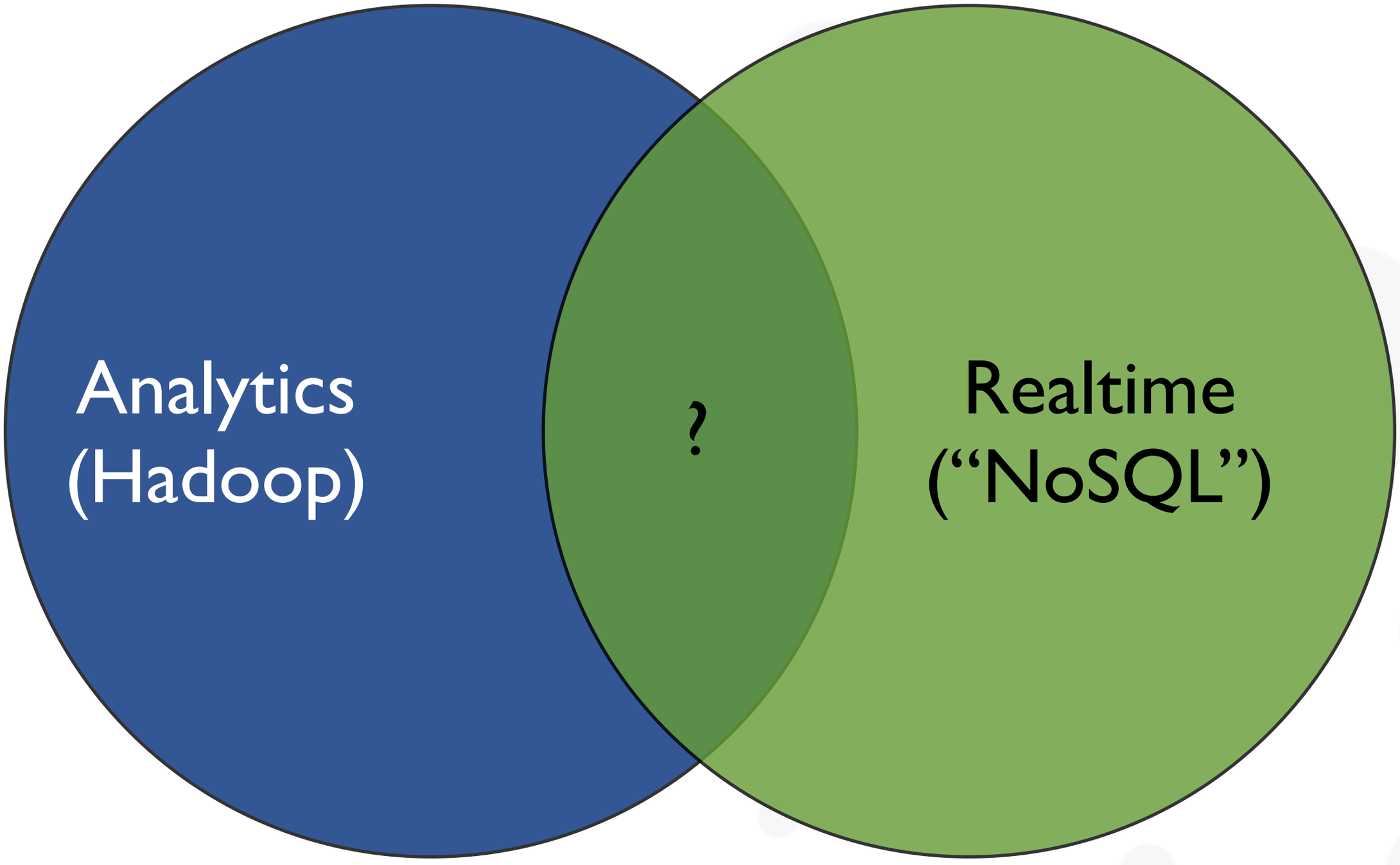
Apache Cassandra in Action

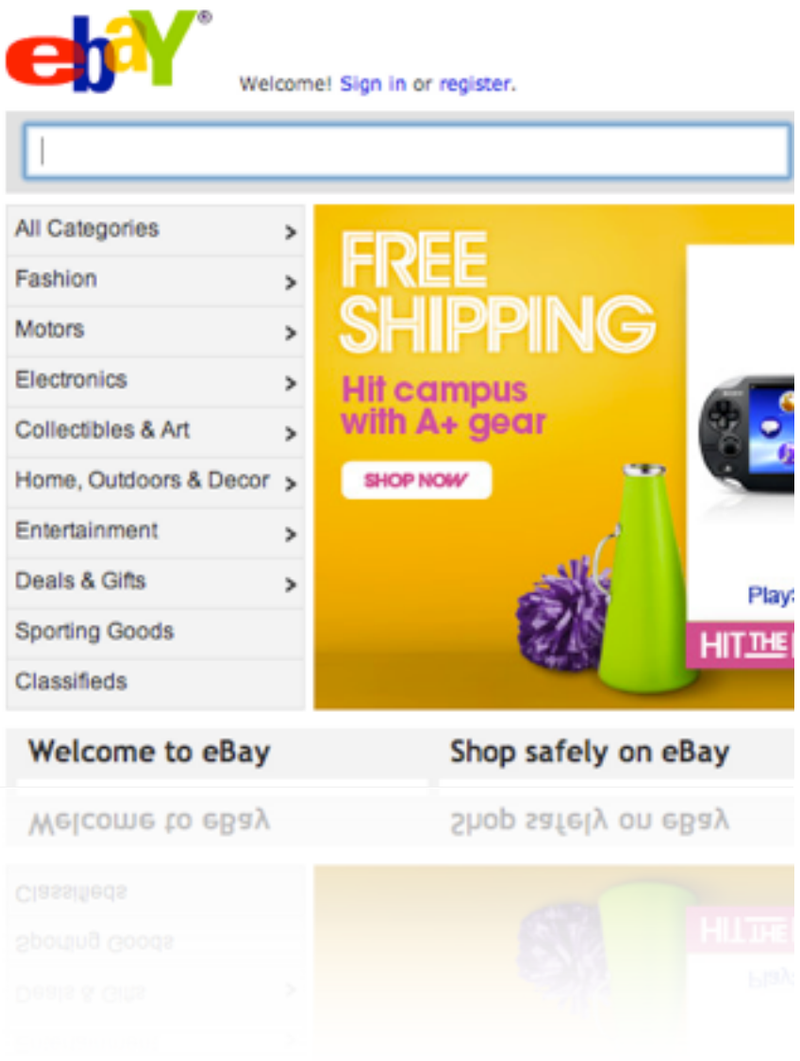
Jonathan Ellis
Project Chair, Apache Cassandra
CTO, DataStax
@spyced

Big data




1 9ths 2 9ths .com
People, Businesses & Places
HOUGHTON MIFFLIN HARCOURT
MORNINGSTAR
platform46
adform
adku
Adobe
Aeris
AGENTIS
aiMatch
ad intelligence
apigee
APIXIO
applifier
AppScale
appssavvy
Apptegic
ARKIVUM
ASSURED ARCHIVING
AZUL SYSTEMS
backupify
Secure your online life
Burt.
CAKESOLUTIONS
CallFire
citypath
YOUR CITY, YOUR WAY
*clearspring
Cloudian
cloudkick
CLOUDTALK
Constant Contact
Connect. Inform. Grow.
DachisGroup
DATADOG
Datagraph, Inc.
Demandforce
Growth. On Demand.
despegar.com
digg
DIGITAL REASONING
Digital River
easou 宜搜
ebuddy
egnyte
Cloud File Server
embed.ly
Engine Yard
enkata
ERICSSON
FanBridge
Fright My Monster
FINN
FLXONE
formspring
Frugal Mechanic
GoDaddy.com
goingon
GOTiME.com
Gowalla
GRIDPOINT
Healthline
Connect to Better Health
HEALTH MARKET SCIENCE
HealthX
hobsons
hp
HubSpot
IBM
impetus
Partners in Software R&D and Engineering
inkling
iovation
ISIDOREY
jetpac
Jumptap
junctionnetworks
business communications
JustSignal
lacuna
SYSTEMS
LeaseEagle
librato
LiveOffice
Your Email Experts
LIVEPERSON
Mahalo
martinimedia
mate1.com
find someone
media6degrees
MEDIALETS
MedProcure
meta broadcast
MetricsHub
MolloM
monitis
monitor everything
movile
napshare
NAVTEQ
nearbuy
NETFLIX
Nextag
NEXT BIG SOUND
NextSlide
Better online presentations
ngmoco:)
nicira
nSCALED
nTokio
одноклассники
ONEsite
one spot
OOYALA
OpenAdExchange
OpenFeint
opentracker
event tracking and analytics
OPENWAVE
OpenX
Other inbox
outbrain
page2rss
Palantir
PANTHEON
PitneyBowes
PBS KIDS
PCPhase
PeopleBrowsr
PERFECT WORLD
plaxo
POLYVORE
PostRank
PRESS+
proofpoint
Control tomorrow's email risks today
quantiFind
QUOVA
a Neustar service
rackspace
Rakuten
RALLY SOFTWARE
RAZORlabs
ReachLocal
real
reddit
referentia
Reltio
retickr
Rhapsody
RIGHTSCALE
CLOUD MANAGEMENT
RockMelt
rockyou
Ruckus
WIRELESS
Scandit
SEOmoz
SEVEN
shazam
shopsavvy
SimpleGeo
simplereach
skillpages
smeet
SOCIALFLOW
SPI
SOFTWARE PROJECTS INC
sonico
SOUNDCLOUD
SourceNinja
spidertracks
Spotify
Spredfast
springpad
SQUIDOO
Symantec
Tapulous
TEADS
Tellybug
TENDRIL
thefind
trap!t
TRIPLE POINT TECHNOLOGY
TWIMPACT
twitter
Urban Airship
powering modern mobile
V2i
verticalscope
VigLink
Visual DNA
VTEX
Remarkable Technology
wajam
WeRWhere
WILDFIRE
WORKWARESYSTEMS
WSO2
Wupa
Simple website manager
xobni
Yakaz
YCHARTS
Zemanta
ZIPZAPPLAY
zoemob
zoominfo

eBay



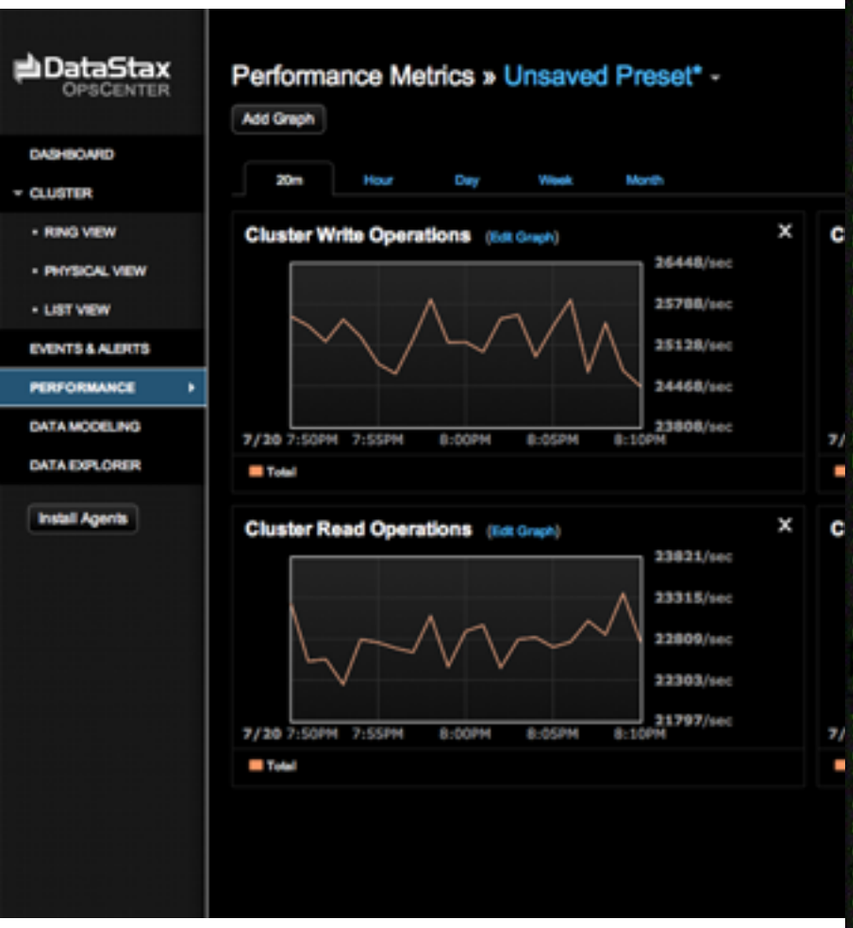
Application/Use Case

- Social Signals: like/want/own features for eBay product and item pages
- Hunch taste graph for eBay users and items
- Many time series use cases

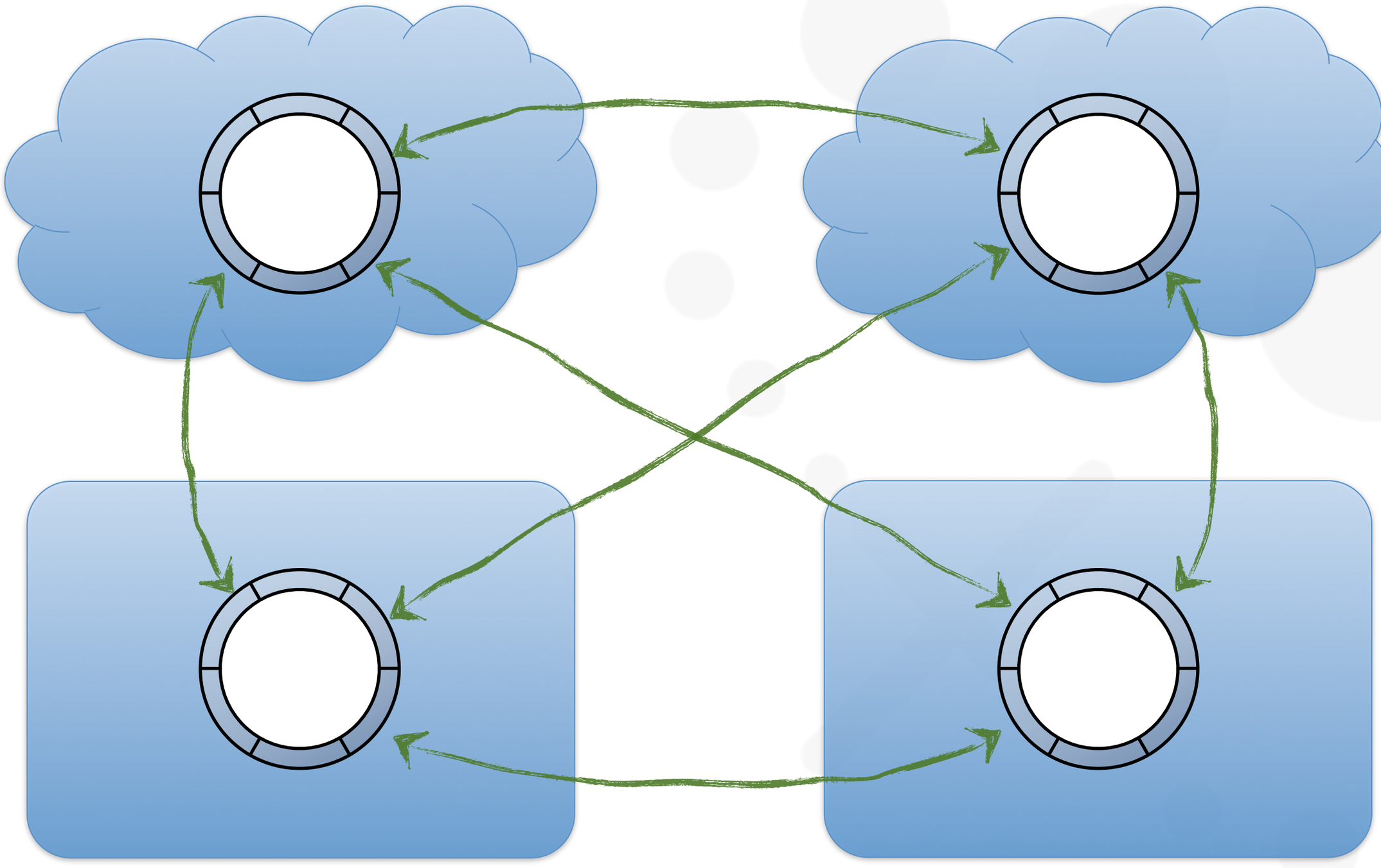
Why Cassandra?

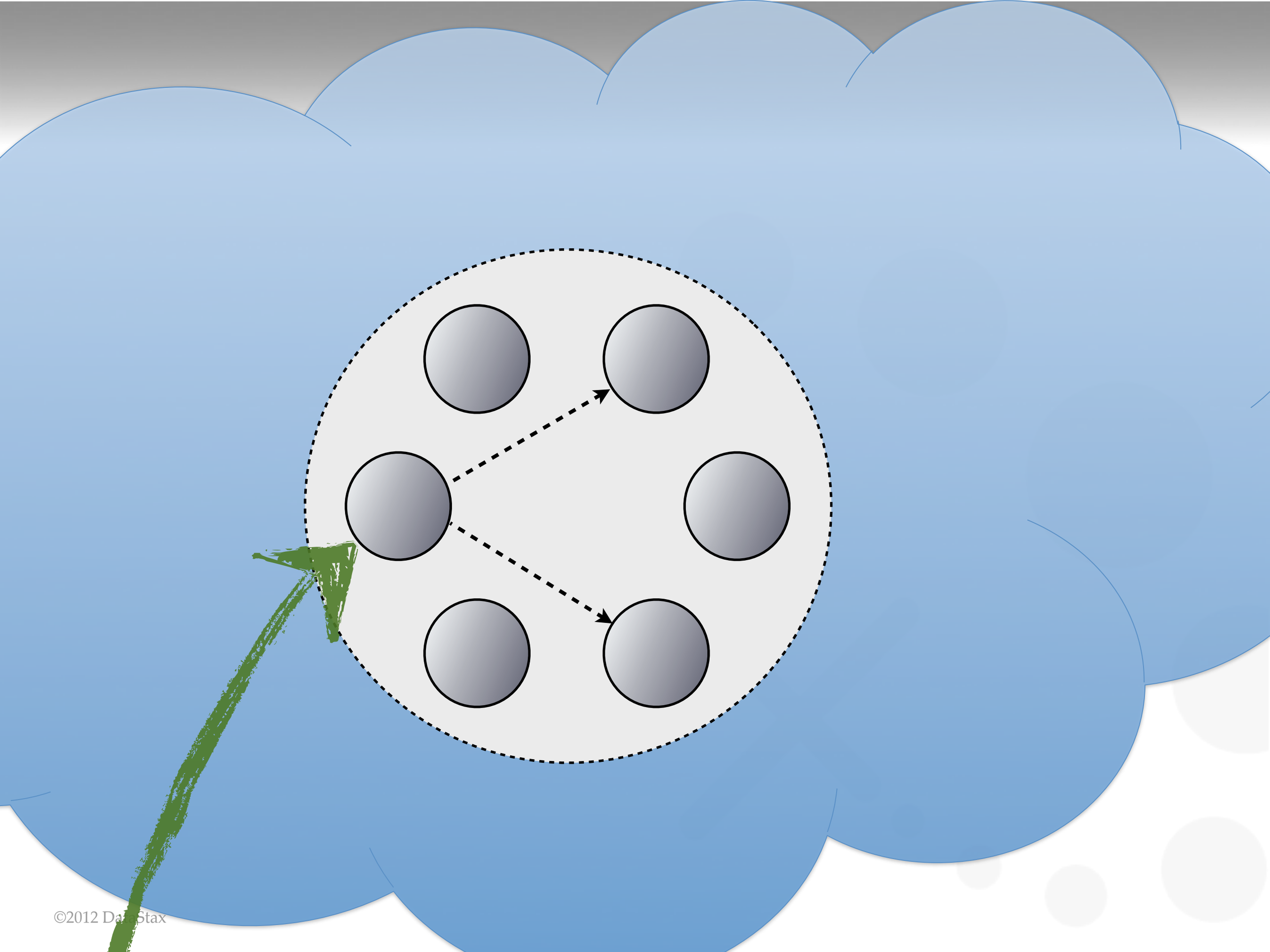
- Multi-datacenter
- Scalable
- Write performance
- Distributed counters
- Hadoop support

Time series data

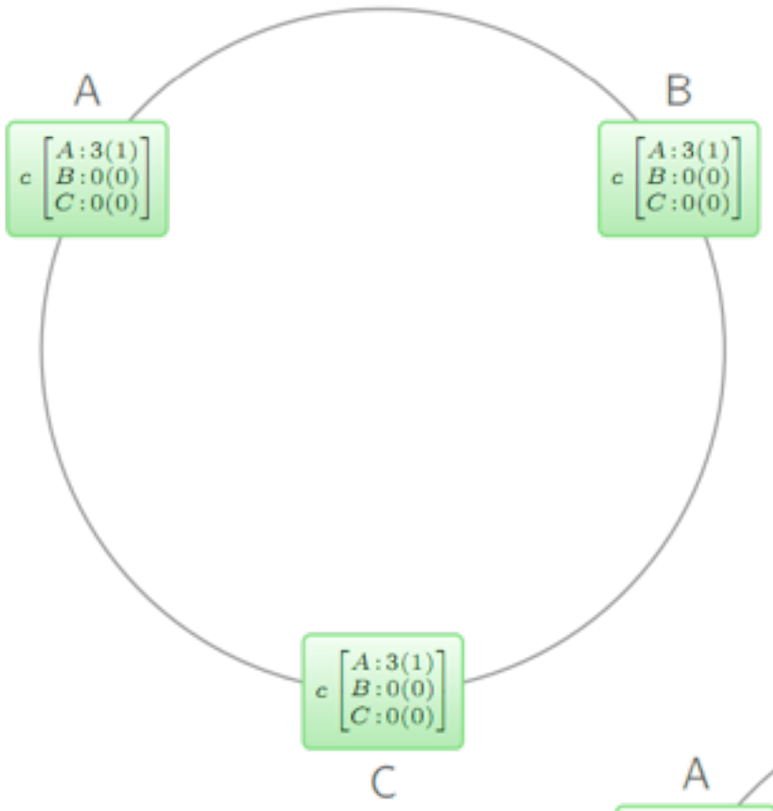


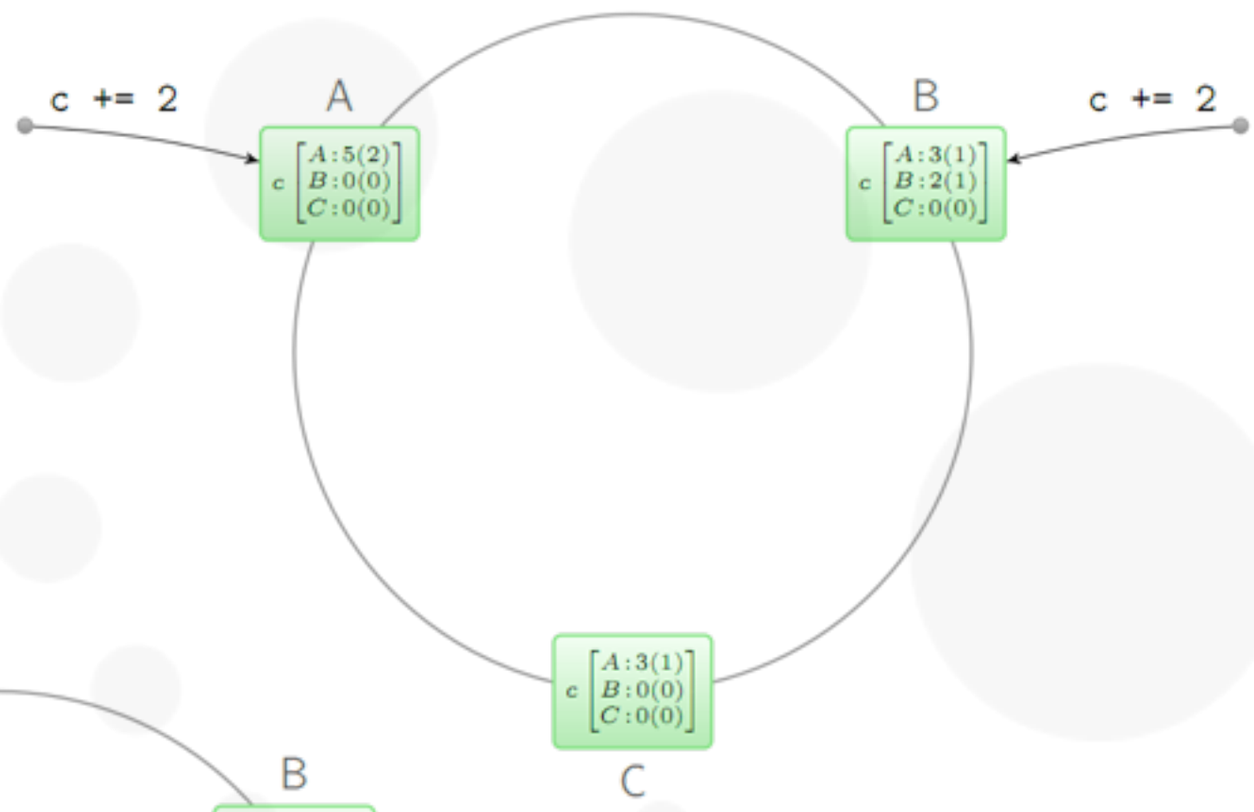
Multi-datacenter support

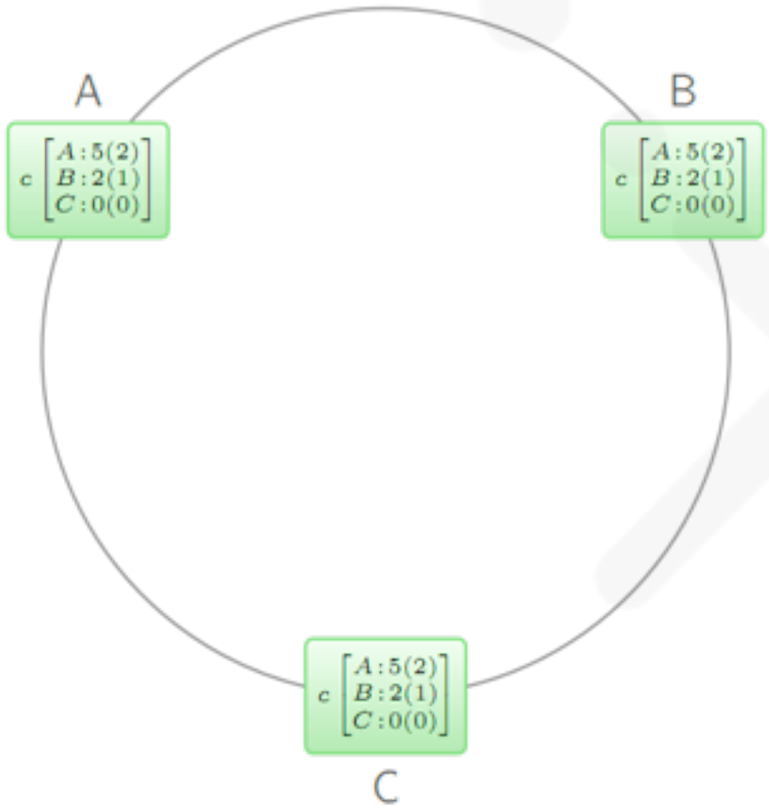




Distributed counters







Hadoop support

Disney

Application/Use Case

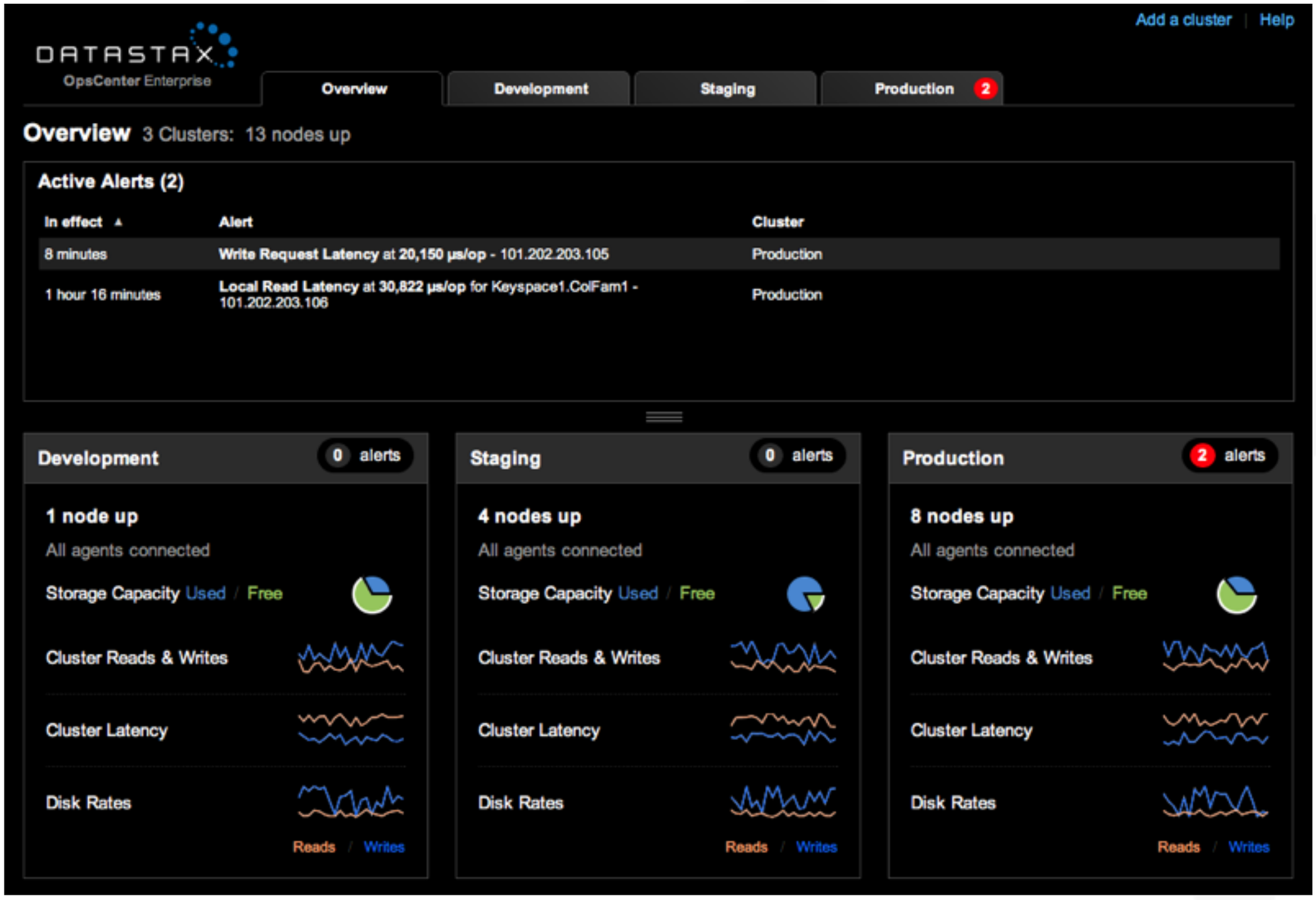
- Meet the data management needs of user facing applications across The Walt Disney Company with a single platform

Why Cassandra?

- DataStax Enterprise can tackle real-time and search functions in the same cluster
- Scalability
- 24x7 uptime

Multitenancy

Multitenancy



Enterprise search

Documents in Solr Core : Users

Field	Value
user	jake
state	CT
status	at work

Cassandra ColumnFamily : Users

row_key	state	status
jake	CT	at work
jason	NY	at home
jonathan	TX	in bed

Field	Value
user	jason
state	NY
status	at home

Field	Value
user	jonathan
state	TX
status	in bed

Netflix

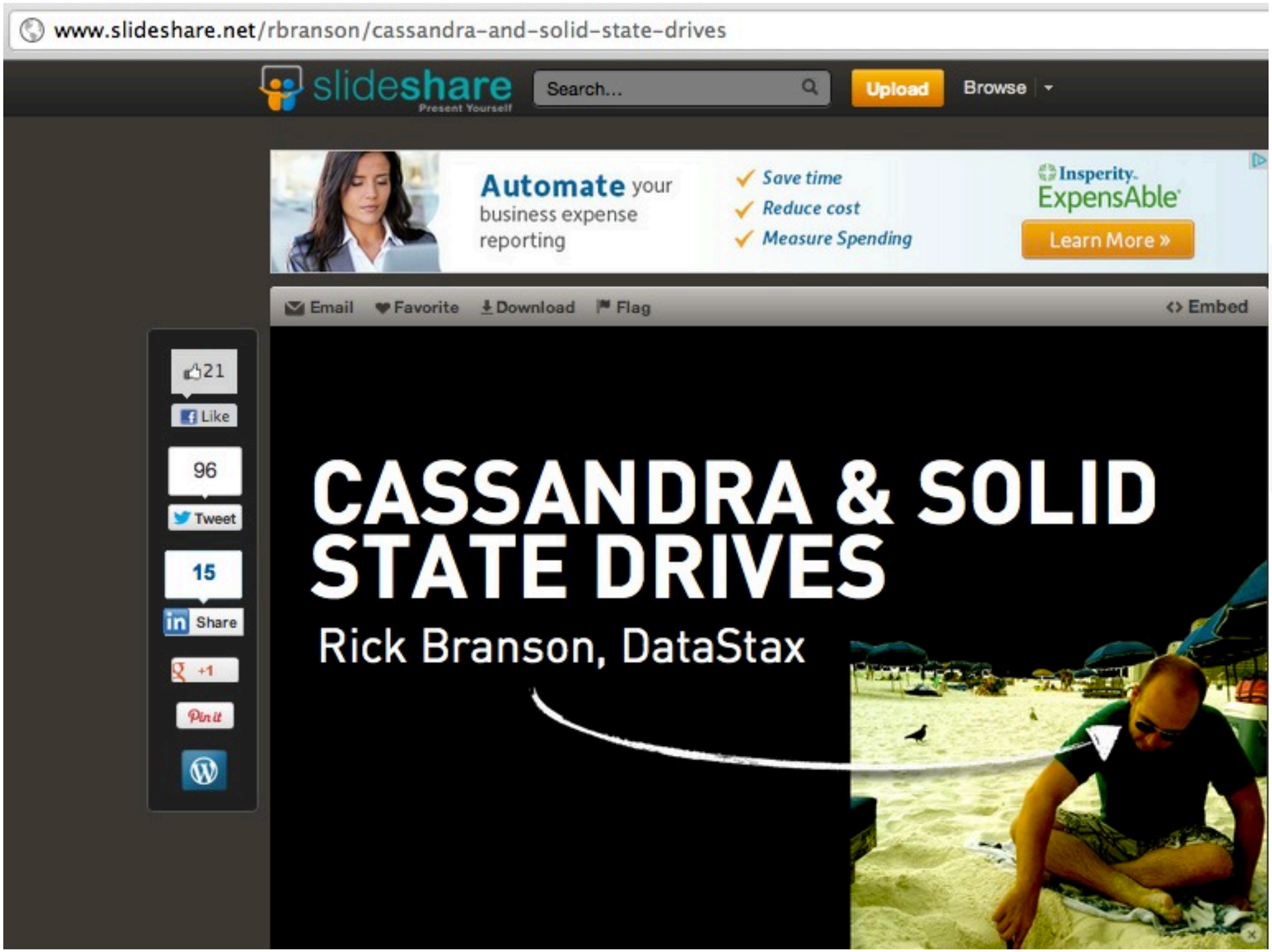
Application/Use Case

- General purpose backend for large scale highly available cloud based web services supporting Netflix Streaming

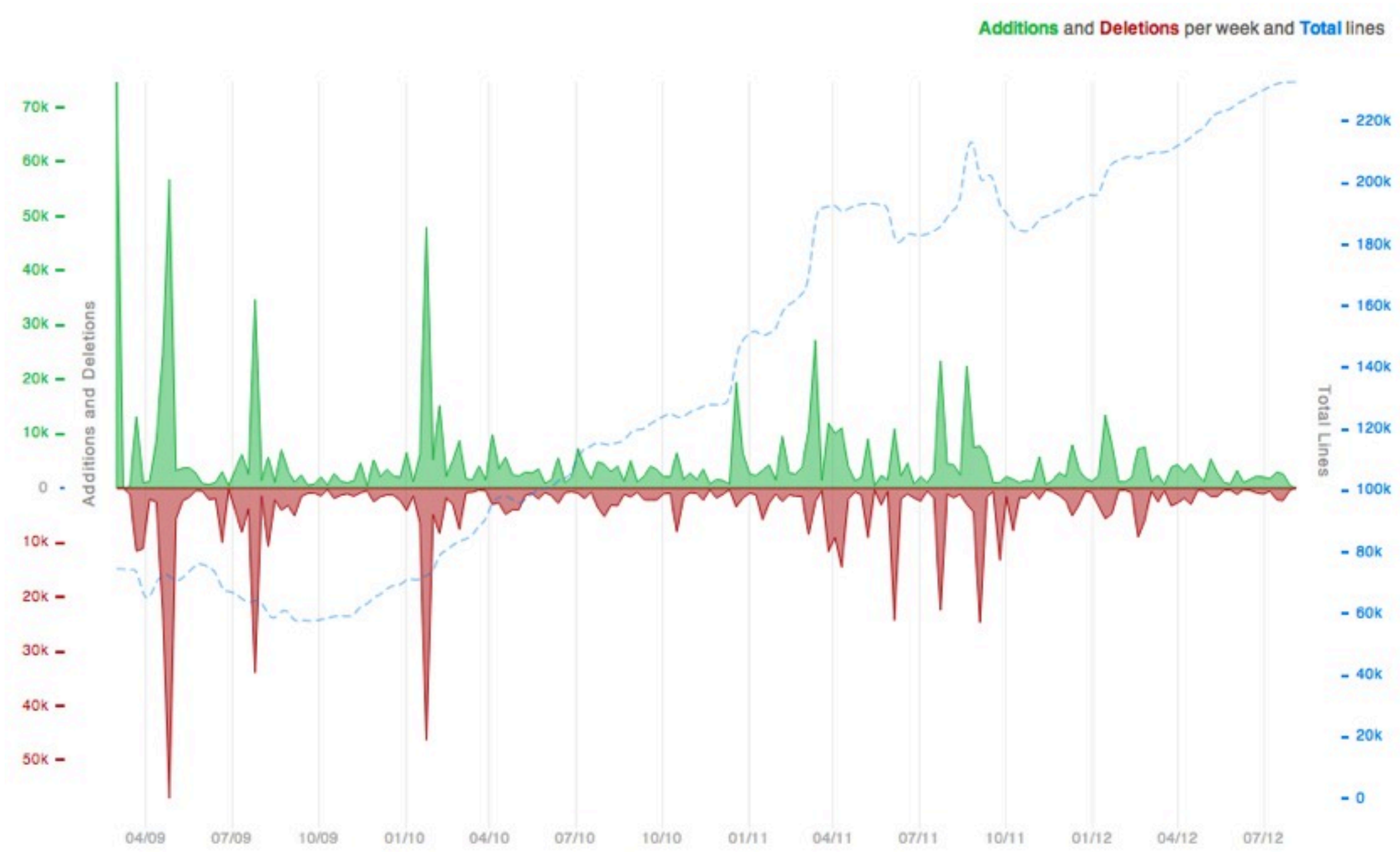
Why Cassandra?

- Highly available, highly robust and no schema change downtime
- Highly scalable, optimized for SSD
- Much lower cost than previous Oracle and SimpleDB implementations
- Flexible data model
- Ability to directly influence/implement OSS feature set
- Supports local and wide area distributed operations, spanning US and Europe

Optimized for SSD

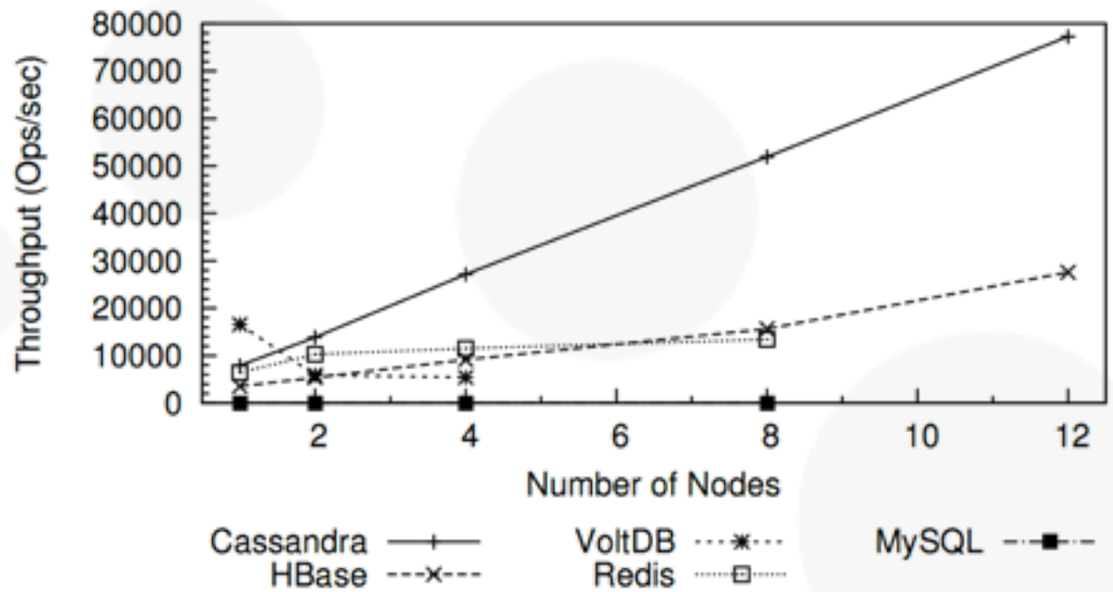


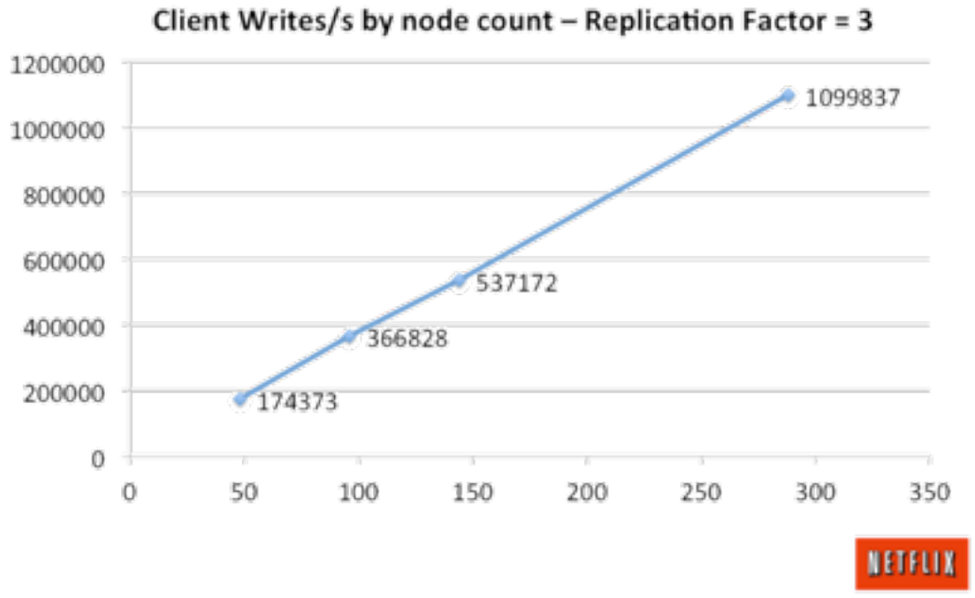
Open source

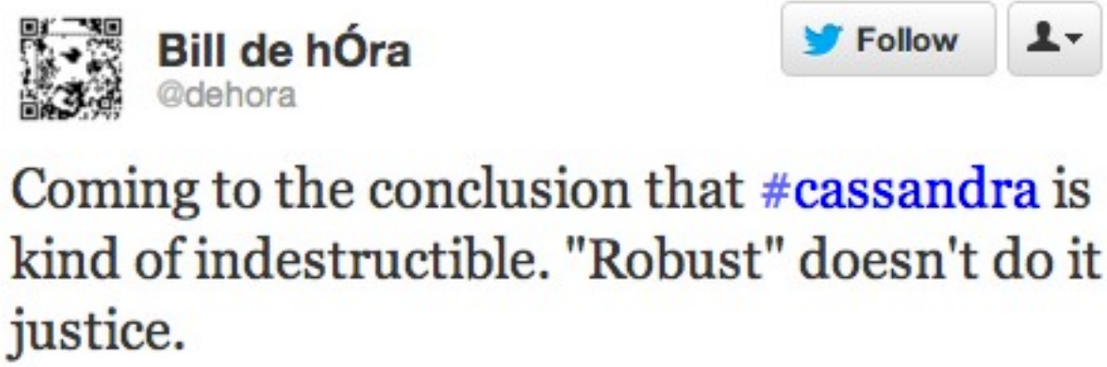


Use case patterns

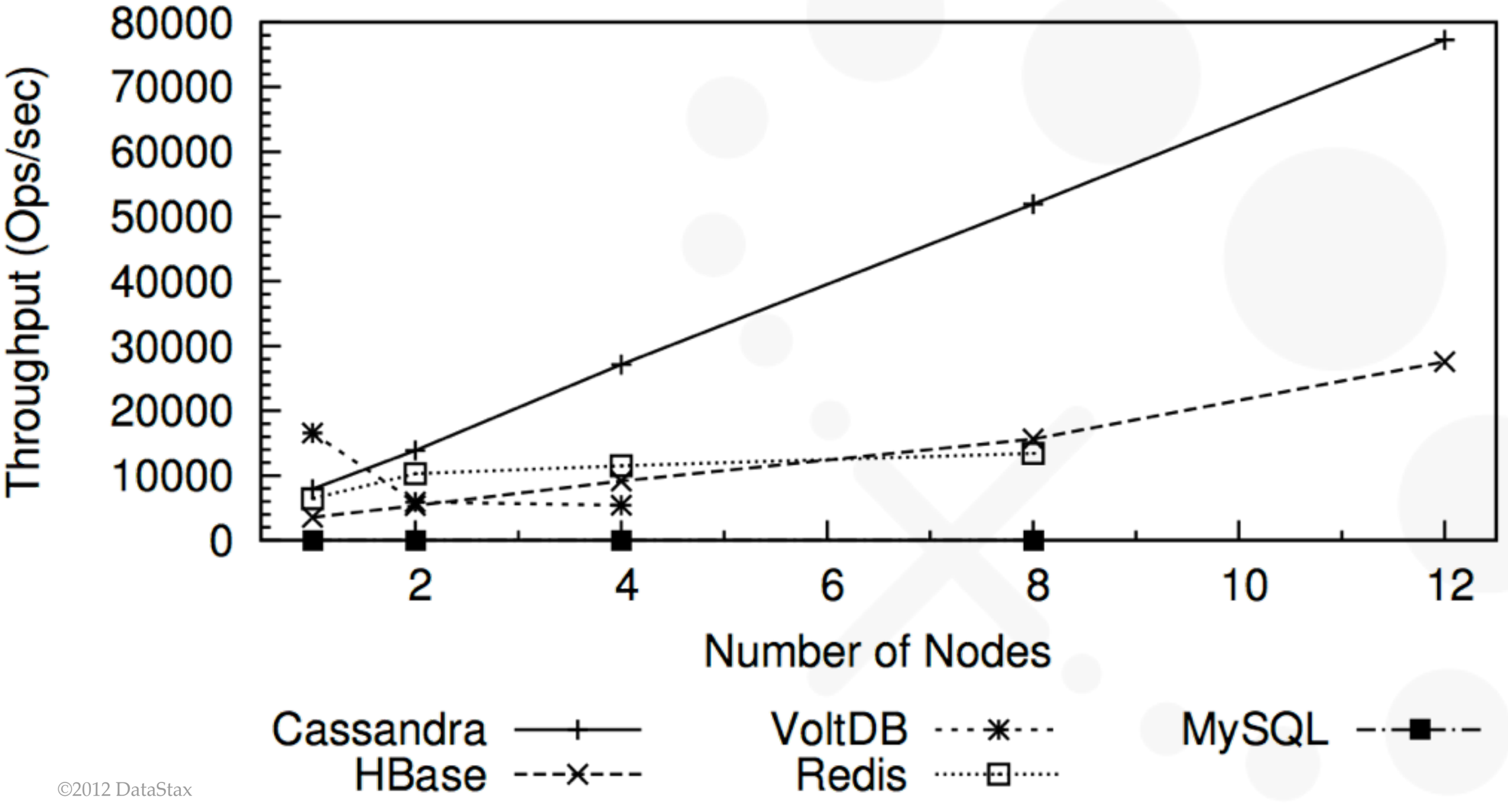
- High performance
- Massively Scalable
- Reliable/Available



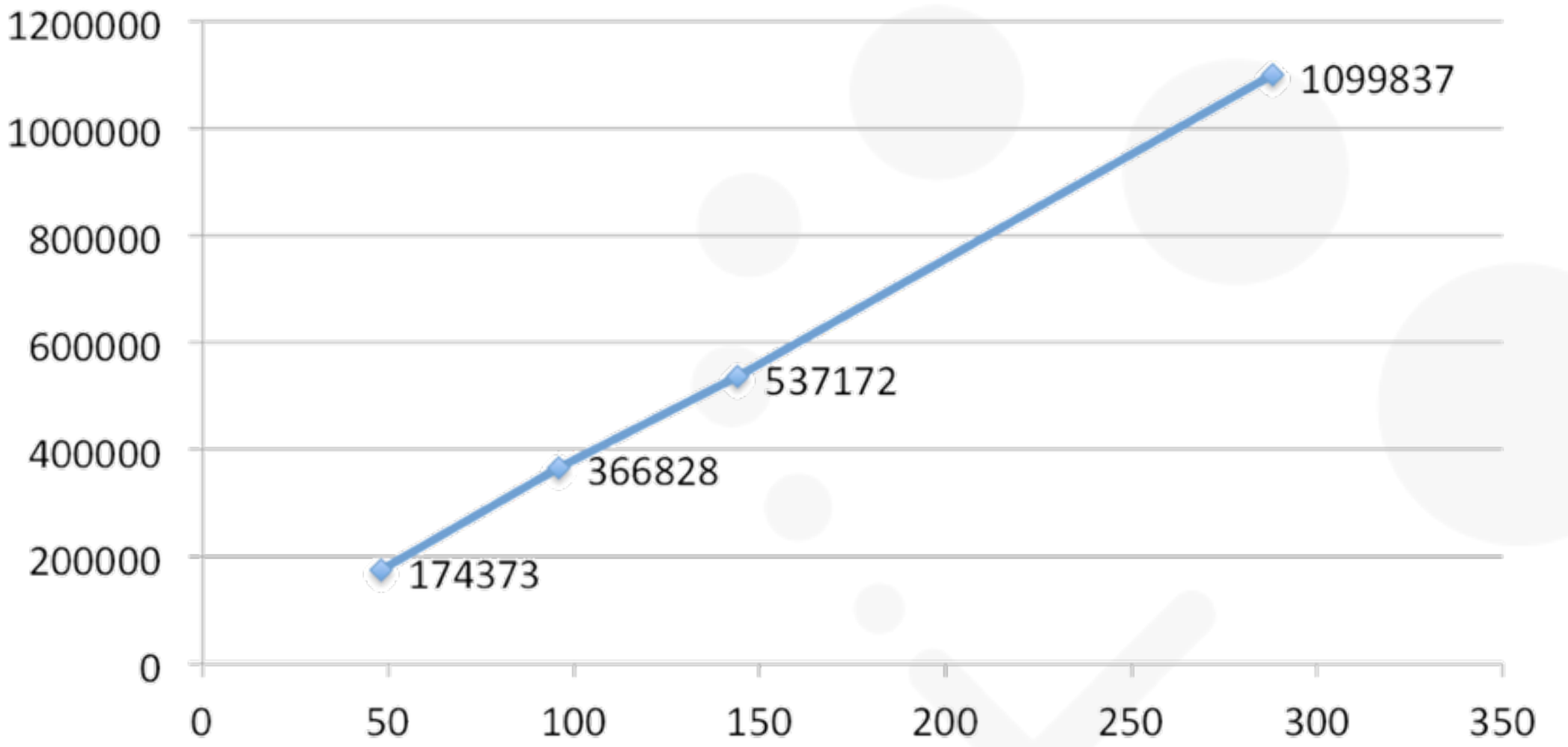




VLDB benchmark



Client Writes/s by node count – Replication Factor = 3



NETFLIX

Bill de hÓra
@dehora

Follow

Coming to the conclusion that #cassandra is kind of indestructible. "Robust" doesn't do it justice.

Aaron Turner
@synfinatic

Follow

took me 10hrs to notice a #cassandra node had a hw failure because everything just kept working. #sweet

Eric Florenzano
@ericflo

Following

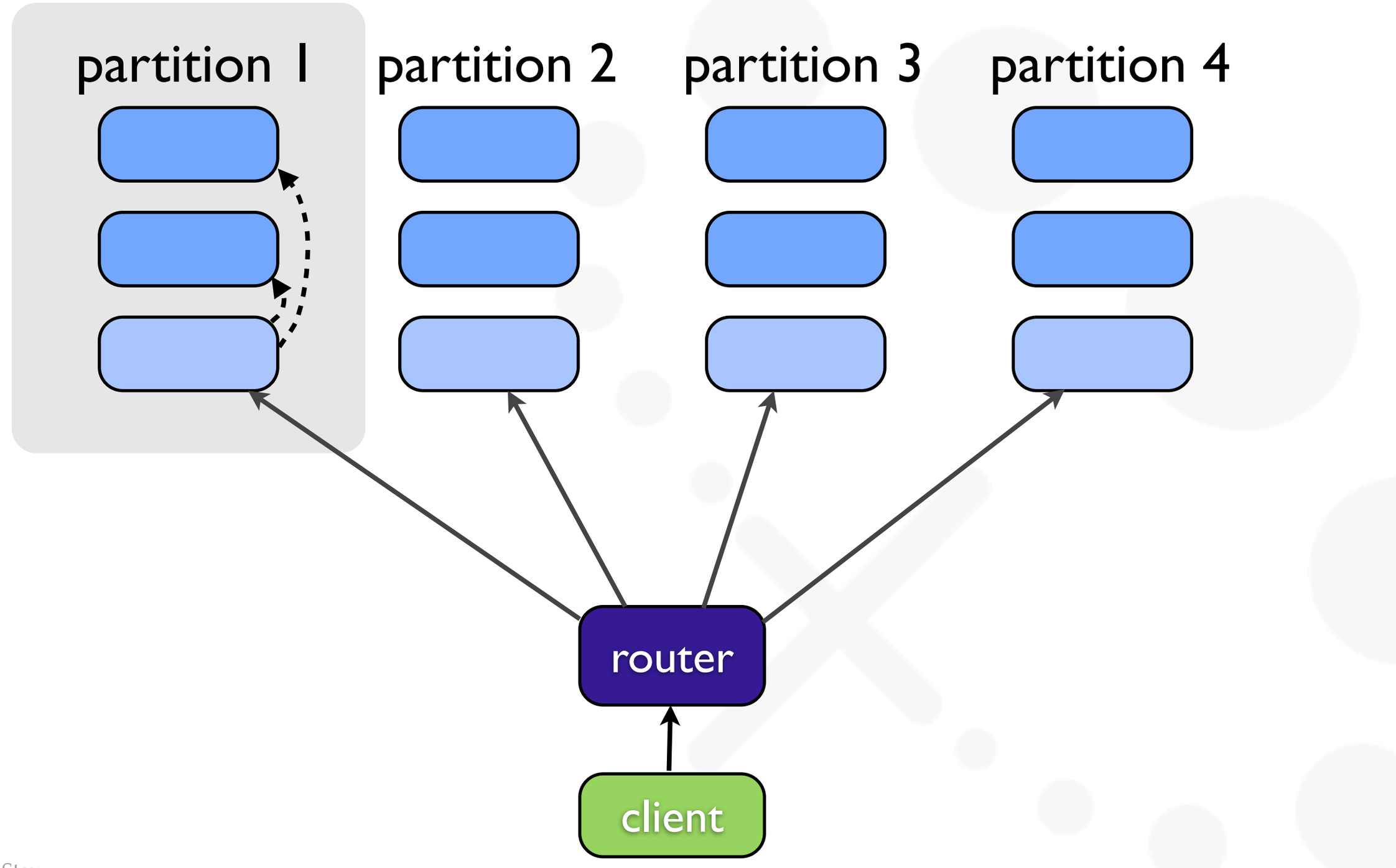
"Cassandra ... dealt with the loss of one third of its regional nodes without any loss of data or availability." techblog.netflix.com/2012/07/lesson... - Nice!

Nathan Milford
@NathanMilford

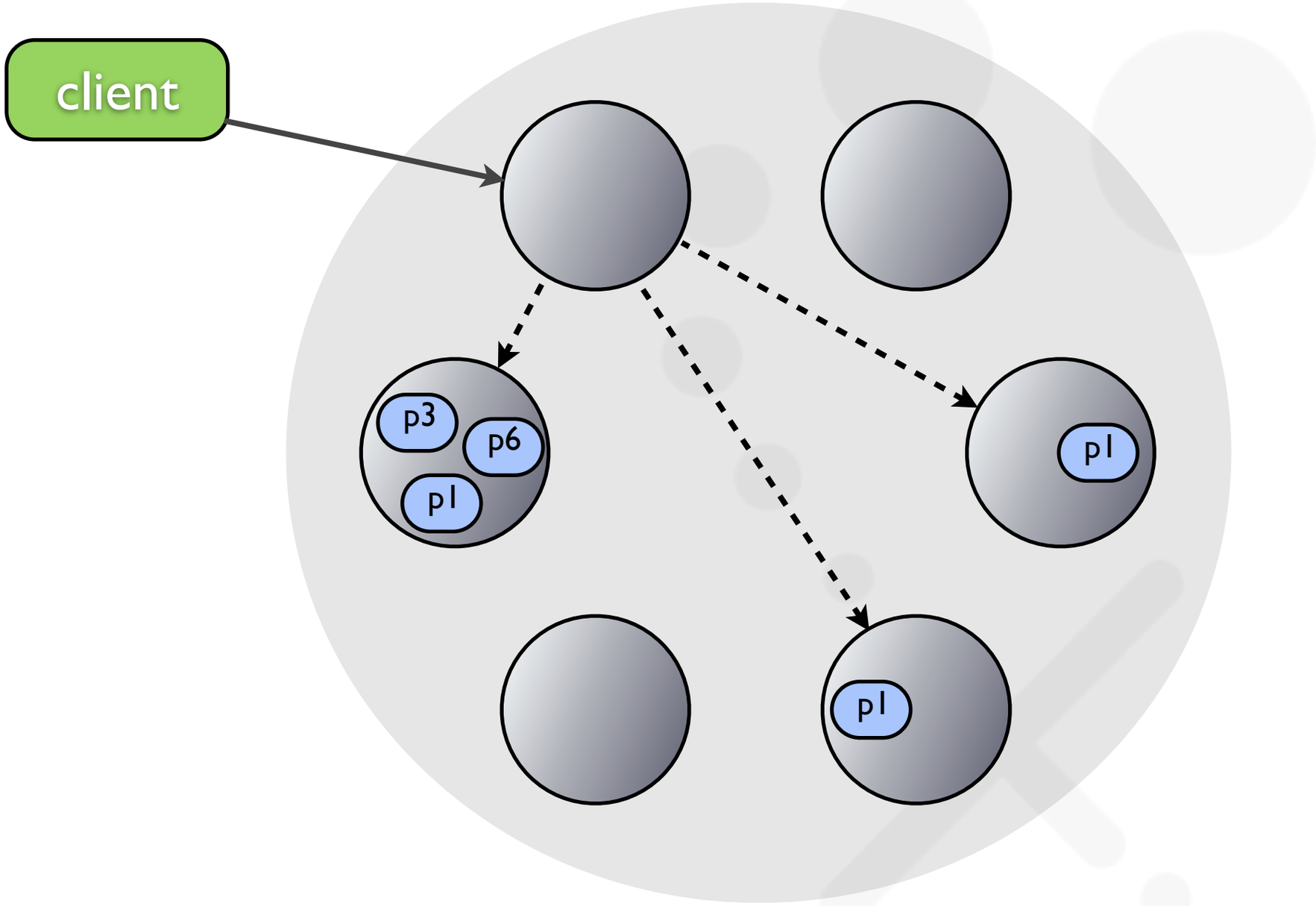
Follow

Man, I just love Cassandra. Lost a data center Hurricane Sandy, nodes came up and started working with no pain.

Classic partitioning with SPOF



Fully distributed, no SPOF



Partitioning

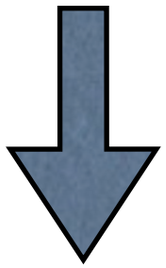
jim	age: 36	car: camaro	gender: M
carol	age: 37	car: subaru	gender: F
johnny	age:12	gender: M	
suzy	age:10	gender: F	

Partitioning

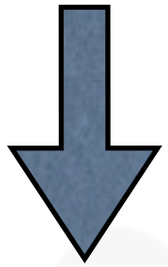
Primary key determines placement*

jim	age: 36	car: camaro	gender: M
carol	age: 37	car: subaru	gender: F
johnny	age:12	gender: M	
suzy	age:10	gender: F	

PK



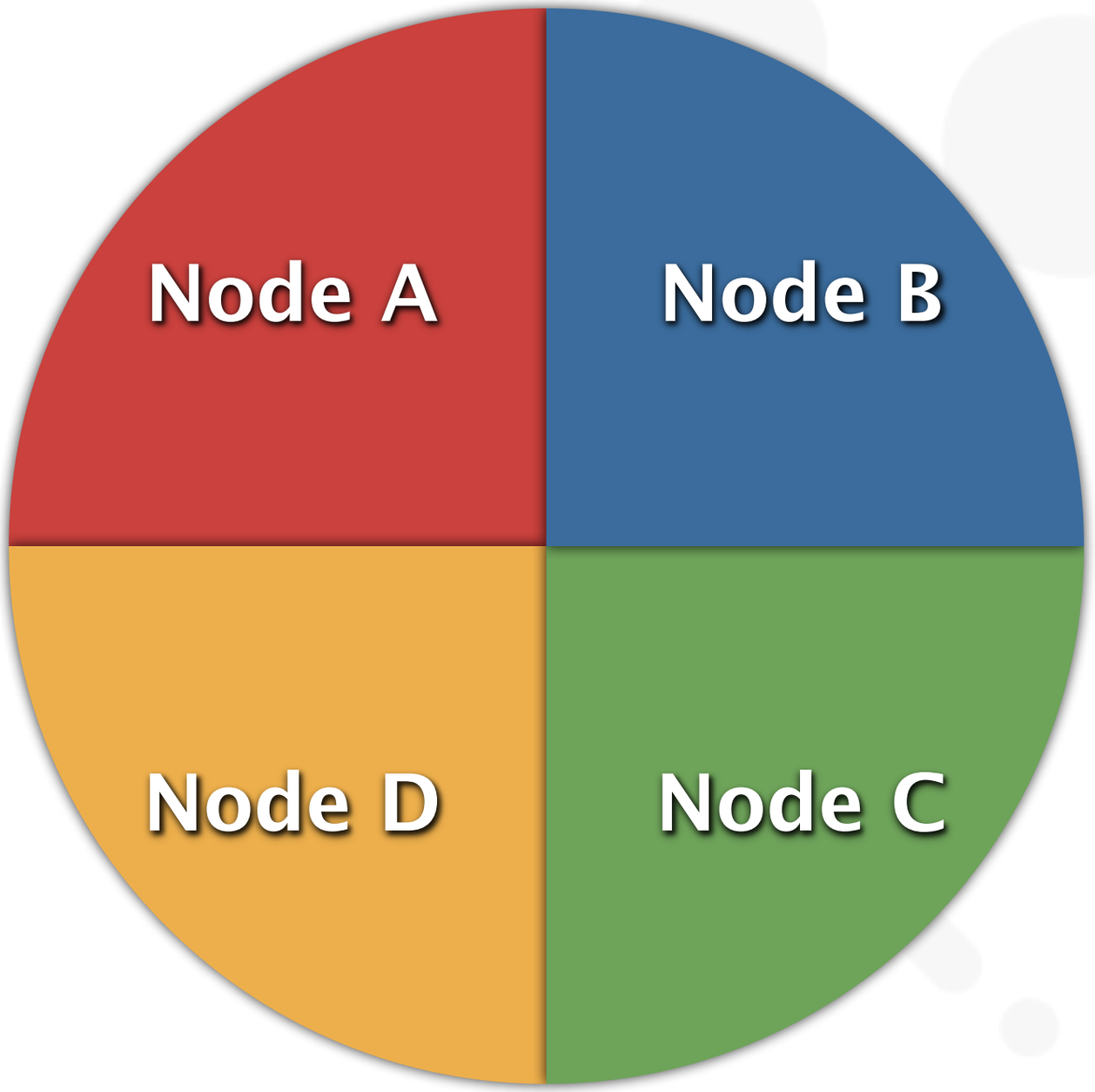
MD5 Hash



PK	MD5 Hash
jim	5e02739678...
carol	a9a0198010...
johnny	f4eb27cea7...
suzy	78b421309e...

MD5* hash operation yields a 128-bit number for keys of any size.

The “token ring”



	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

jim	5e02739678...
carol	a9a0198010...
johnny	f4eb27cea7...
suzy	78b421309e...

	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

jim	5e02739678...
carol	a9a0198010...
johnny	f4eb27cea7...
suzy	78b421309e...

	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

jim	5e02739678...
carol	a9a0198010...
johnny	f4eb27cea7...
suzy	78b421309e...

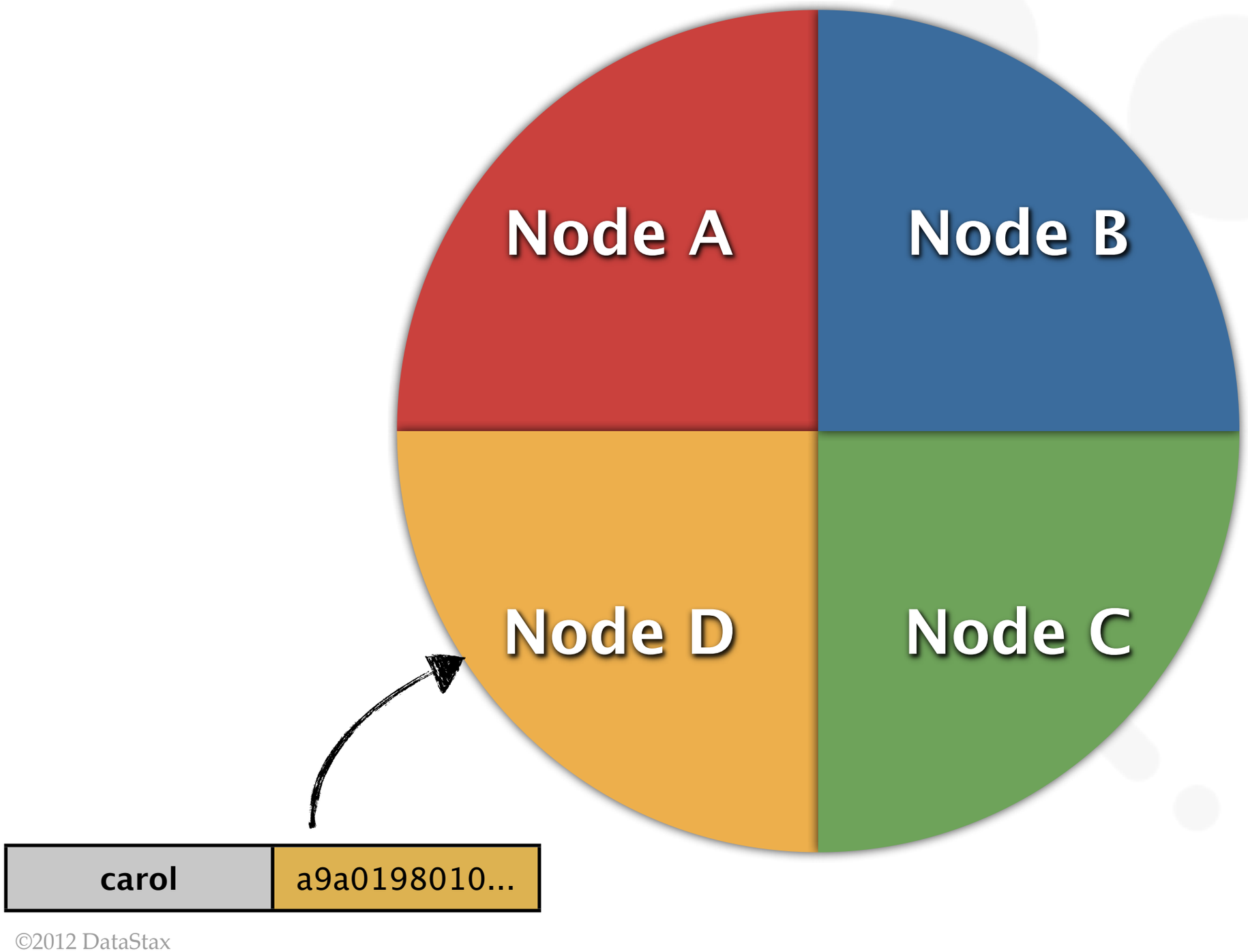
	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

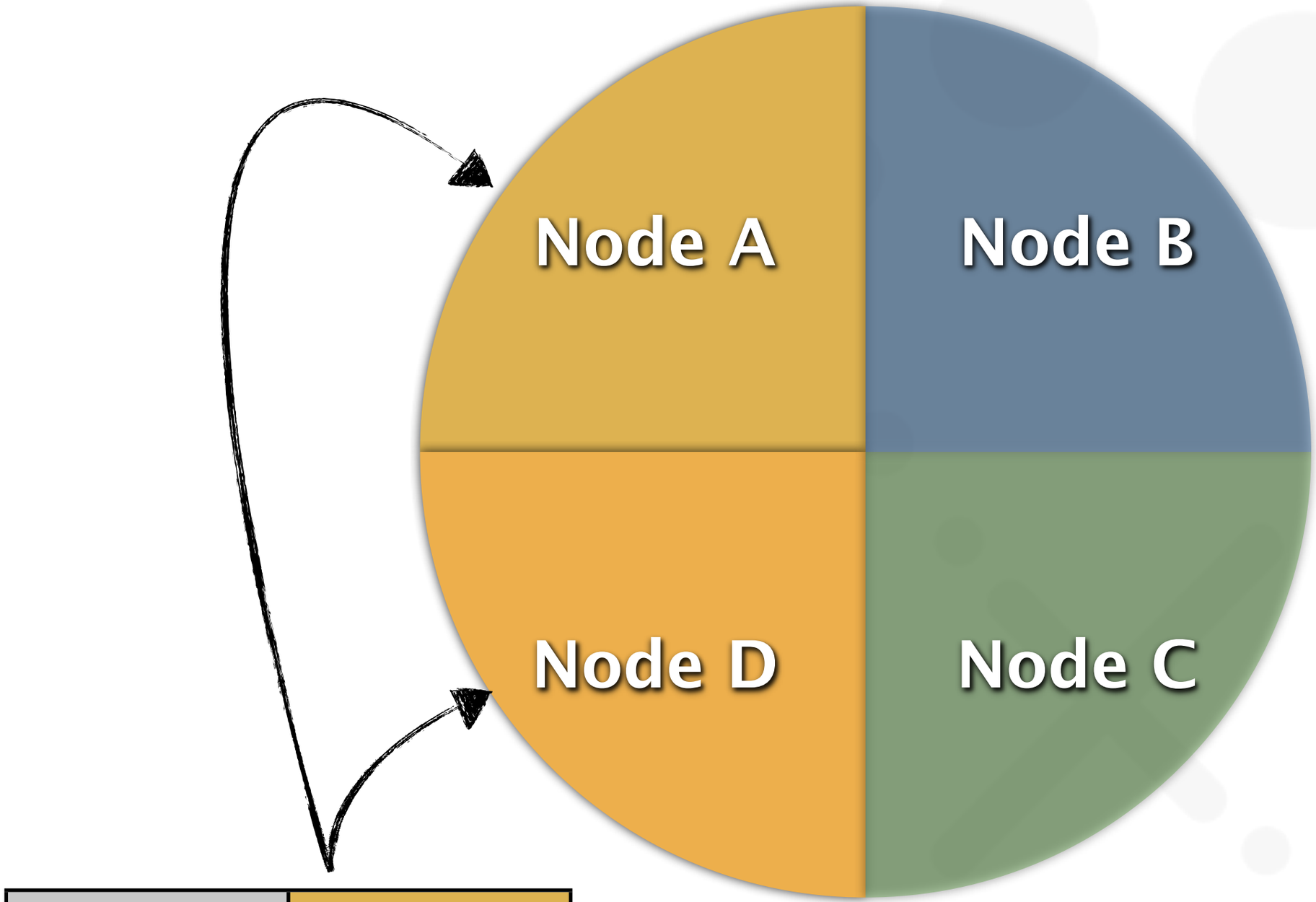
jim	5e02739678...
carol	a9a0198010...
johnny	f4eb27cea7...
suzy	78b421309e...

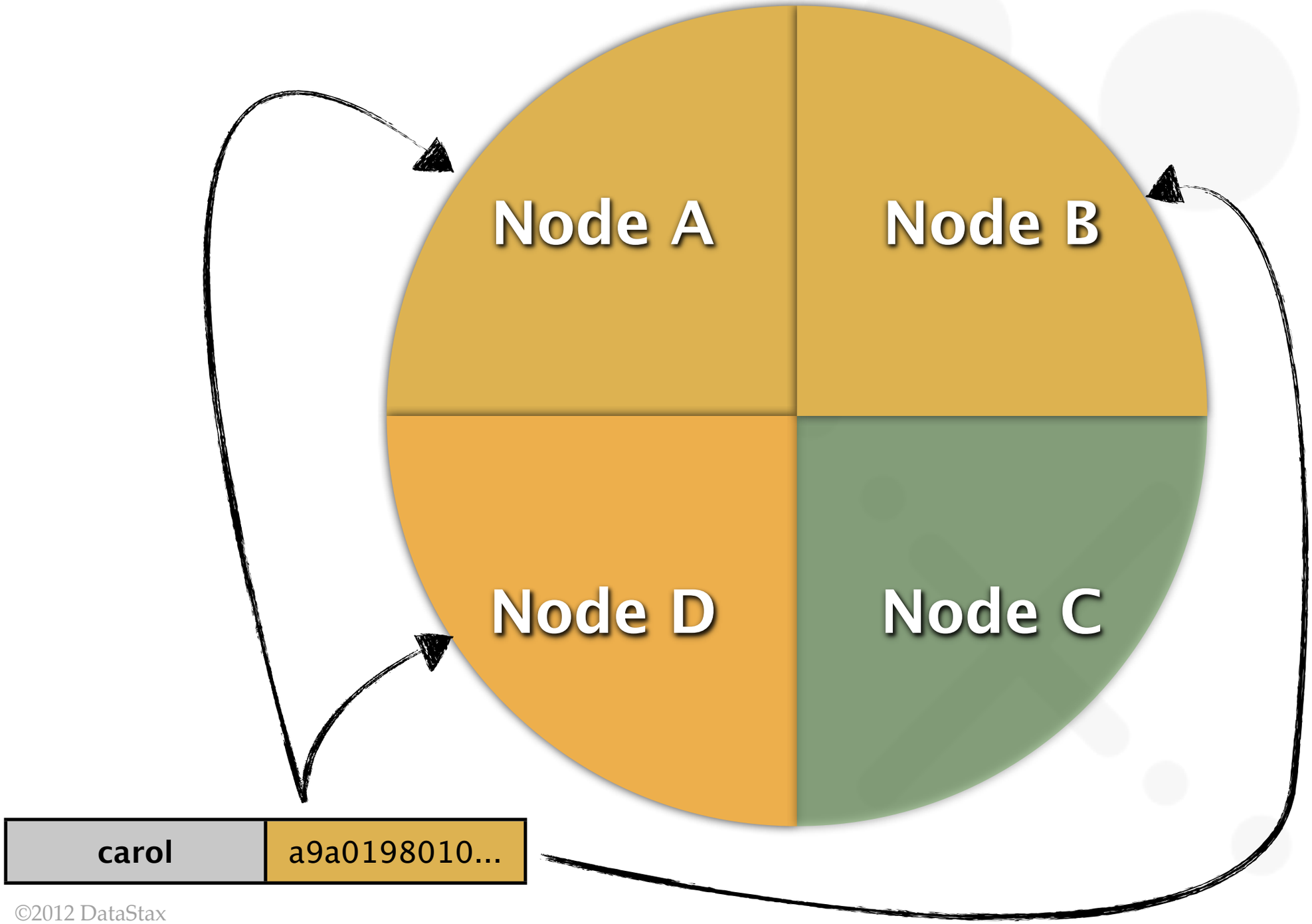
	Start	End
A	0xc000000000..1	0x0000000000..0
B	0x0000000000..1	0x4000000000..0
C	0x4000000000..1	0x8000000000..0
D	0x8000000000..1	0xc000000000..0

jim	5e02739678...
carol	a9a0198010...
johnny	f4eb27cea7...
suzy	78b421309e...

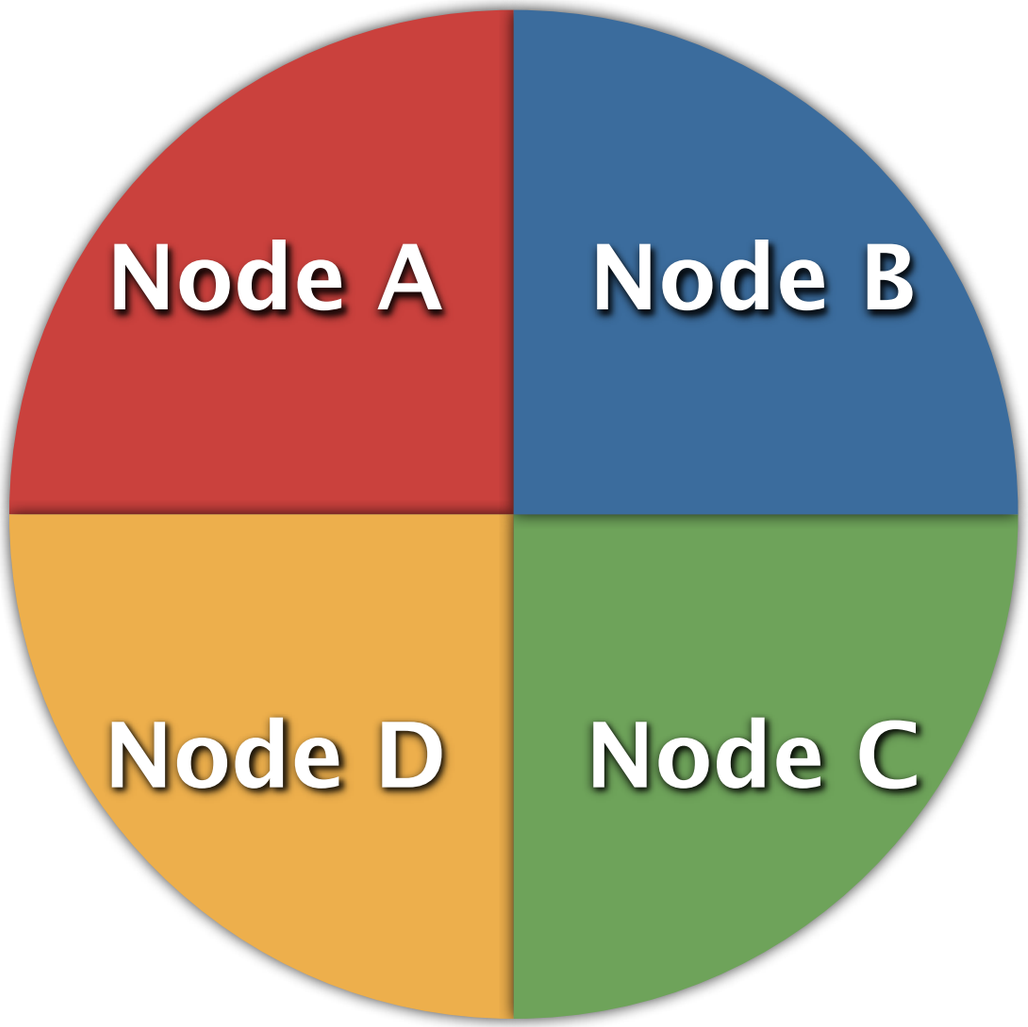
Replication



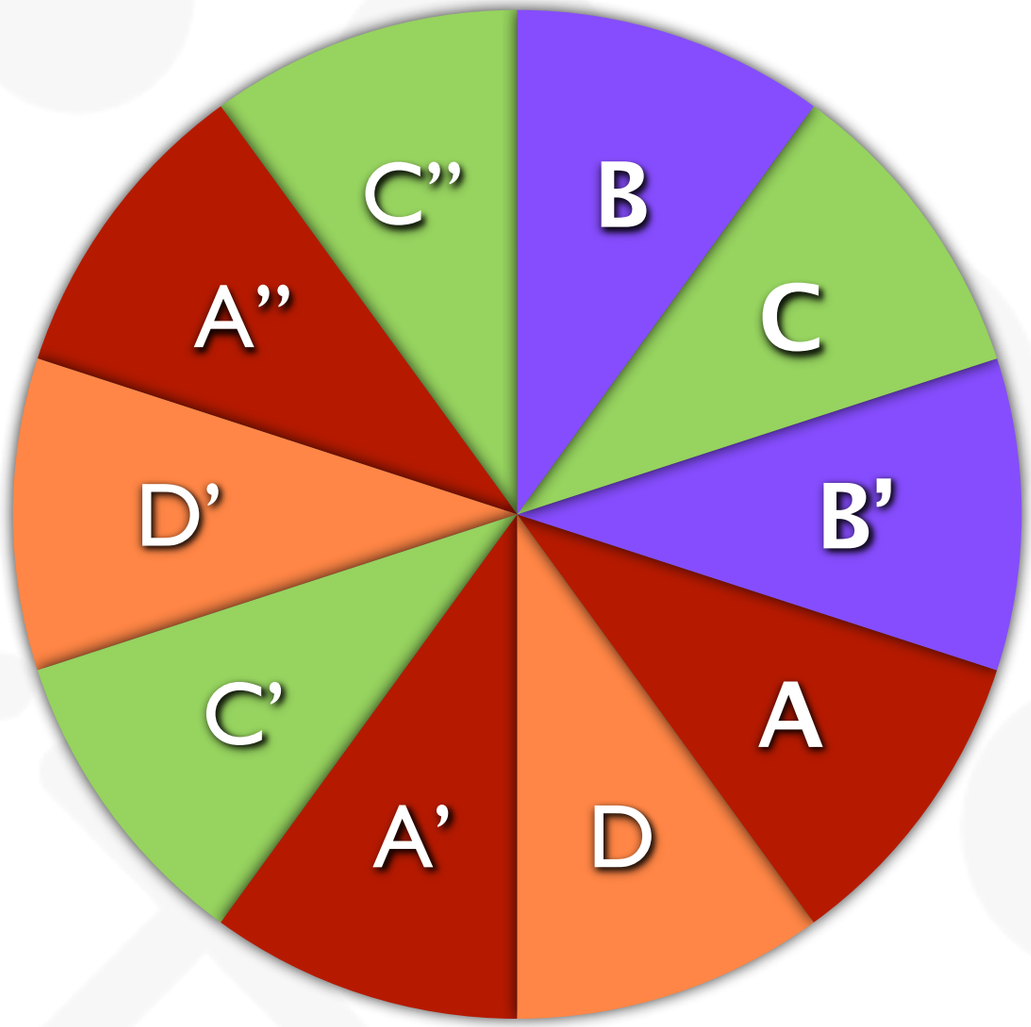




Virtual nodes



Without vnodes



With vnodes

Highlights

- Adding capacity is application-transparent and requires no downtime
- No SPOF, not even temporarily
  - No “primary” replica
- Configurable synchronous/asynchronous
- Tolerates node failure; never have to restart replication “from scratch”
- “Smart” replication avoids correlated failures

CQL: You got SQL in my NoSQL!

```
CREATE TABLE users (
  id uuid PRIMARY KEY,
  name text,
  state text,
  birth_date int
);

CREATE INDEX ON users(state);

SELECT * FROM users WHERE state='Texas' AND birth_date > 1950;
```

Joins?

```
CREATE TABLE users (
  id uuid PRIMARY KEY,
  name text,
  state text,
  birth_date int
);

CREATE TABLE users_addresses (
  user_id uuid REFERENCES users,
  email text
);

SELECT *
FROM users NATURAL JOIN users_addresses;
```

Joins?

```
CREATE TABLE users (
  id uuid PRIMARY KEY,
  name text,
  state text,
  birth_date int
);

CREATE TABLE users_addresses (
  user_id uuid REFERENCES users,
  email text
);

SELECT *
FROM users NATURAL JOIN users_addresses;
```

Collections

```
CREATE TABLE users (
  id uuid PRIMARY KEY,
  name text,
  state text,
  birth_date int,
  email_addresses set<text>
);
```

```
UPDATE users
SET email_addresses = email_addresses + {‘jbellis@gmail.com’,
‘jbellis@datastax.com’};
```

Strictly "realtime" focused

- No joins
- No subqueries
- No aggregation functions* or GROUP BY
- ORDER BY?

Songs

```
CREATE TABLE songs (
  id uuid PRIMARY KEY,
  title text,
  artist text,
  album text,
  tags set<text>,
  data blob
);
```

id	title	artist	album	tags
a3e64f8f...	La Grange	ZZ Top	Tres Hombres	{blues, 1973}
8a172618...	Moving in Stereo	Fu Manchu	We Must Obey	{covers, 2003}
2b09185b...	Outside Woman Blues	Back Door Slam	Roll Away	

Playlists

```
CREATE TABLE playlists (
  id uuid,
  title text,
  album text,
  artist text,
  song_id uuid,
  PRIMARY KEY (id, title, album, artist)
);
```

id	title	artist	album	song_id
62c36092...	La Grange	ZZ Top	Tres Hombres	a3e64f8f...
62c36092...	Moving in Stereo	Fu Manchu	We Must Obey	8a172618...
62c36092...	Outside Wo...	Back Door Slam	Roll Away	2b09185b...

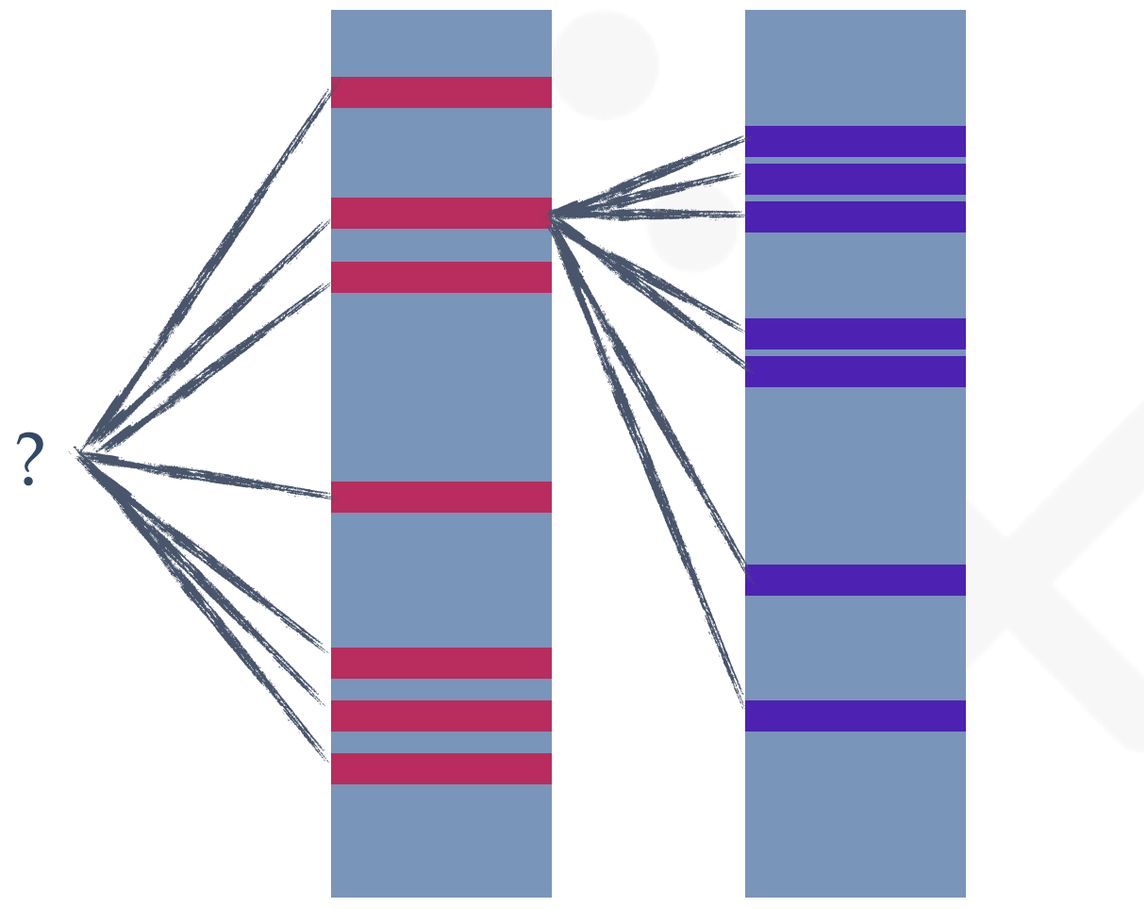
Partition keys

```
CREATE TABLE playlists (
  id uuid,
  title text,
  album text,
  artist text,
  song_id uuid,
  PRIMARY KEY (id, title, album, artist)
);
```

id	title	artist	album	song_id
62c36092...	La Grange	ZZ Top	Tres Hombres	a3e64f8f...
62c36092...	Moving in Stereo	Fu Manchu	We Must Obey	8a172618...
62c36092...	Outside Wo...	Back Door Slam	Roll Away	2b09185b...

Tweets in SQL

```
SELECT * FROM tweets
WHERE user_id IN (SELECT follower FROM followers
                  WHERE user_id = ?)
ORDER BY tweet_id desc
```



Clustering

```
CREATE TABLE timeline (
  user_id uuid,
  tweet_id timeuuid,
  tweet_author uuid,
  tweet_body text,
  PRIMARY KEY (user_id,
               tweet_id)
);

SELECT * FROM timeline
WHERE user_id = 'driftx';
```

user_id	tweet_id	_author	_body
jbellis	3290f9da..	rbranson	lorem
jbellis	389541 1a..	tjake	ipsum
...	...	...	
driftx	3290f9da..	rbranson	lorem
driftx	71b46a84..	yzhang	dolor
...	...	...	
yukim	3290f9da..	rbranson	lorem
yukim	e451dd42..	tjake	amet
...	...	...	

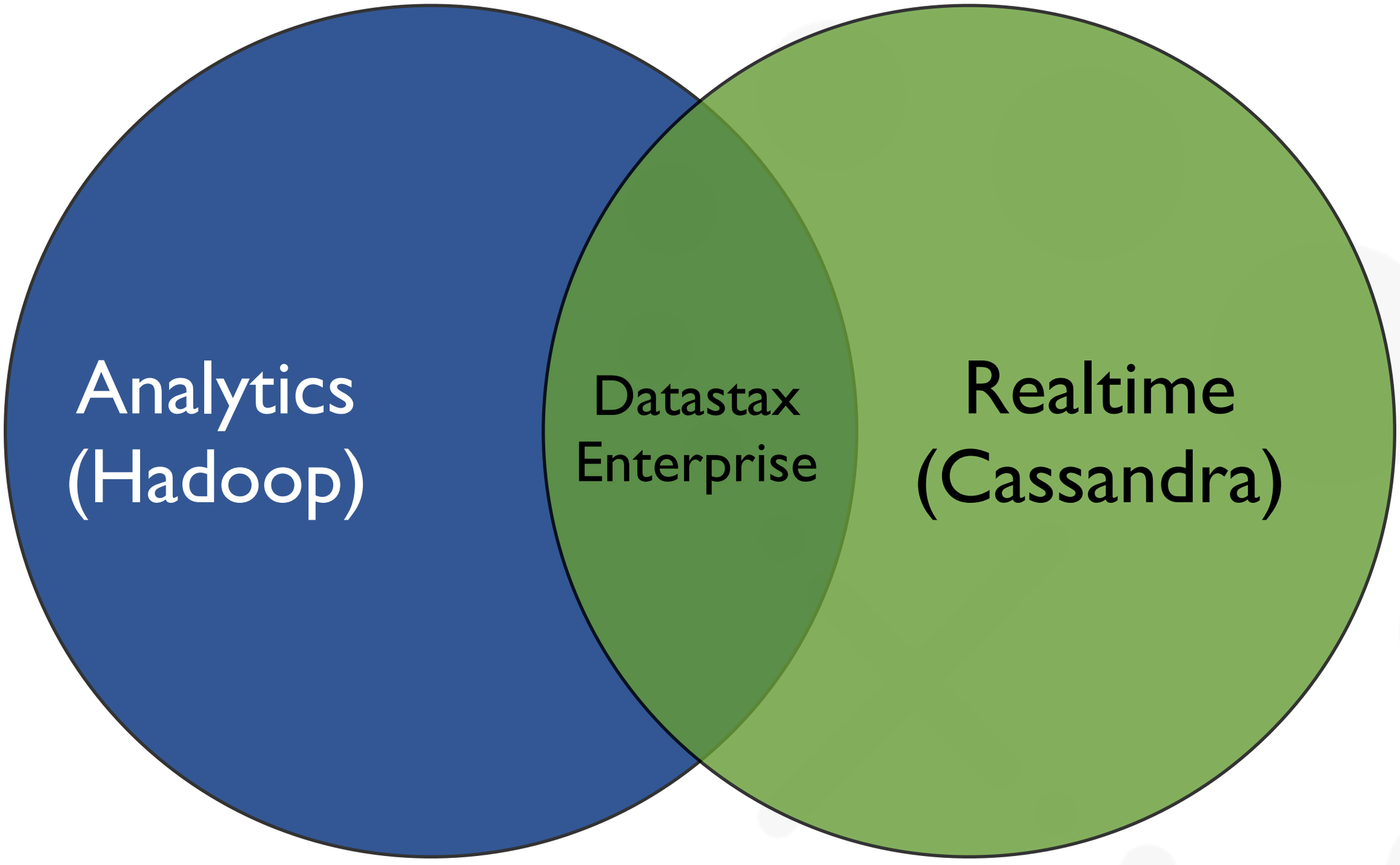
Clustering

```
CREATE TABLE timeline (
  user_id uuid,
  tweet_id timeuuid,
  tweet_author uuid,
  tweet_body text,
  PRIMARY KEY (user_id,
               tweet_id)
);

SELECT * FROM timeline
WHERE user_id = 'driftx';
```

user_id	tweet_id	_author	_body
jbellis	3290f9da..	rbranson	lorem
jbellis	389541 1a..	tjake	ipsum
...	...	...	
driftx	3290f9da..	rbranson	lorem
driftx	71b46a84..	yzhang	dolor
...	...	...	
yukim	3290f9da..	rbranson	lorem
yukim	e451dd42..	tjake	amet
...	...	...	

DataStax Enterprise



DataStax Enterprise

Portfolio #5329 **$13778.89 (+55%)**

Largest Historical 10day Loss:

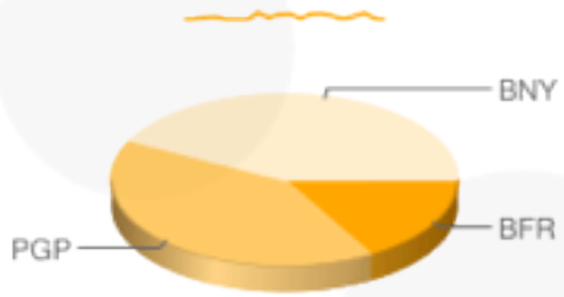
?

Portfolio #1970 **$4325.45 (-59%)**

Largest Historical 10day Loss:

?

Portfolio #9700 **$3526.94 (-85%)**



Largest Historical 10day Loss:

?

Portfolio #4612 **$796.03 (+98%)**

Largest Historical 10day Loss:

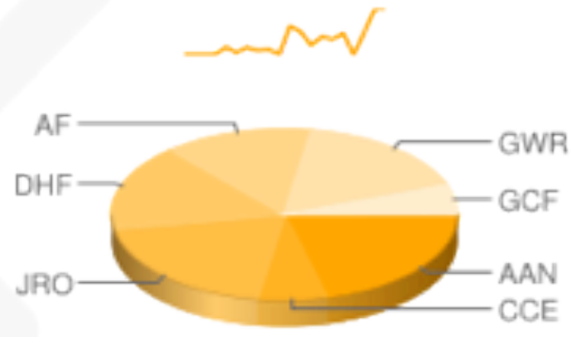
?

Portfolio #6095 **$721.58 (-33%)**

Largest Historical 10day Loss:

?

Portfolio #1126 **$9193.65 (-82%)**



Largest Historical 10day Loss:

?

Data model: Realtime

LiveStocks

stock	last
GOOG	$95.52
AAPL	$186.10
AMZN	$112.98

Portfolios

user	stock	shares
jbellis	GOOG	80
jbellis	LNKD	20
yukim	AMZN	100

StockHist

stock	date	price
GOOG	2011-01-01	$8.23
GOOG	2011-01-02	$6.14
GOOG	2011-001-03	$7.78

Data model: Analytics

HistLoss

	worst_date	loss
Portfolio1	2011-07-23	-$34.81
Portfolio2	2011-03-11	-$11432.24
Portfolio3	2011-05-21	-$1476.93

Data model: Analytics

10dayreturns

stock	rdate	return
GOOG	2011-07-25	$8.23
GOOG	2011-07-24	$6.14
GOOG	2011-07-23	$7.78
AAPL	2011-07-25	$15.32
AAPL	2011-07-24	$12.68

```
INSERT OVERWRITE TABLE 10dayreturns
SELECT a.stock,
       b.date as rdate,
       b.price - a.price
FROM StockHist a
JOIN StockHist b
ON (a.stock = b.stock
    AND date_add(a.date, 10) = b.date);
```

Data model: Analytics

portfolio_returns

portfolio	rdate	preturn
Portfolio1	2011-07-25	$118.21
Portfolio1	2011-07-24	$60.78
Portfolio1	2011-07-23	-$34.81
Portfolio2	2011-07-25	$2143.92
Portfolio3	2011-07-24	-$10.19

```
INSERT OVERWRITE TABLE portfolio_returns
SELECT portfolio,
       rdate,
       SUM(b.return)
FROM portfolios a JOIN 10dayreturns b
ON (a.stock = b.stock)
GROUP BY portfolio, rdate;
```

Data model: Analytics

HistLoss

	worst_date	loss
Portfolio1	2011-07-23	-$34.81
Portfolio2	2011-03-11	-$11432.24
Portfolio3	2011-05-21	-$1476.93

```
INSERT OVERWRITE TABLE HistLoss
SELECT a.portfolio, rdate, minp
FROM (
  SELECT portfolio, min(preturn) as minp
  FROM portfolio_returns
  GROUP BY portfolio
) a
JOIN portfolio_returns b
ON (a.portfolio = b.portfolio and a.minp = b.preturn);
```

Documents in Solr Core : Users

Field	Value
user	jake
state	CT
status	at work

Field	Value
user	jason
state	NY
status	at home

Field	Value
user	jonathan
state	TX
status	in bed

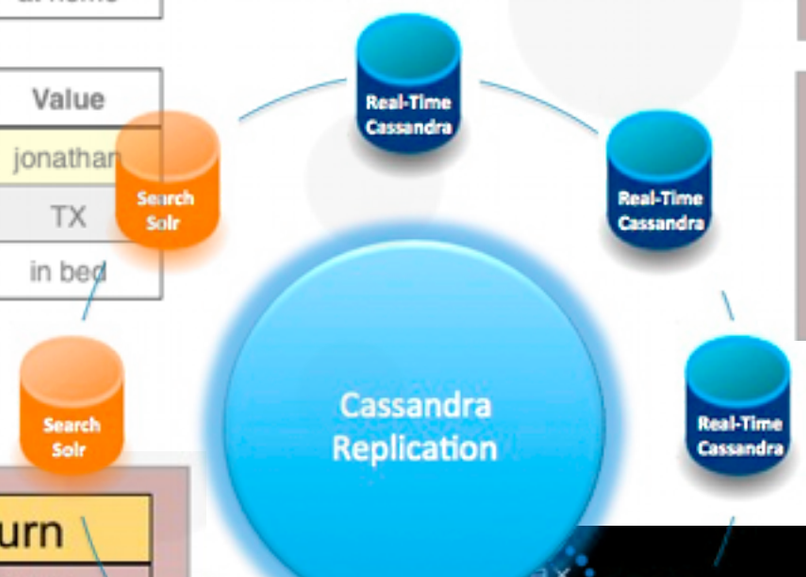
Cassandra ColumnFamily : Users

row_key	state	status
jake	CT	at work
jason	NY	at home
jonathan	TX	in bed

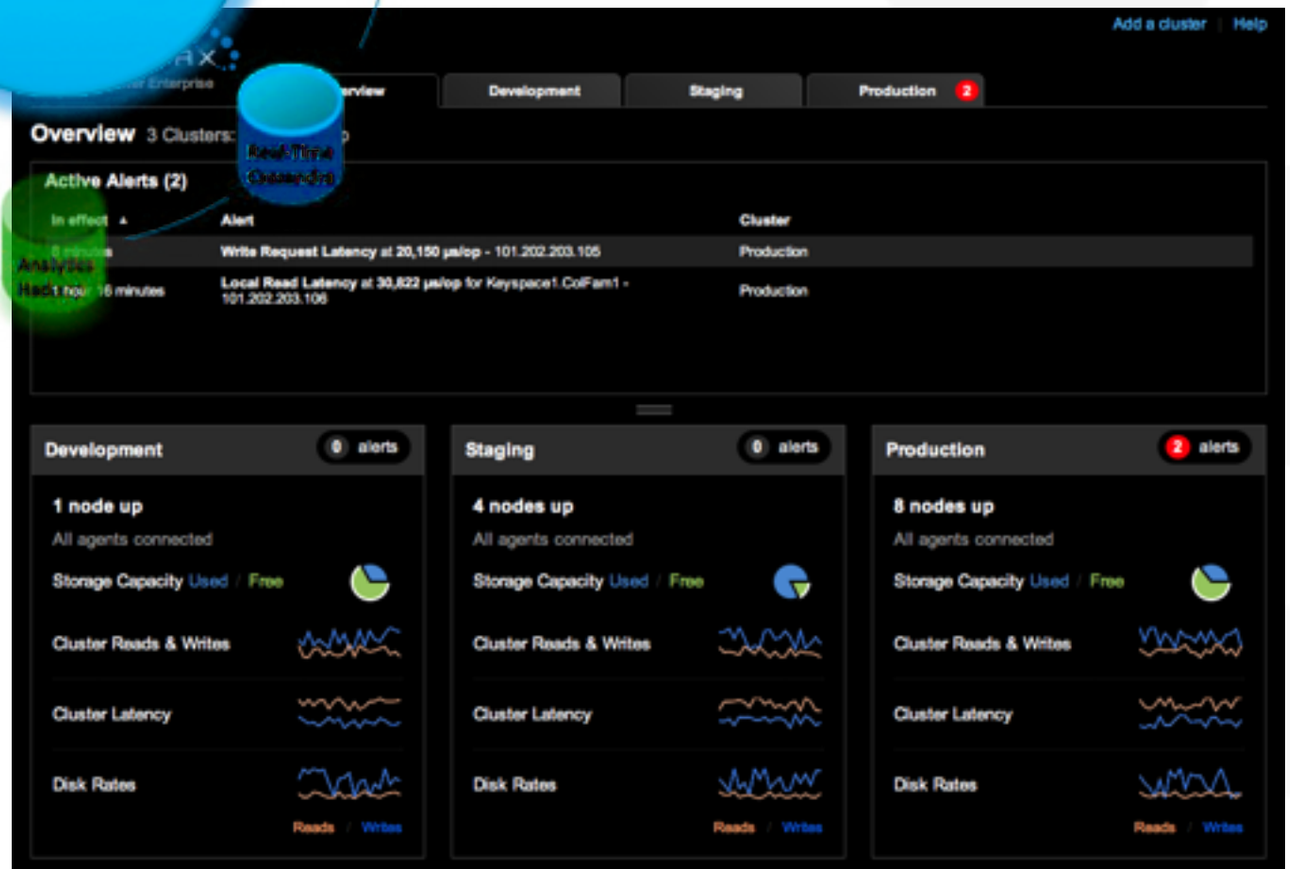
stock	last
GOOG	$95.52
AAPL	$186.10
AMZN	$112.98

user	stock	shares
jbellis	GOOG	80
jbellis	LNKD	20
yukim	AMZN	100

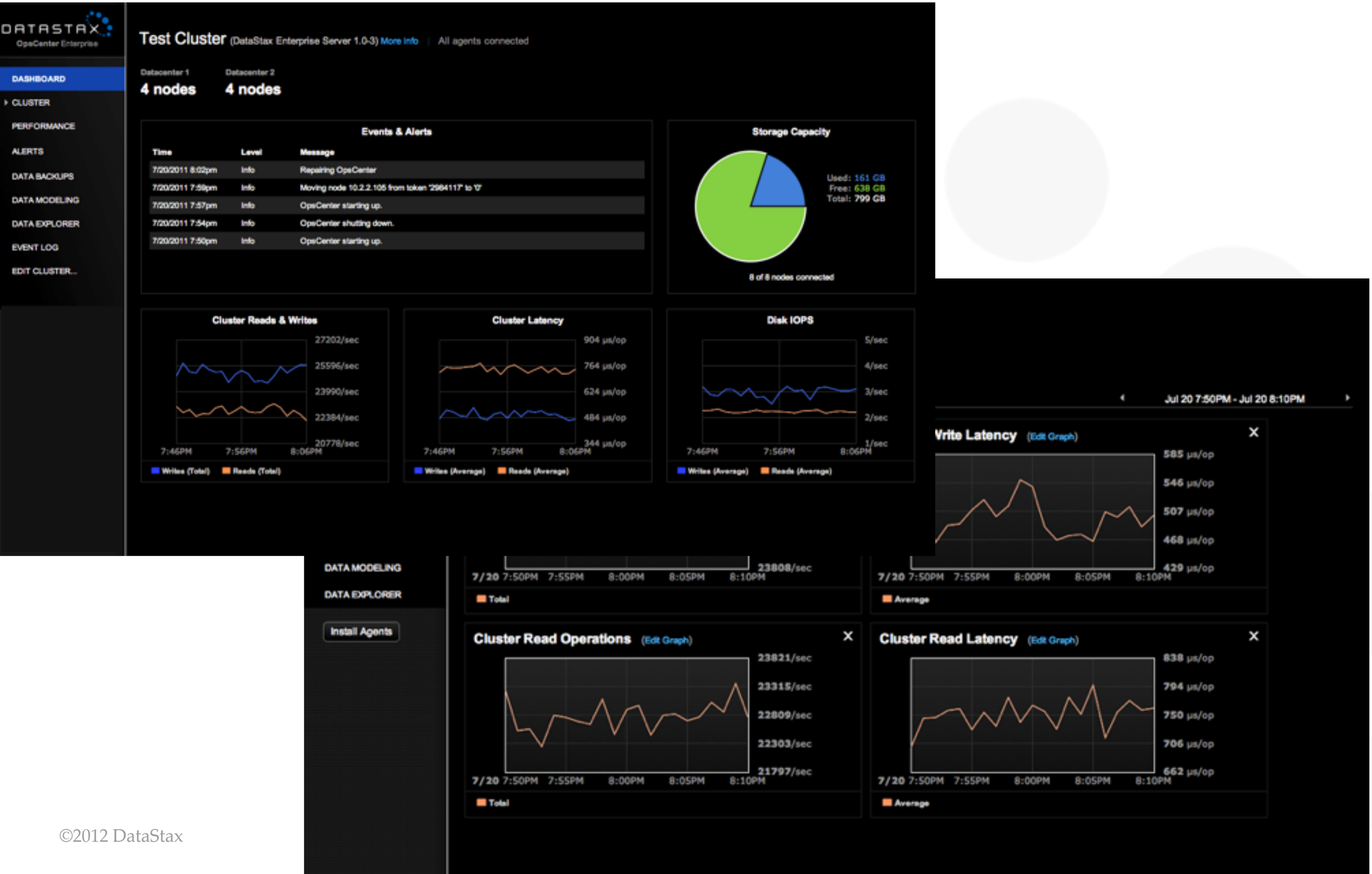
stock	date	price
GOOG	2011-01-01	$8.23
GOOG	2011-01-02	$6.14
GOOG	2011-001-03	$7.78



portfolio	rdate	preturn
Portfolio1	2011-07-25	$118.21
Portfolio1	2011-07-24	$60.78
Portfolio1	2011-07-23	-$34.81
Portfolio2	2011-07-25	$2143.92
Portfolio3	2011-07-24	-$10.19



DataStax Community



Questions?