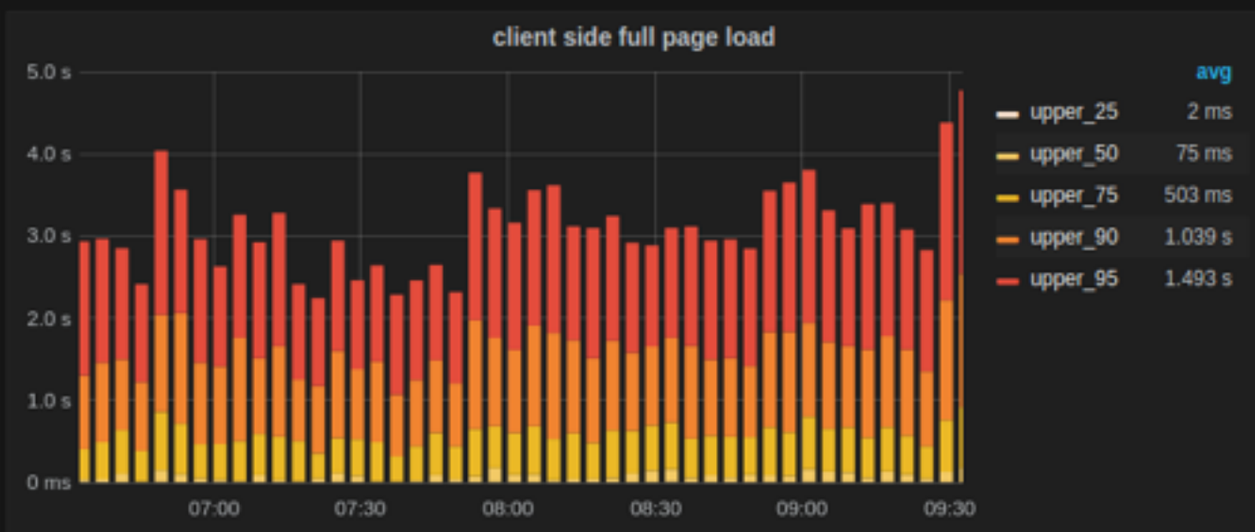
Memory / CPU

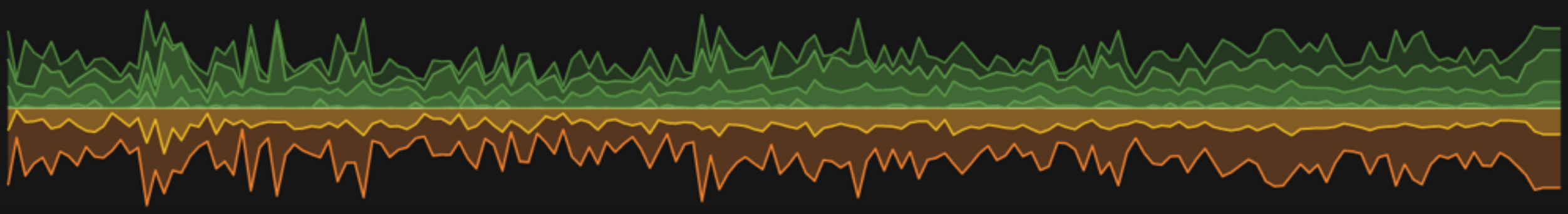


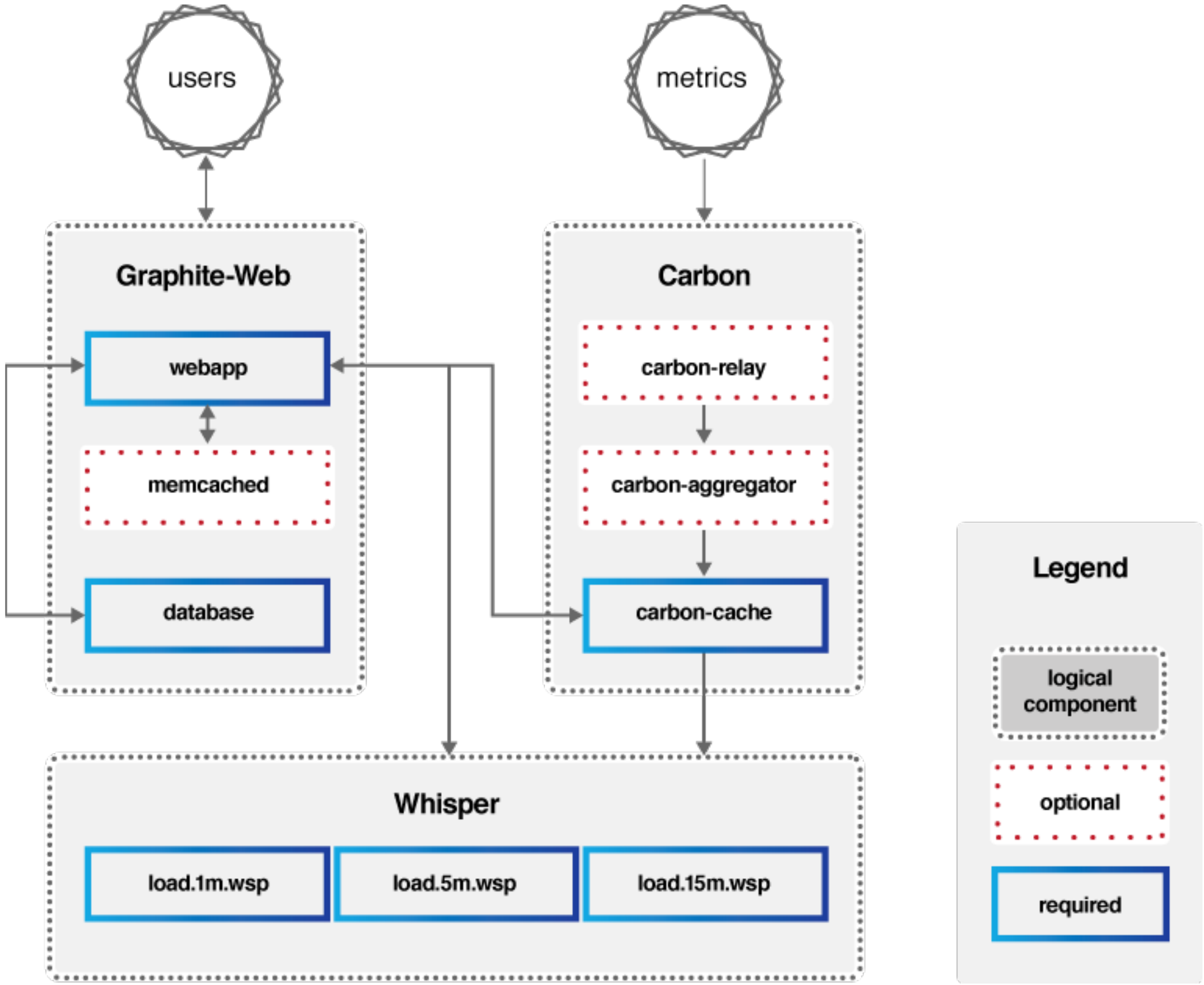
server requests



client side full page load






users
metrics
Graphite-Web
webapp
memcached
database
Carbon
carbon-relay
carbon-aggregator
carbon-cache
Whisper
load.1m.wsp
load.5m.wsp
load.15m.wsp
Legend
logical component
optional
required

Downsides

Single-host solution
Plenty of disk seeks
Optimised for space
Sharding / replication is “manual”

Scaling

Cluster topology stored on every node

Manual replication

Changing cluster topology non-trivial

Scaling

Stateless service

Automatic shard assignment

Replication

Easy management

Easy topology changes

reminds you of anything?

Cassandra

perfect match

Cyanite

Stateless
Async I/O
Custom scheduler
No Whisper files
Distributed
Horizontally scalable

Cyanite responsibilities

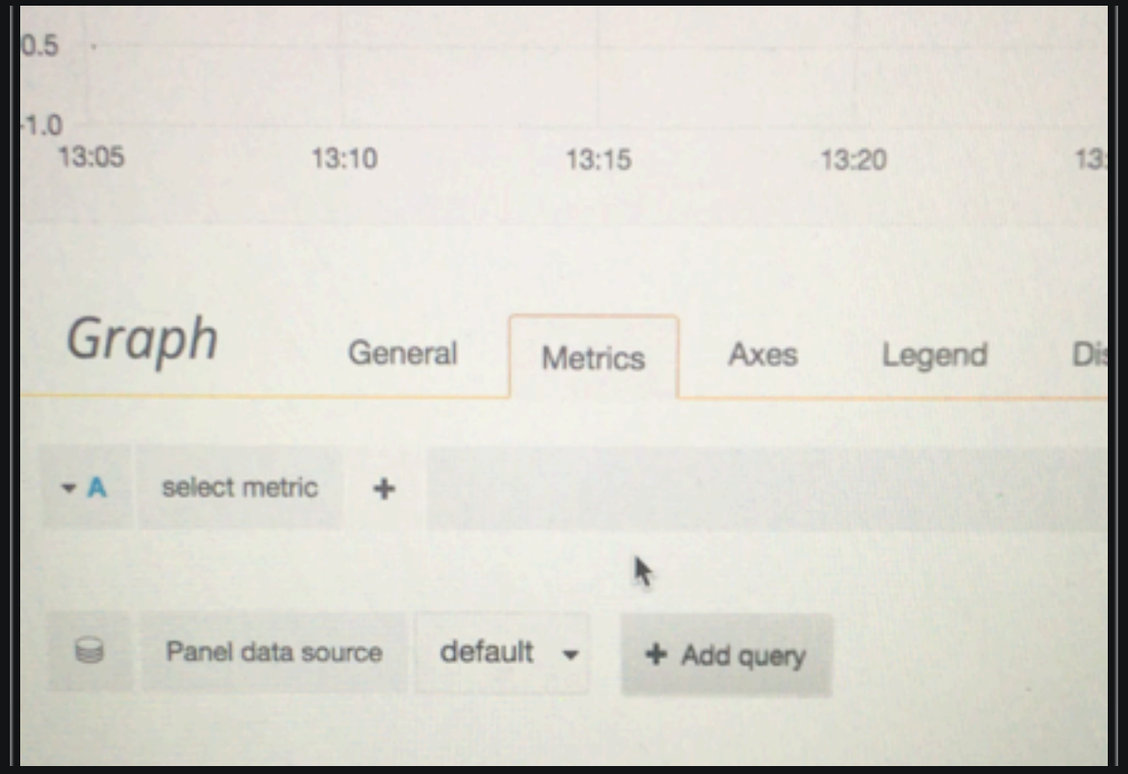
Carbon-compatible listener
Aggregate data in-memory
Flush aggregates to Cassandra
Path storage
Retrieve paths for query
Aggregate the query results

Cassandra features

Metric expiry, TTL
User Defined Types
SASI Indexes, `LIKE` queries
Aggregate Functions
Offline data loading
Clustering range queries
IN partition key locking

Globbing

index



app.cluster.server.subsystem.metric

Built with SASI indexes
Fast, scalable queries
Optimised for glob

```
CREATE TABLE segment (
    parent text,
    segment text,
    pos int,
    length int,
    leaf boolean,
    PRIMARY KEY (parent, segment))
```

app.cluster.server.subsystem.metric

Wildcard: *

```
SELECT * FROM segments WHERE
parent = 'root' AND
pos = 1
```

Postfix: *.*.*.*.metric

SELECT * FROM segments WHERE pos = 4

Prefix: app.*

```
SELECT * FROM segments WHERE
parent = 'app' AND
pos = 2 ALLOW FILTERING
```

Suffix: abc.*.metric

```
SELECT * FROM SEGMENTS
WHERE
pos = 3 AND
segment LIKE 'abc.%'
ALLOW FILTERING
```

Query engine

Query engine

2-Step transformations:

- Inner
- Outer
- Cross metric

scale(a.b.*, 10.0)

↓

a.b.c1 a.b.c2

↓

```
{"a.b.c1" [1 2 3]
 "a.b.c2" [5 6 7]}
```

↓

```
{"a.b.c1" [10 20 30]
 "a.b.c2" [50 60 70]}
```

derivative(a.b.c)

↓

a.b.c

↓

```
{"a.b.c" [1 3 6]}
```

↓

```
{“derivative(a.b.c)" [nil 2 3]}
```

sumSeries(a.b.c1,a.b.c2)

↓

a.b.c1 a.b.c2

↓

```
{"a.b.c1" [1 1 1]
 "a.b.c2" [2 2 2]}
```

↓

```
{“sumSeries(a.b.c,a.b.d)" [3 3 3]}
```

Data model

```
CREATE TYPE IF NOT EXISTS metric_resolution (
  precision  int,
  period     int
);
```

```
CREATE TYPE IF NOT EXISTS metric_id (
  path       text,
  resolution frozen<metric_resolution>
);
```

```
CREATE TYPE IF NOT EXISTS metric_point (
   max double,
   mean double,
   min double,
   sum double
);
```

Cassandra

Aggregates

Cassandra aggregates

Locked partition
Flexible query definition
Local aggregation
Reduced raw data chatter

```
CREATE OR REPLACE FUNCTION sumState
    (state map<bigint, double>, ts bigint, val metric_point)
CALLED ON NULL INPUT RETURNS map<bigint, double>
LANGUAGE java AS $$

  if (state.containsKey(ts)) {
    state.put(ts, state.get(ts) + val.getDouble("mean"));
  } else {
    state.put(ts, val.getDouble("mean"));
  }

return state;
$$;
```

```
CREATE OR REPLACE FUNCTION sumFinal
                    (state map<bigint, double>)
CALLED ON NULL INPUT RETURNS map<bigint, double>
LANGUAGE java AS
'return state;';
```

```
SELECT sum(time, point) FROM metric WHERE id
IN ({path: 'a', resolution: {precision: 1, period: 3600}},
    {path: 'b', resolution: {precision: 1, period: 3600}})
AND time > 0
AND time < 5;
```

Scaling

Scaling cyanite

Use DTSC
Cyanite is stateless
For high loads, split readers and writers
Load-balance with HAProxy
Colocate Cyanite & Cassandra

Coming soon

Coming soon

P-Square histograms
T-Digest quantiles
Gorilla Compression
Cassandra aggregates
Custom glob indexes
Scheduler

Coming soon

Kafka ingester
Statsd protocol support
Standalone cyanite
Dynamic thresholds


SUPERB
wallpapers
That's all Folks!