Evolution of Cassandra at Signal

Matthew Kemp

Cut through the noise. www.signal.co

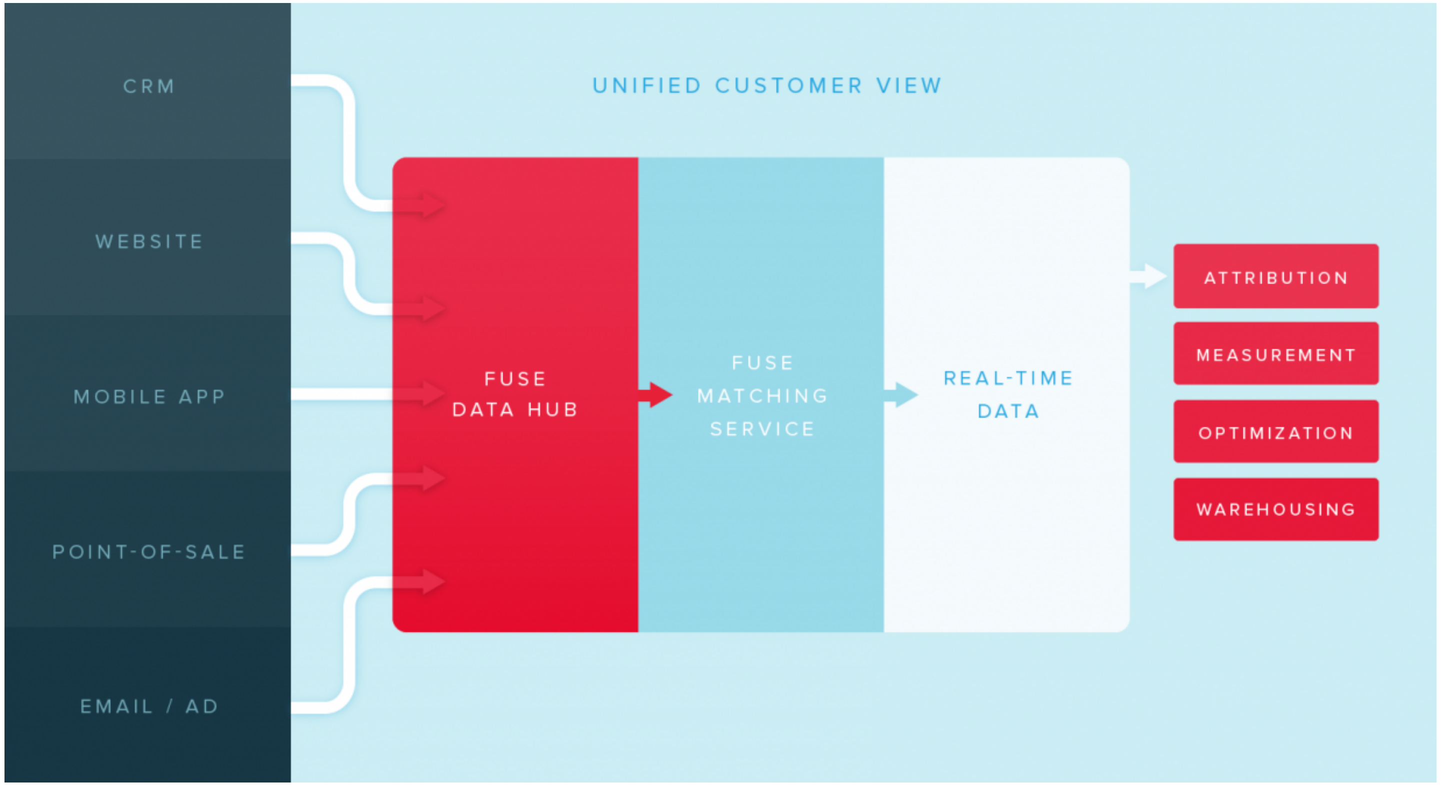
Our Cassandra Use Cases

Matching Service
- Unified Customer View
- Measurement and Activation

Metrics
- Using KairosDB and Cyanite

Audit Log
- Event persistence

Our First Use Cases



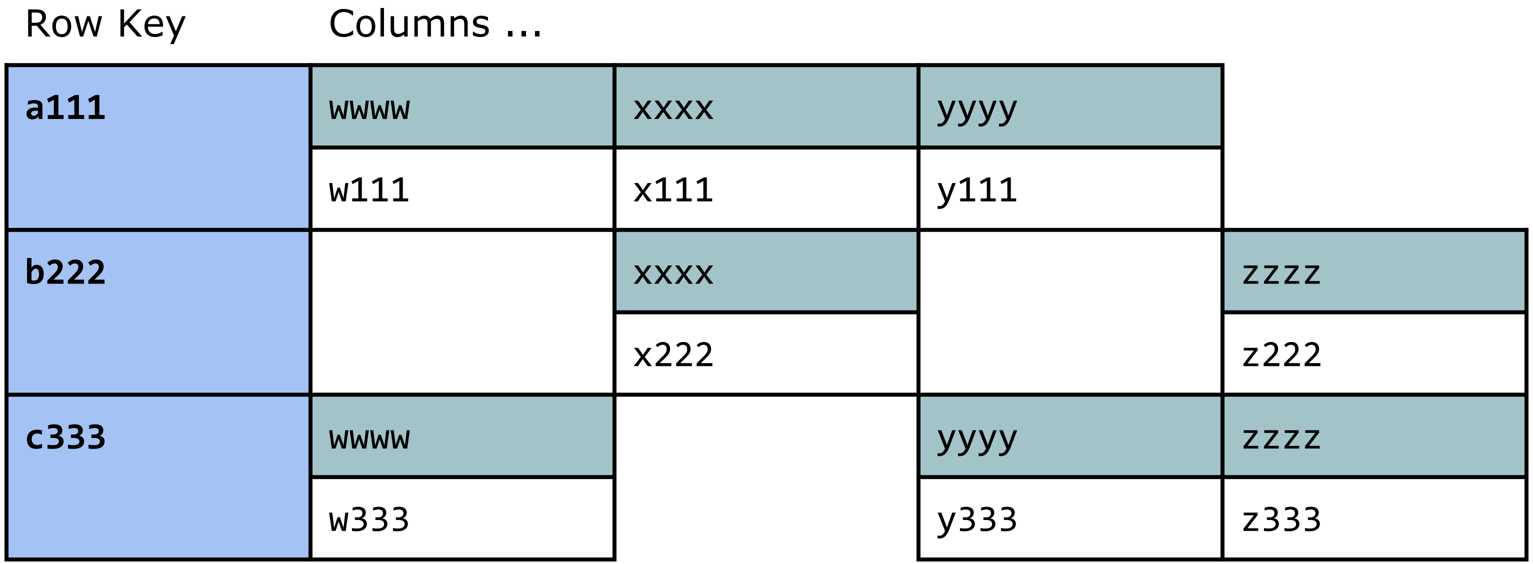
Cassandra Timeline: 2011

Spring 2011	Summer 2011	Fall 2011	Winter 2011
	Evaluated Cassandra 0.8, Couch, Mongo and Riak	Decided on Cassandra (1.0 just released)	Initial AWS deployment in two regions (8 total nodes)

Our Very First Schema

```
create column family signal_data
   with comparator = 'UTF8Type'
   and default_validation_class = 'UTF8Type'
   and key_validation_class = 'UTF8Type'
   and read_repair_chance = 0.0
   and gc_grace = 864000;
```

Example Data Rows

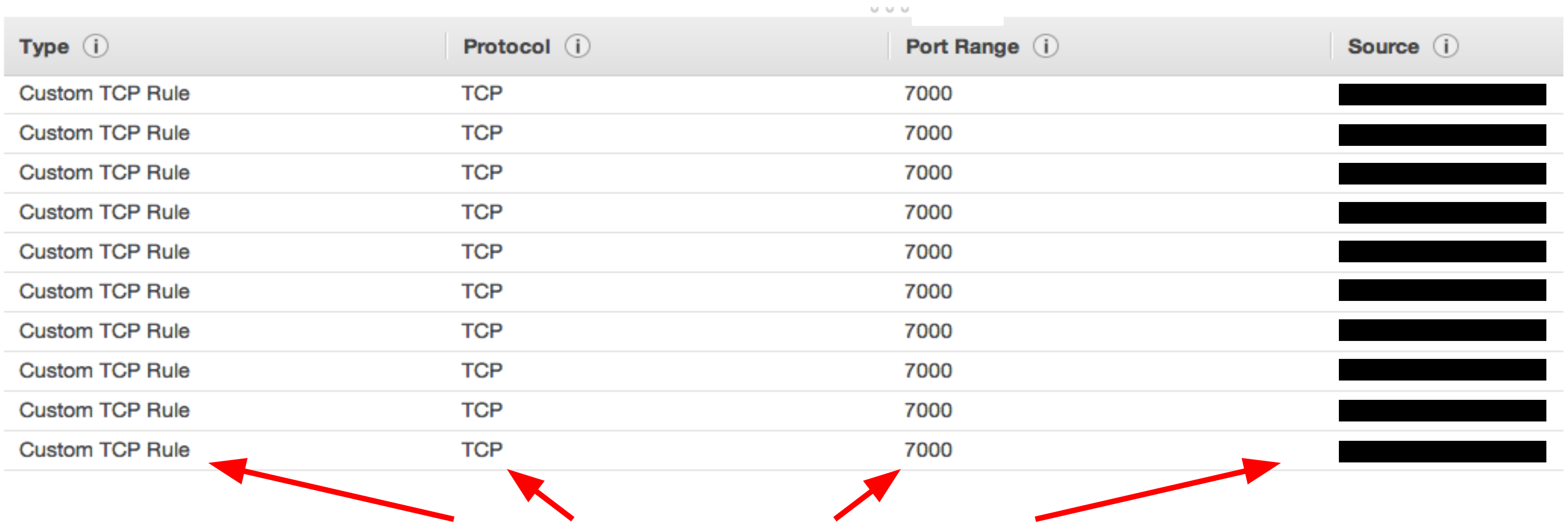


Row Key	Columns ...			
a111	wwww	xxxx	yyyy	
	w111	x111	y111	
b222		xxxx		zzzz
		x222		z222
c333	wwww		yyyy	zzzz
	w333		y333	z333

Cassandra Timeline: 2012

Spring 2012	Summer 2012	Fall 2012	Winter 2012
Attempted to use SuperColumns and started using Priam	Upgraded to Cassandra 1.1 and start of production usage	Attempted to use Composite Columns and switched to m2.xlarges	Added index column family

Priam



Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	7000	[redacted]
Custom TCP Rule	TCP	7000	[redacted]
Custom TCP Rule	TCP	7000	[redacted]
Custom TCP Rule	TCP	7000	[redacted]
Custom TCP Rule	TCP	7000	[redacted]
Custom TCP Rule	TCP	7000	[redacted]
Custom TCP Rule	TCP	7000	[redacted]
Custom TCP Rule	TCP	7000	[redacted]
Custom TCP Rule	TCP	7000	[redacted]
Custom TCP Rule	TCP	7000	[redacted]

Because nobody wants to deal with this

Pre-CQL Data Types

Row Key	Columns ...			
a111	wwww:id	wwww:attr_1	wwww:attr_2	xxxx:id
	w111	value1	value2	x111
b222				xxxx:id
				x222
c333	wwww:id		wwww:attr_2	
	w333		something	

Example Index Rows

Row Key	Columns ...
wwww:w111	sid
	a111
xxxx:x111	sid
	a111
yyyy:y111	sid
	a111

... continued ...	
xxxx:x222	sid
	b222
zzzz:z222	sid
	b222
wwww:w333	sid
	c333

Cassandra Timeline: 2013

Spring 2013

Summer 2013

Performance testing, evaluating 1.2.x, switched to Agathon

Fall 2013

Upgraded to Cassandra 1.2.9 and schema redesign

Winter 2013

Upgraded to Virtual Nodes and started using OpsCenter

Keyspace

```
CREATE KEYSPACE signal WITH replication = {
  'class': 'NetworkTopologyStrategy',
  'eu-west': '2',
  'ap-northeast': '2',
  'us-west': '2',
  'us-east': '2'
};
```

Data Table

```
CREATE TABLE signal_data (
  sid uuid,
  identifier varchar,
  ids map<varchar, varchar>,
  data map<varchar, varchar>,
  internal map<varchar, varchar>
  PRIMARY KEY (sid, identifier)
);
```

Example Data Rows

sid	identifier	ids	data	internal
a111	**wwww**	{'id':'w111'}	{'attr_1':'value1'}	{}
a111	**xxxx**	{'xid':'x111'}	{'attr_2':'value-a'}	{}
a111	**yyyy**	{'id':'y111'}	{}	{}
b222	**xxxx**	{'xid':'x222'}	{'attr_2':'value-b'}	{}
b222	**zzzz**	{'zid':'z222'}	{}	{}

Index Table

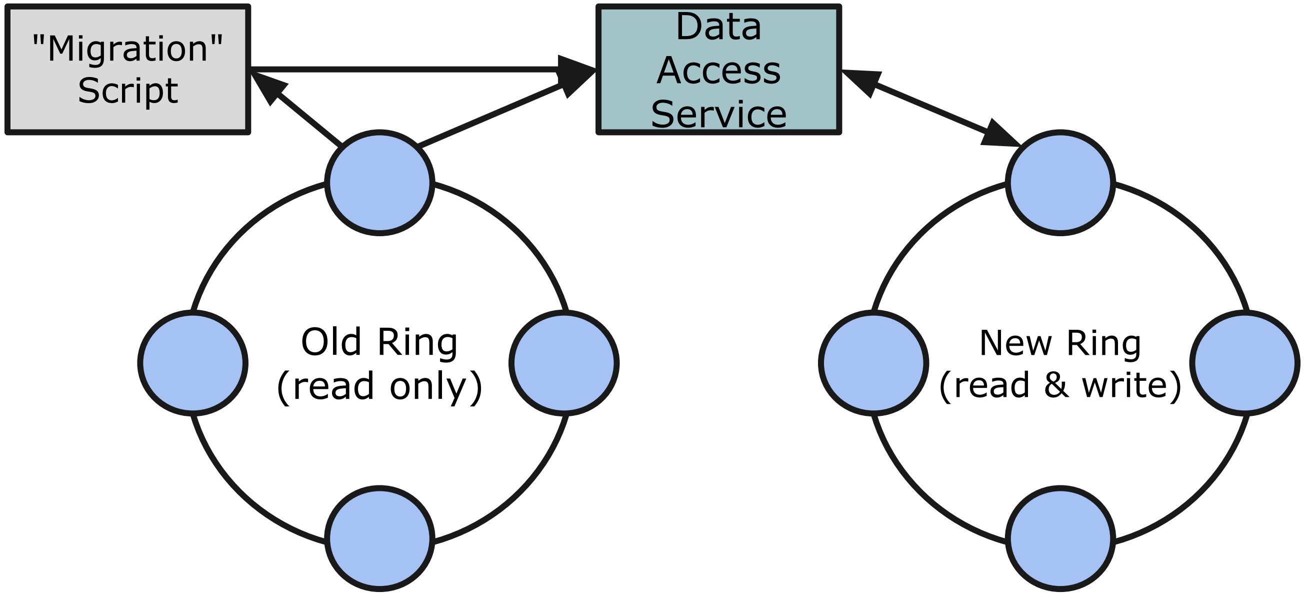
```
CREATE TABLE signal_index (
  identifier varchar,
  partition int,
  id_name varchar,
  id_value varchar,
  sid uuid
  PRIMARY KEY (
    (identifier, partition), id_name, id_value)
);
```

Example Index Rows

identifier	partition	id_name	id_value	sid
wwww	128	id	w111	a111
xxxx	84	xid	x111	a111
yyyy	71	id	y111	a111
xxxx	193	xid	x222	b222
zzzz	3	zid	z222	b222

partition = abs(hash(id_name + id_value)) % partitions

Migration To Virtual Nodes



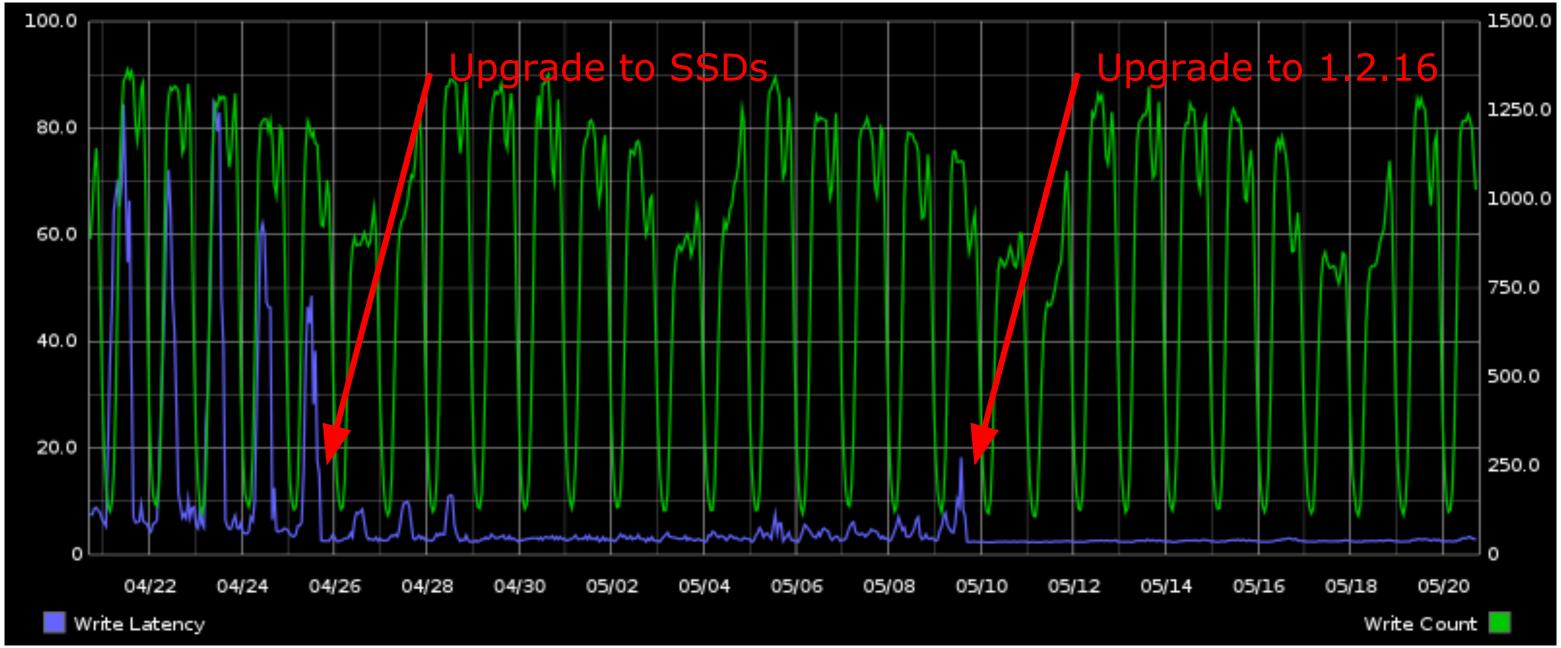
Cassandra Timeline: 2014

Spring 2014	Summer 2014	Fall 2014	Winter 2014
Upgraded to i2. xlarges and Cassandra 1.2.16	Schema improvements and KariosDB backed by Cassandra	Scaling cluster	

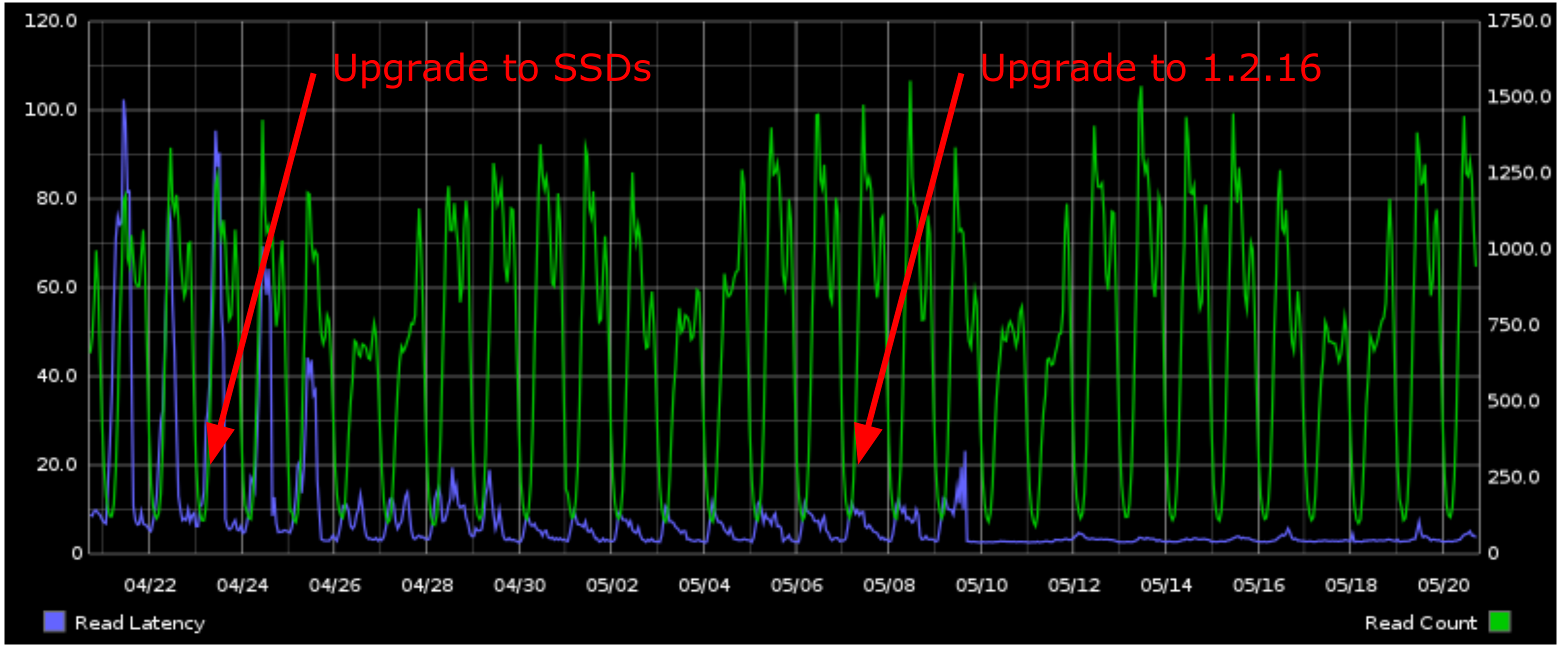
AWS Instance Types

Instance Type	ECUs	Virtual CPUs	Memory	Disks
m2.xlarge	6.5	2	17.1	1x420GB
m2.2xlarge	13	4	34.2	4x420GB
i2.xlarge	14	4	30.5	1x800GB SSD

SSD Performance: Writes



SSD Performance: Reads

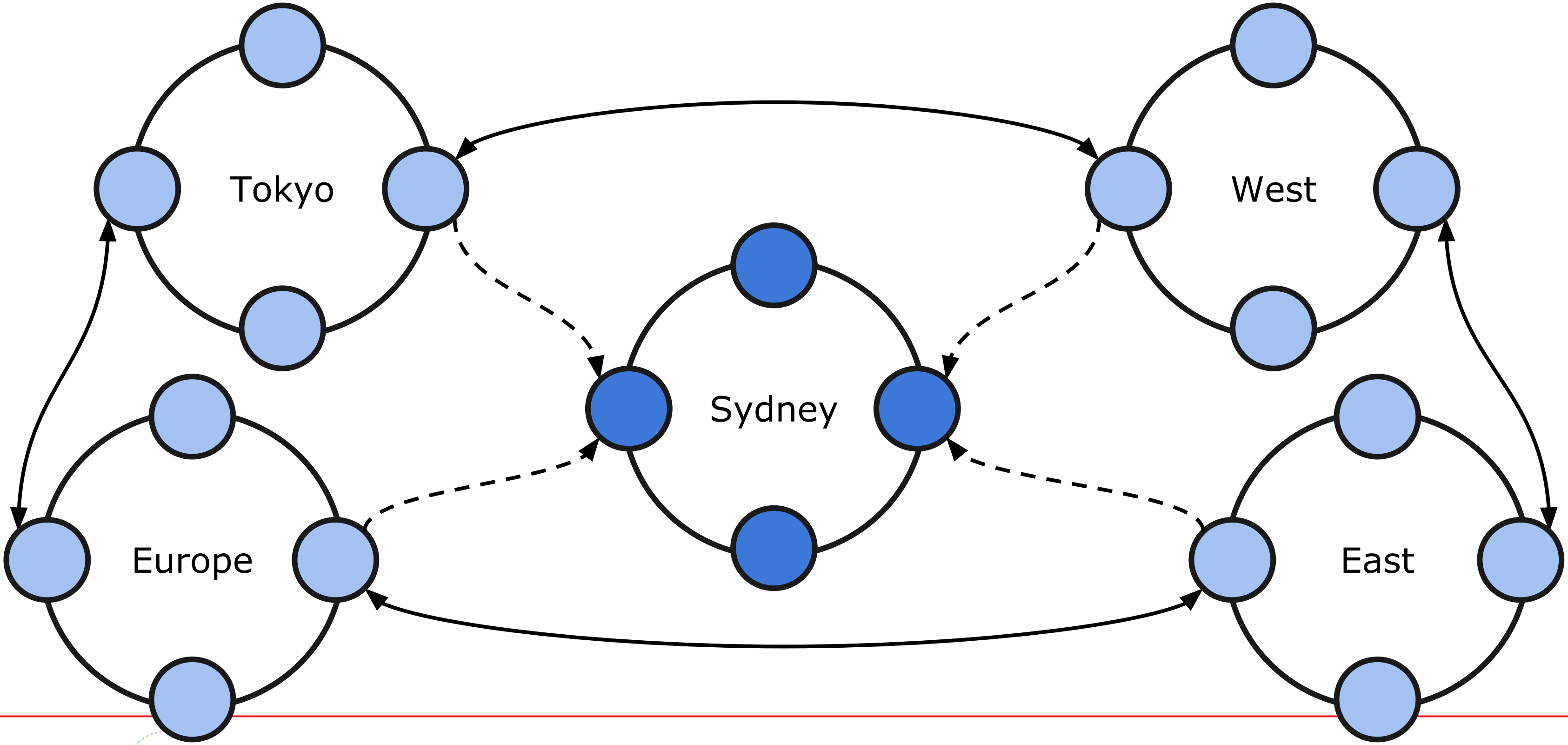


SIGN

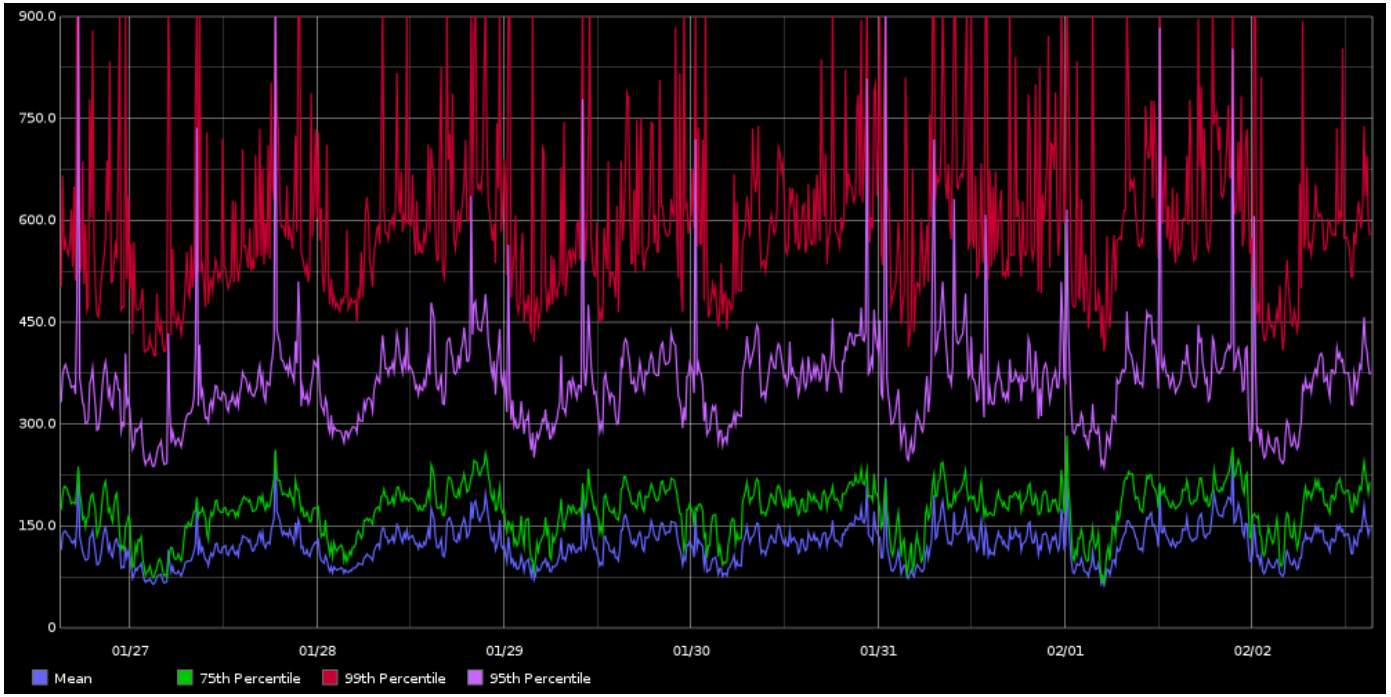
Cassandra Timeline: 2015

Spring 2015	Summer 2015	Fall 2015	Winter 2015
Added a new AWS region and rolled out new audit log use case (separate ring)	Planned schema redesign and focus on performance (latency)	Metrics and reporting (Spark?)	???

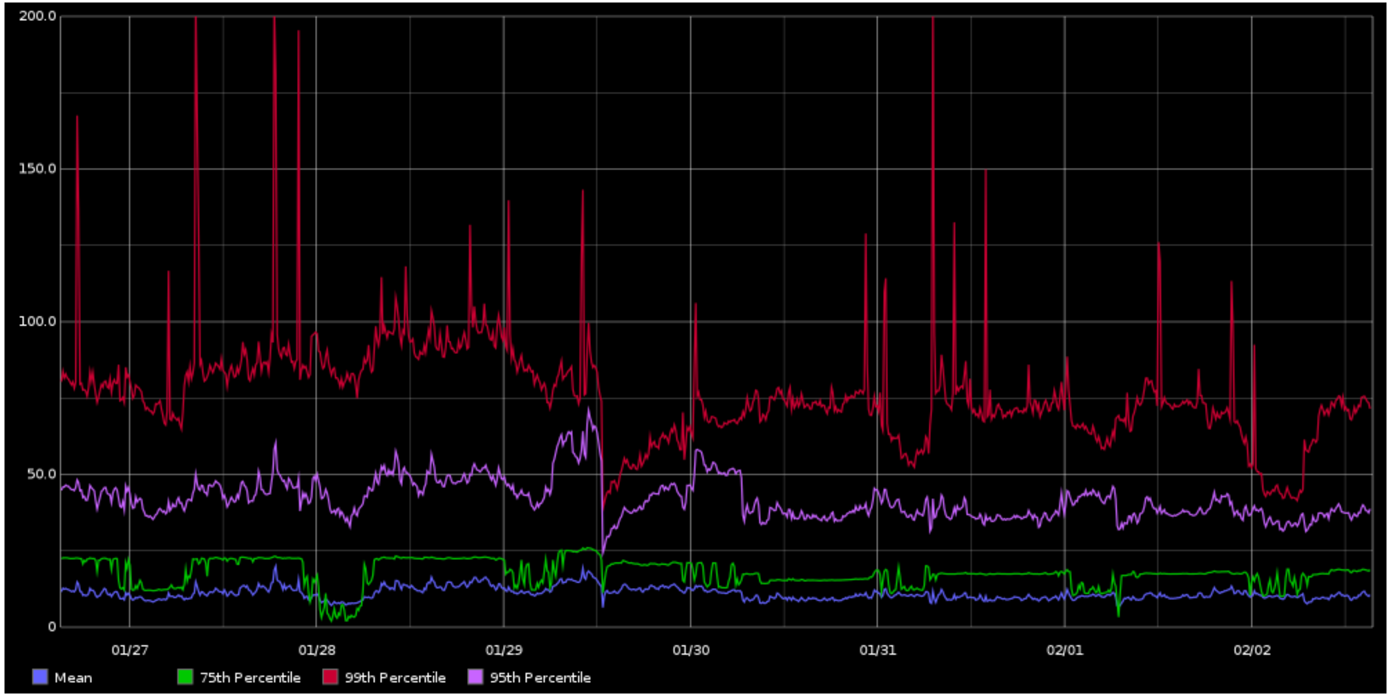
Adding a New Region



Reduce Latency: Writes



Reduce Latency: Reads



Metrics and Reporting

Currently run as a set of ad hoc python scripts

Considering adding reporting tables
- Centered around set membership and overlap

Considering using Spark as job runner

Some Ring Stats

140+ Nodes (i2.xlarge)

15B+ rows in data table

45B+ rows in index table

30+ TB of data

Dealing With Failure

We've survived …

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

Advice

Use SSDs, don't skimp on hardware

Stay current, but not too current

Keep up on repairs

Test various configurations for your use case

Simulating production load is important

The Best Advice

"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."

Questions?

SIGNAL®

Cut through the noise. www.signal.co

Contact Info

mkemp@signal.co

@mattkemp

/in/matthewkemp

SIGNAL®

Cut through the noise. www.signal.co