CASSANDRA SUMMIT2015

Skinny on Wide Rows

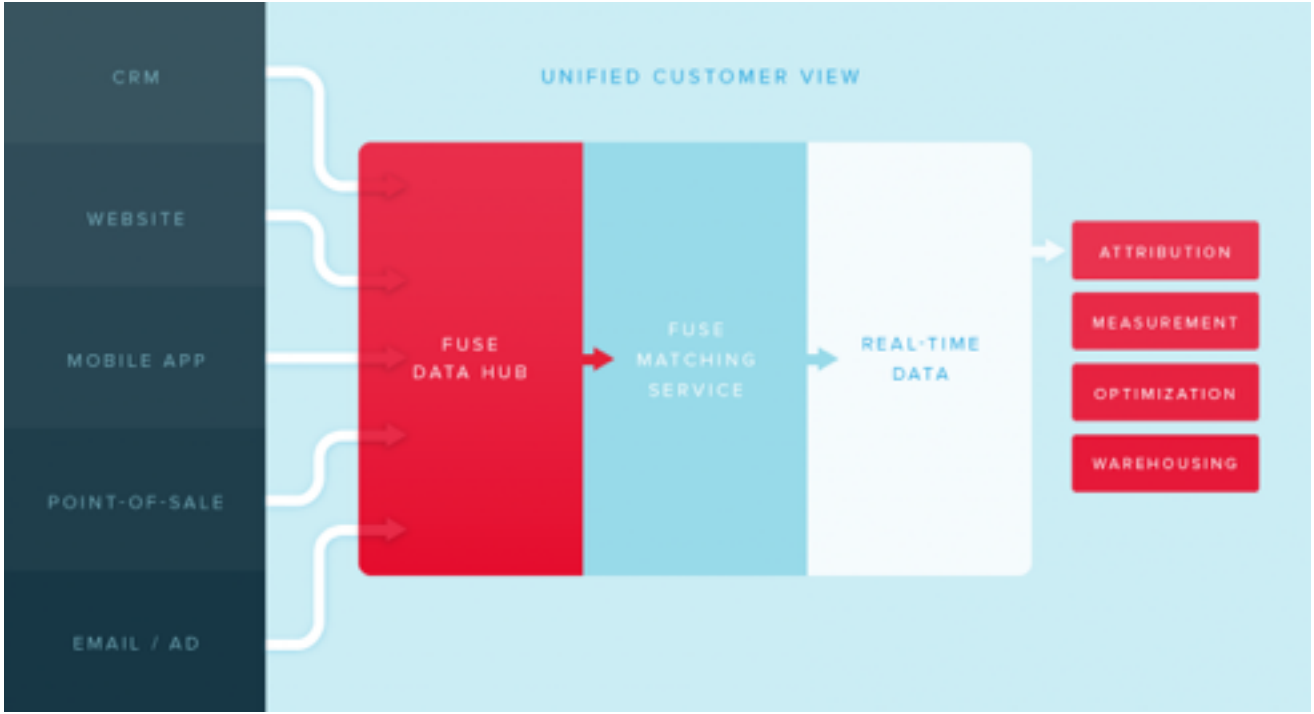
Aki Colovic
Lead Software Engineer

SIGNAL®

About Signal

- collect customer engagement data over across different channels(mobile, desktop, pos…)
- connect customer data and behaviors into identities
- send outbound real-time marketing signals(activation)

Signal's FUSE Platform



Presentation

- Cassandra @ Signal
- Wide Rows
  - How do we use wide rows?
  - Signal’s Identity service
  - Wide rows and compaction
  - Wide rows and caches
  - Don’t mix reads and writes
- Index table rebuild
- GC tuning
- Ring Migration
- Questions

Cassandra @ Signal

- Identity Service
- Activation Metrics

Cassandra Ring Stats

5 rings

190+ Nodes (AWS i2.xlarge)

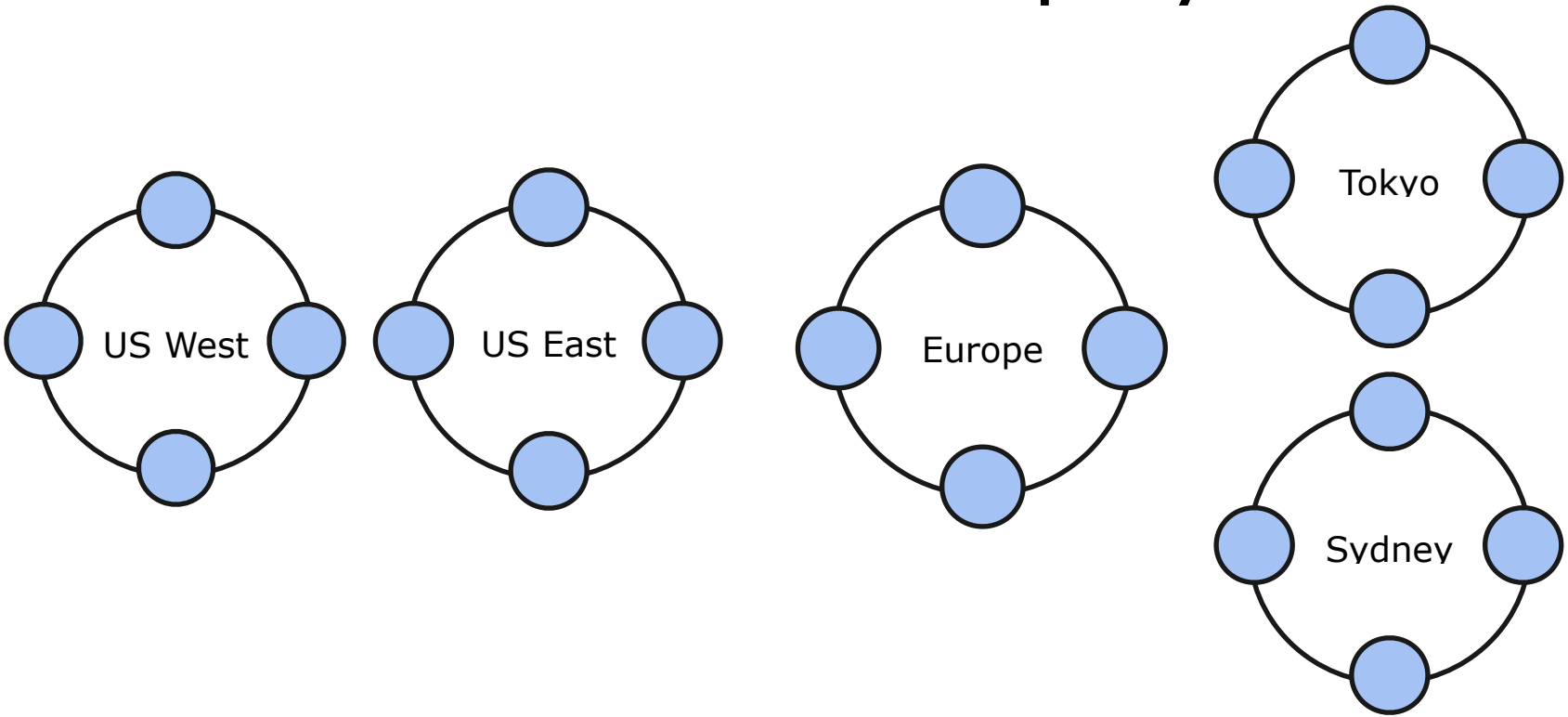
~2 billion queries per day

30+ TB of data

15B+ rows in profiles table

45B+ rows in index table

Global Cassandra Deployment



Dealing With Failure

We've survived …

- Amazon swallowing nodes
- Amazon rebooting the internet
- Disk Failures
- Corrupt SSTables
- Intra and inter region network issues

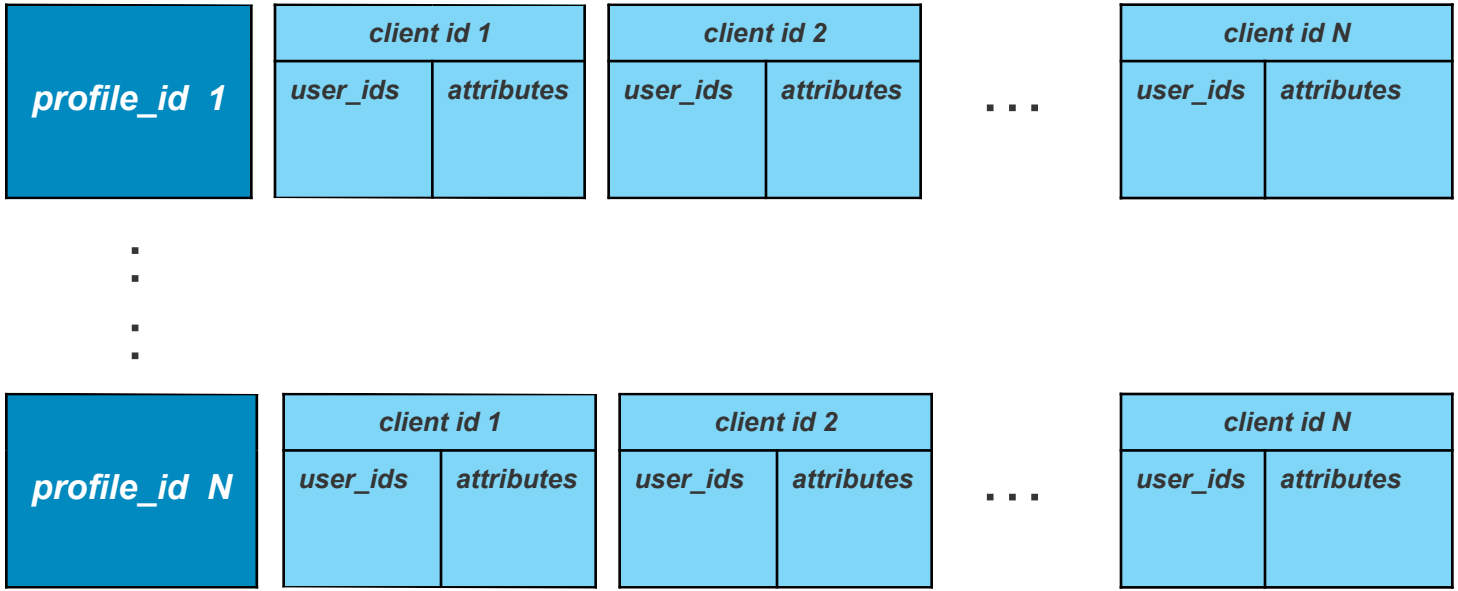
How do we use wide rows?

- store profile info
- build customer cross-channel identity
- identity graph
- client reporting

Profile Data Table

```
CREATE TABLE profile_data (
  profile_id uuid,
  client_id varchar,
  user_ids map<varchar, varchar>,
  attributes map<varchar, varchar>
  PRIMARY KEY (profile_id, client)
);
```

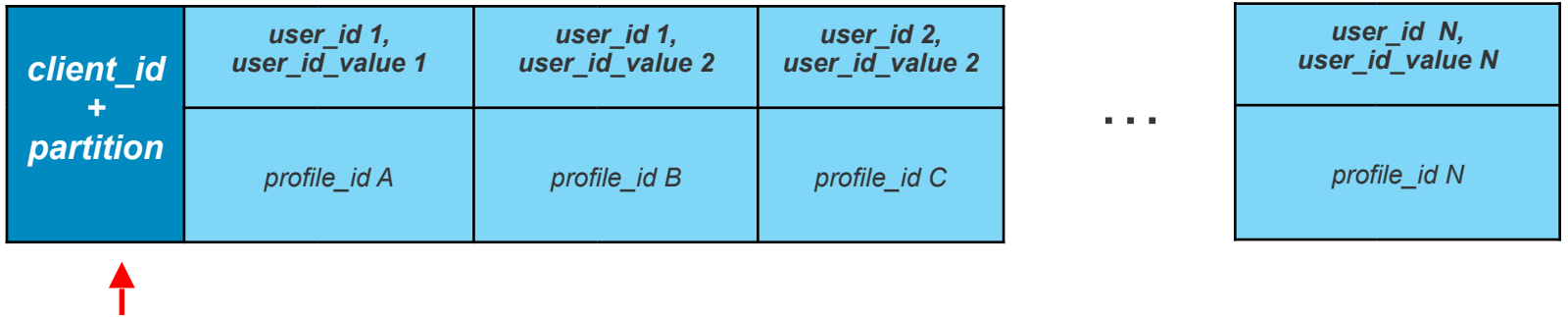
Profile Data Table



Profile Data Index Table

```
CREATE TABLE profile_data_index (
  client_id varchar,
  partition int,
  user_id varchar,
  user_id_value varchar,
  profile_id uuid,
  PRIMARY KEY
  ((client_id, partition), user_id, user_id_value)
);
```

Profile Data Index Table



Identity Graph

- multiple customer profiles are connected into single identity

- connect identities across different clients

Identity Graph Table

```
CREATE TABLE identity_graph (
  identity_id uuid,
  client varchar,
  profile_id map<uuid, timestamp>
  PRIMARY KEY (identity_id, client)
);
```

Identity Graph Table

identity_id	client id 1	client id 2	client id 3	...	client id N
	{profile id 1, profile id 2}	{profile id 4, profile id 5, profile id 6}	{profile id 7, profile id 8}		{set of profile ids}

Final thoughts on modeling

"The best way to approach data modeling for Cassandra is to start with your queries and work backwards from there. Think about the actions your application needs to perform, how you want to access the data, and then design column families to support those access patterns."

Wide Rows and Compactions

- size-tiered compaction
- slower compactions due to wide rows
- wide row limits (100MB or 100,000 elements)
- monitor wide row size
- monitor your sstable count
- *in_memory_compaction_limit_in_mb*,
- your widest row should be able to fit in memory
    - avoids slow 2 pass disk-based compaction
- set_compaction_throughput

Wide Rows and Caches

- caches are good
- unless your wide rows are too big
- be aware of your cache hits
- key_cache_size_in_mb
- row_cache_size_in_mb

Mixing Reads and Writes

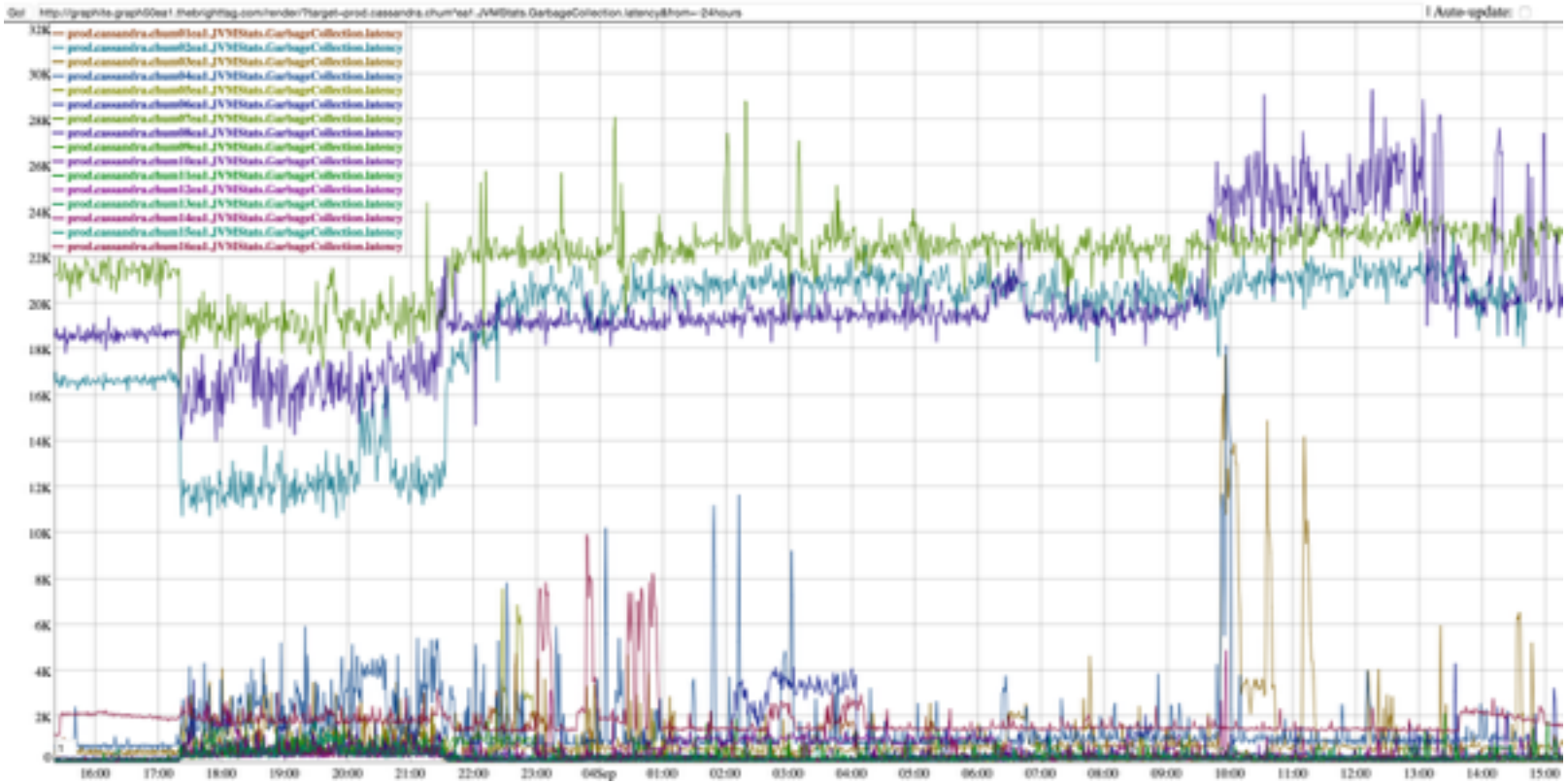
- slower reads can affect your FAST writes
- handle your read and writes paths separately
- if chaining reads and writes do it async

Index Rebuild

- it is awesome to grow
- our wide rows became too wide
- 256 partitions was not enough
- solution: rebuild the index with more partitions
- index re-builder

GC and Cassandra

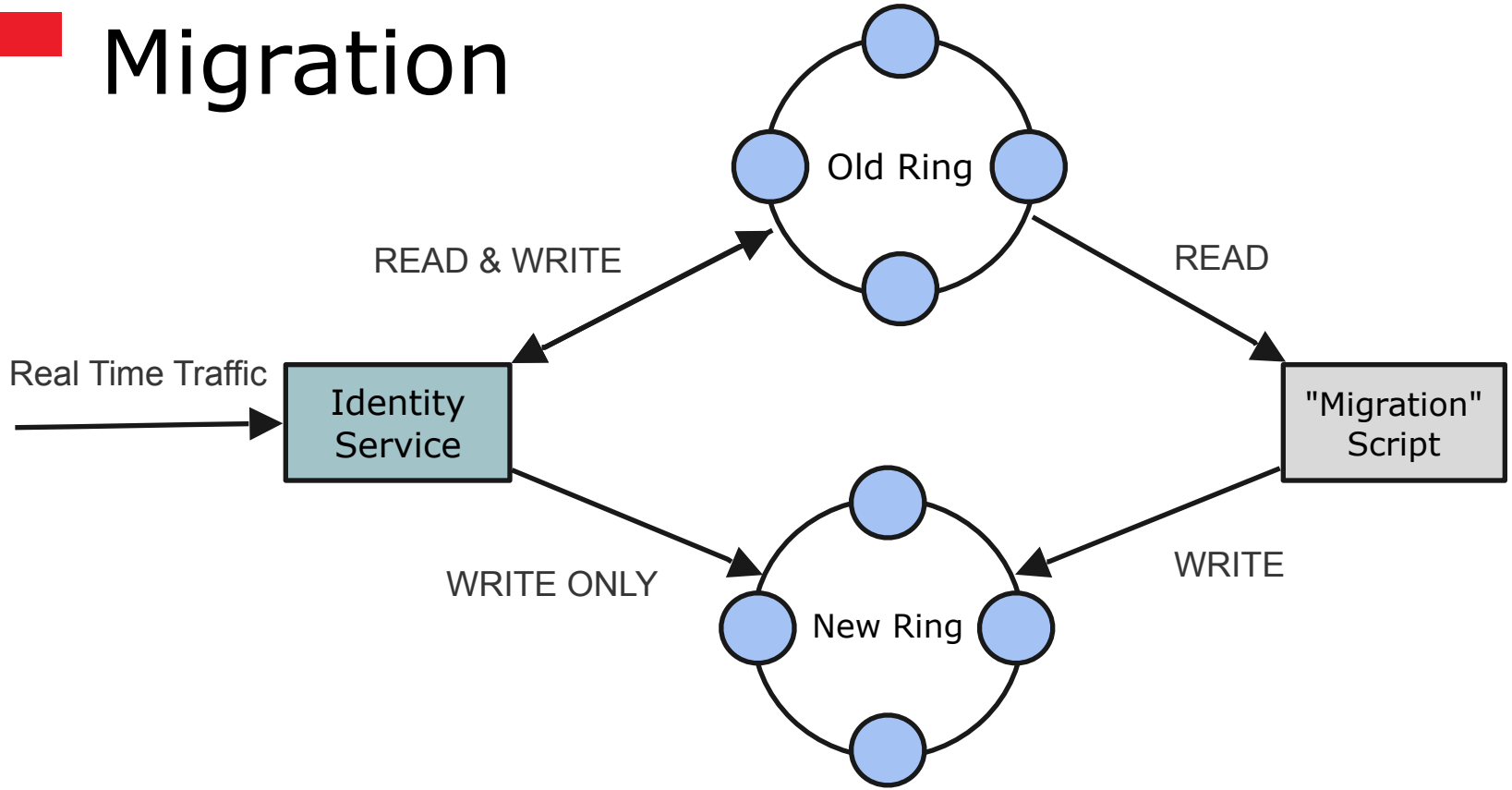
- we tuned our GC settings for low latency and to prevent full GC as much as possible
- know your GC settings
- monitor your GC latency - it has an impact on your cassandra performance
- monitor your traffic patterns during troubling GC latencies
- using CMSIncrementalMode to break up the concurrent GC phases into short bursts of activity
- every environment different, you will have to


Auto-update:
prod.cassandra.chum01ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum02ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum03ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum04ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum05ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum06ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum07ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum08ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum09ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum10ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum11ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum12ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum13ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum14ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum15ea1.JVMStats.GarbageCollection.latency
prod.cassandra.chum16ea1.JVMStats.GarbageCollection.latency
32K
30K
28K
26K
24K
22K
20K
18K
16K
14K
12K
10K
8K
6K
4K
2K
0
16:00
17:00
18:00
19:00
20:00
21:00
22:00
23:00
04Sep
01:00
02:00
03:00
04:00
05:00
06:00
07:00
08:00
09:00
10:00
11:00
12:00
13:00
14:00
15:00

Ring Migration/Upgrade

- upgrading to i2.2xlarge
- virtual nodes, increase replication factor

Migration


Old Ring
READ & WRITE
READ
Real Time Traffic
Identity Service
"Migration" Script
WRITE ONLY
WRITE
New Ring

CASSANDRA SUMMIT2015

Questions?

SIGNAL®

CASSANDRA SUMMIT2015

Contact Info

acolovic@signal.co

@akicolovic

CASSANDRA SUMMIT2015

SIGNAL®