Building fault tolerant applications with Apache Cassandra

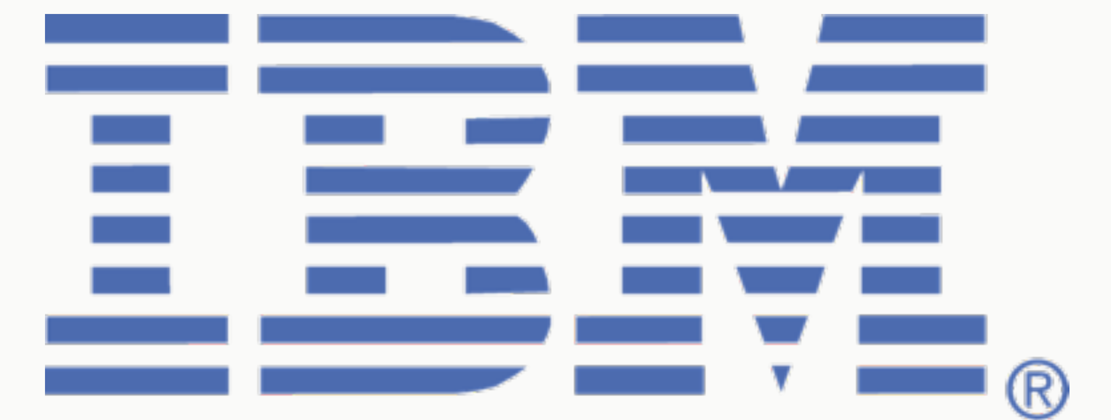
Christopher Batey

@chbatey

geecon

Who am I?

- Built a a lot of systems with Apache Cassandra at Sky
- Work on a testing library for Cassandra
- Help out Cassandra users
- Twitter: @chbatey

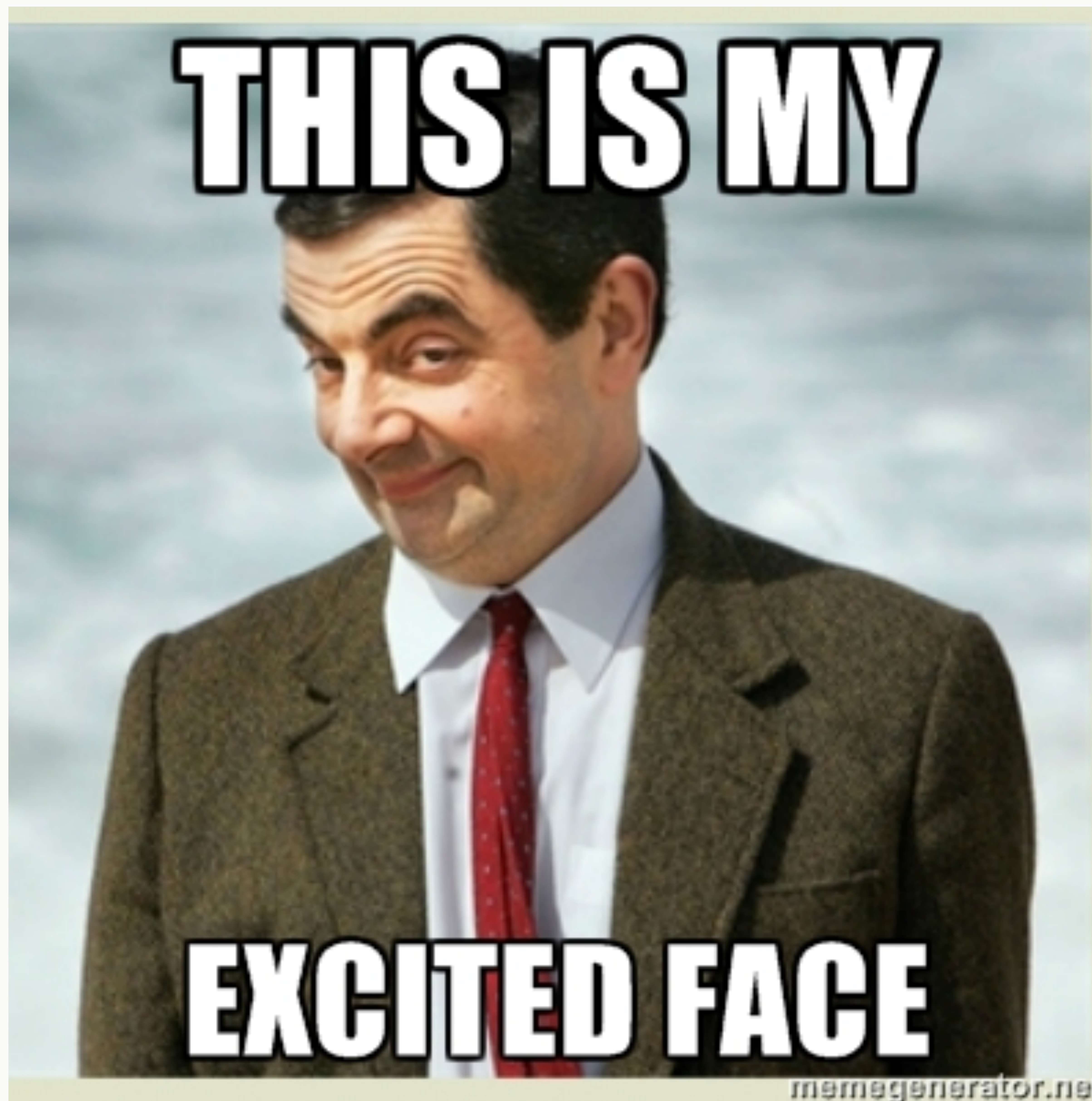


Overview

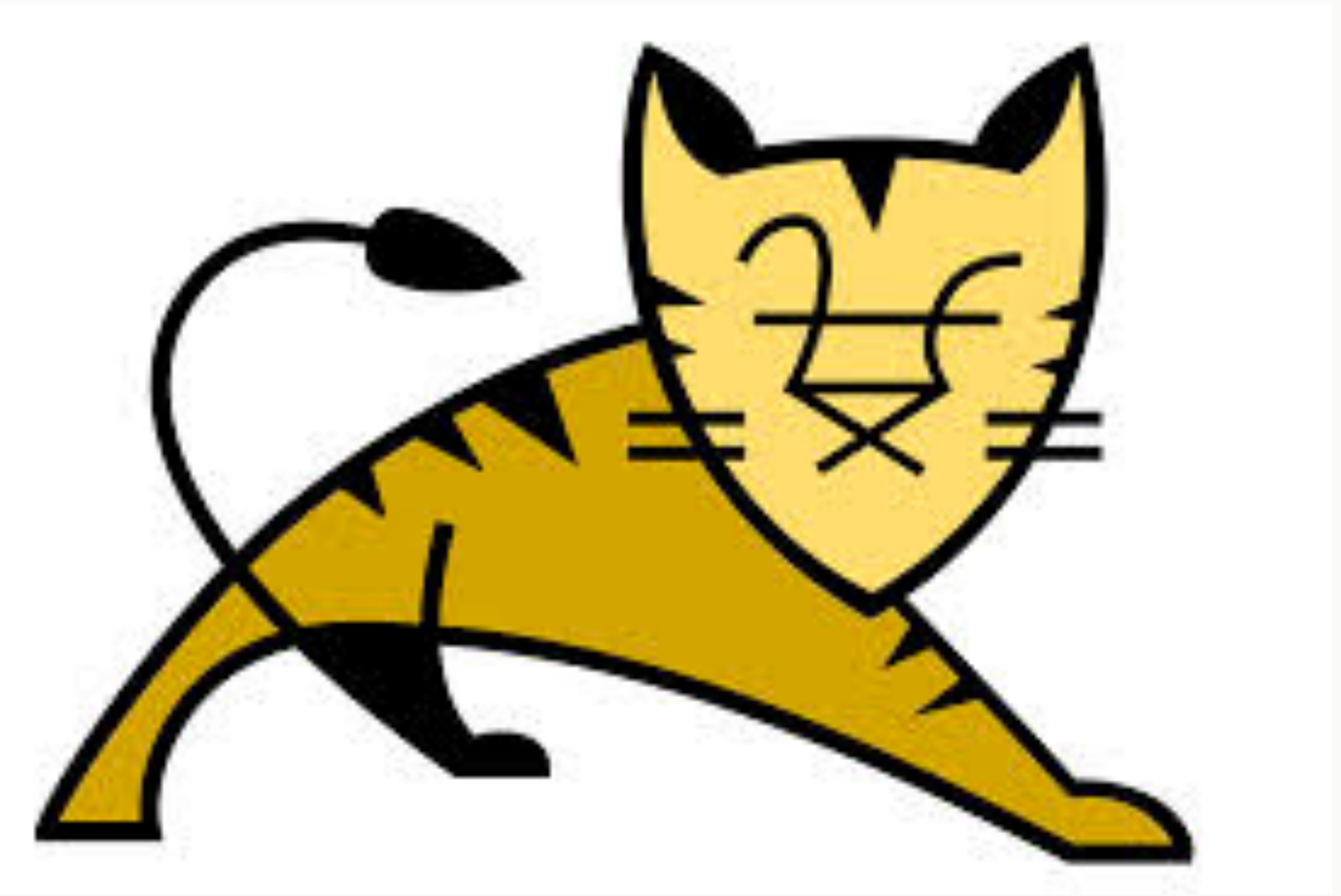
DATASTAX

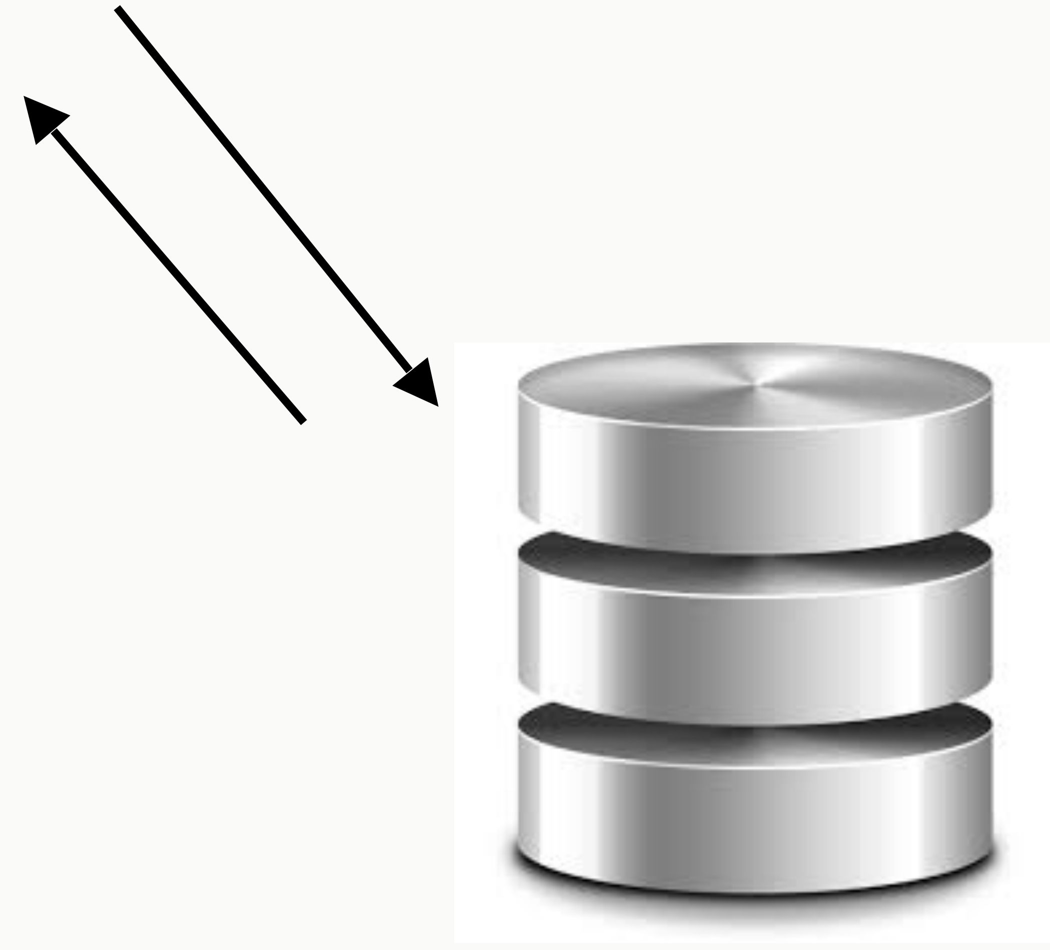
- My path to fault tolerant software
- What do we mean by a fault
- Cassandra deep dive
- Questions

DATASTAX



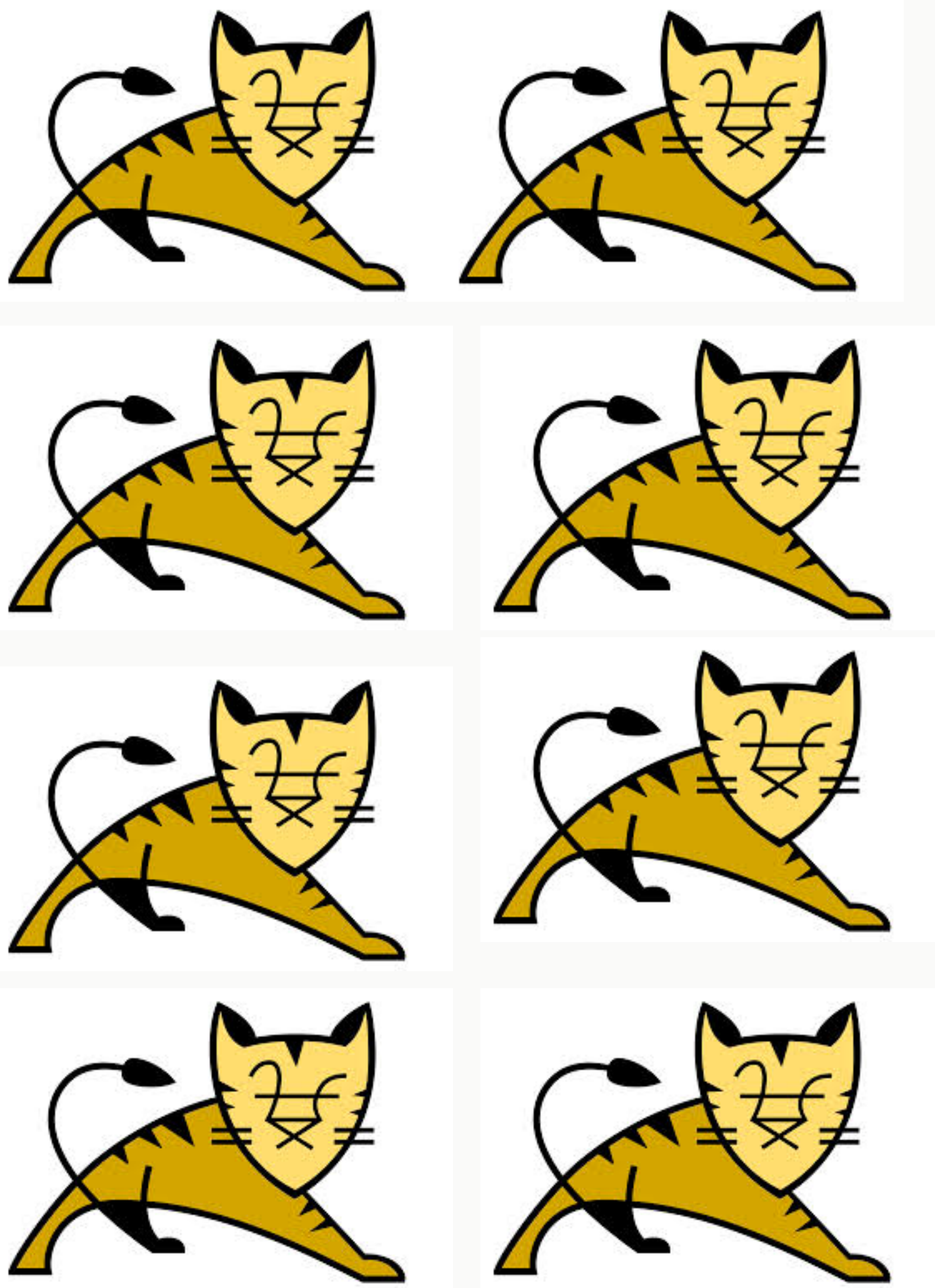
The old





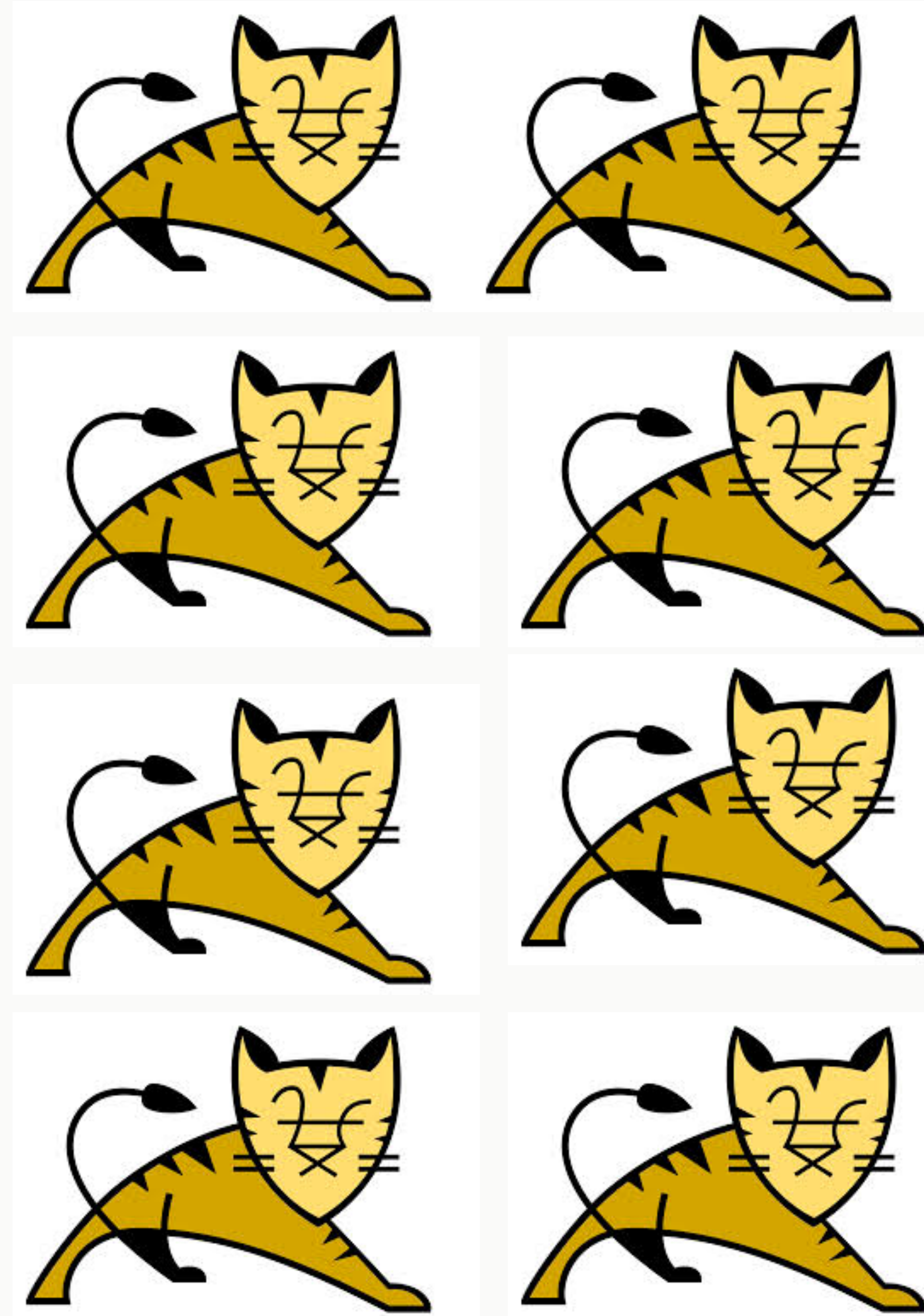
Scaling the old

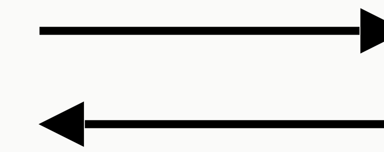
DATASTAX



@chbatey

Uh oh :(

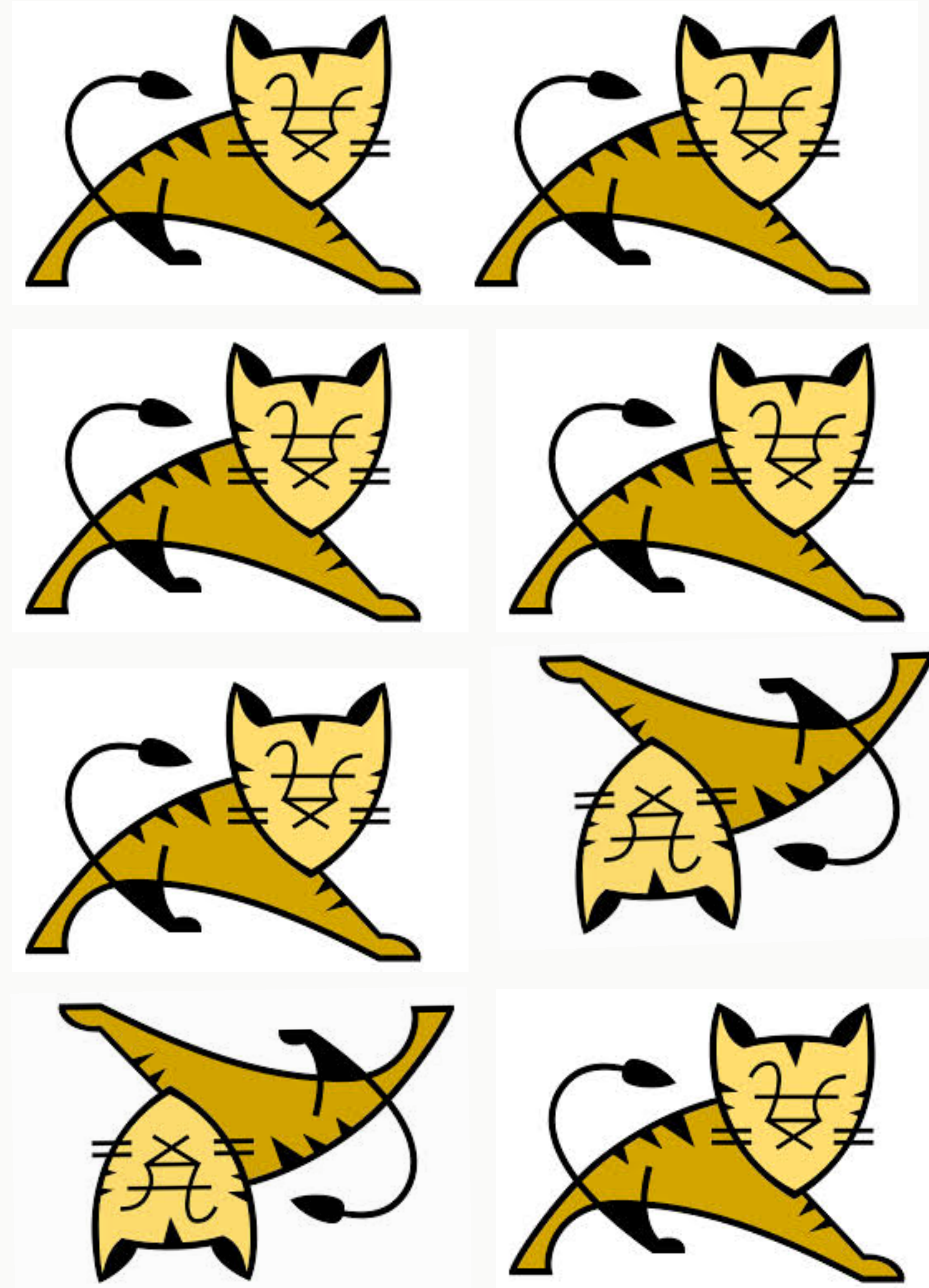


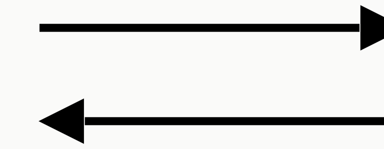


@chbatey

Uh oh :(

DATASTAX



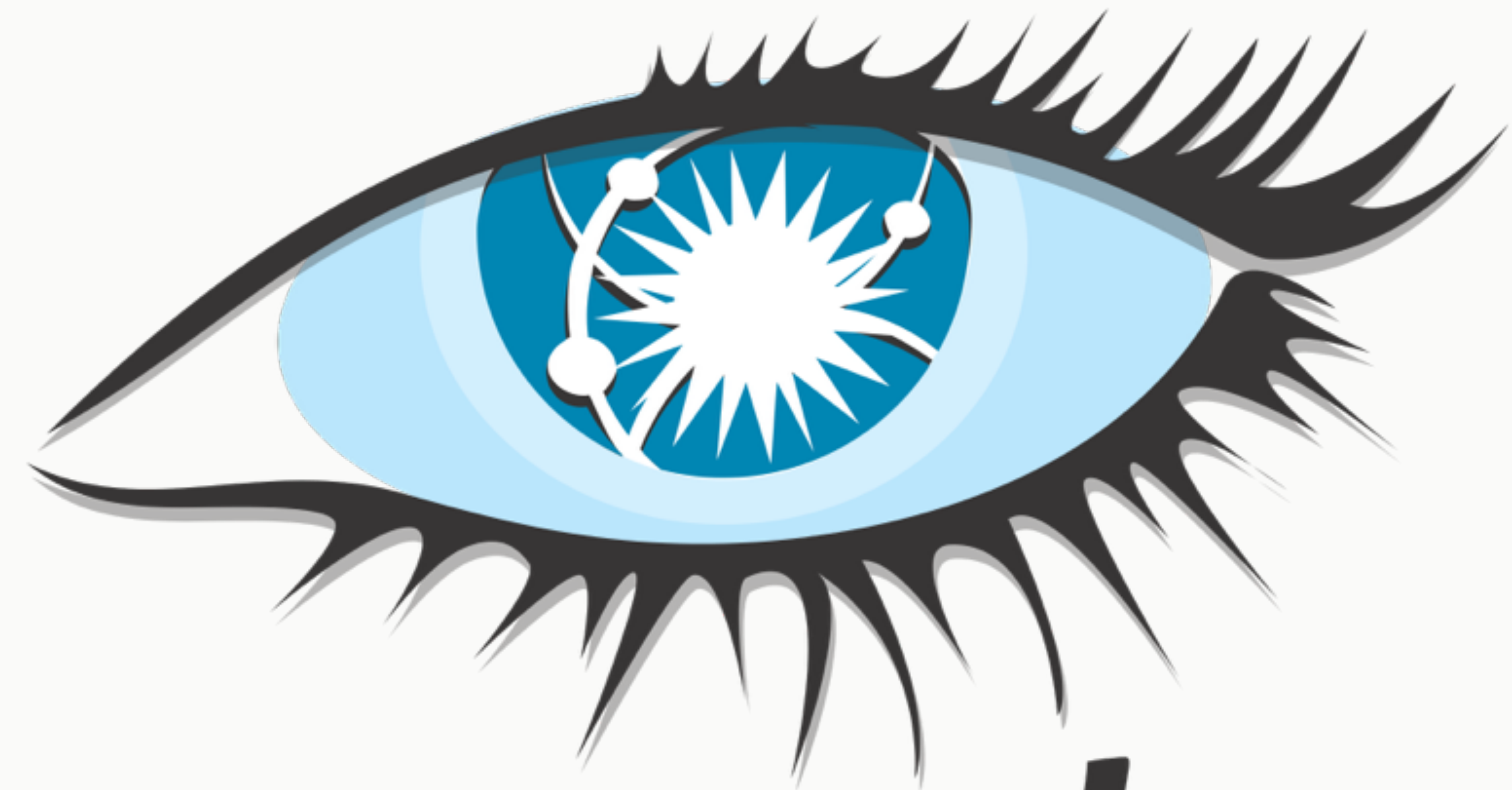


A pile of cats

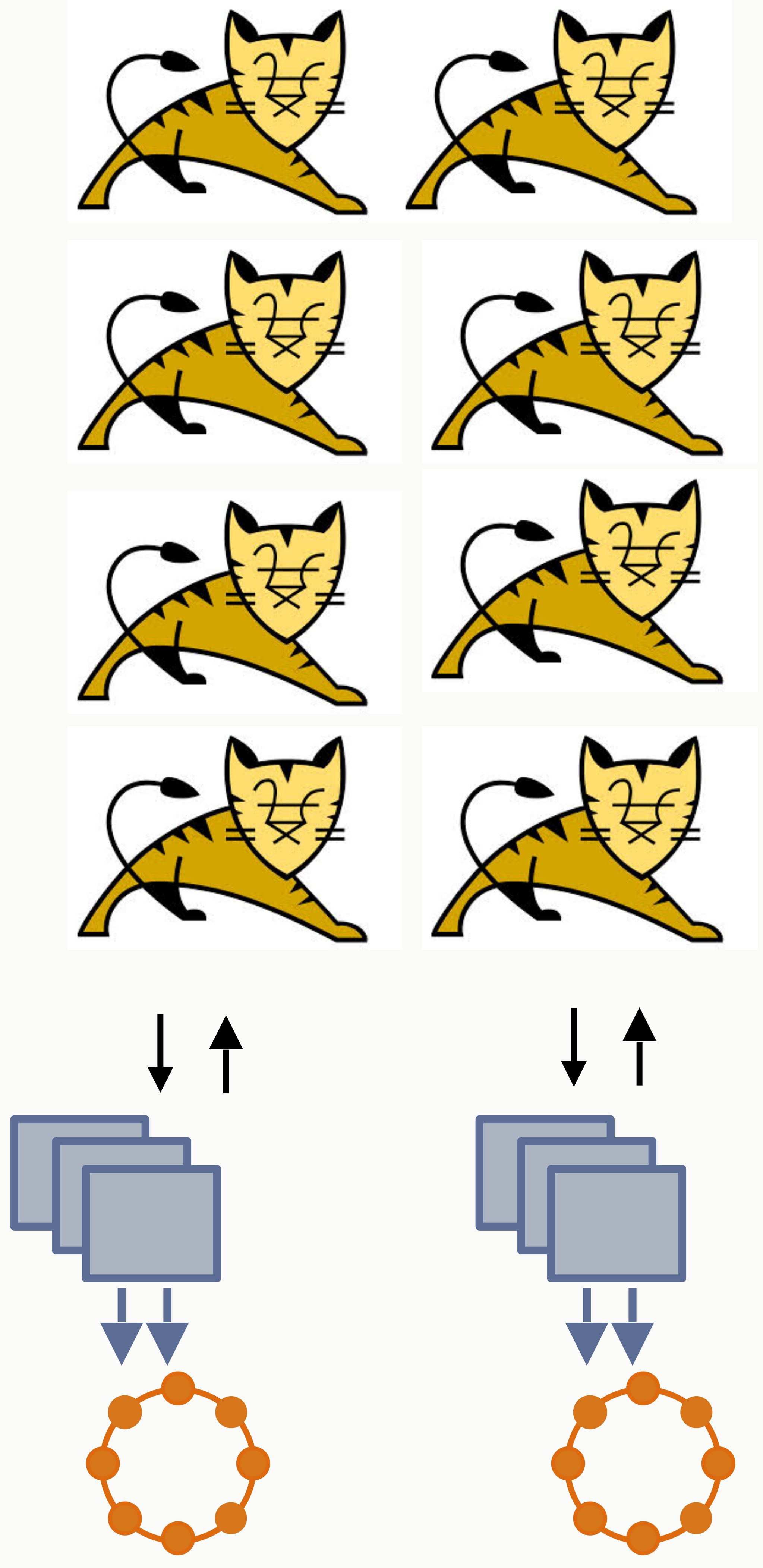
DATASTAX

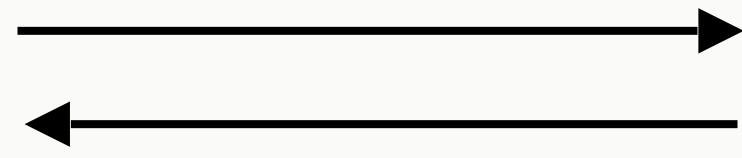
The solution

Micro(ish)services +



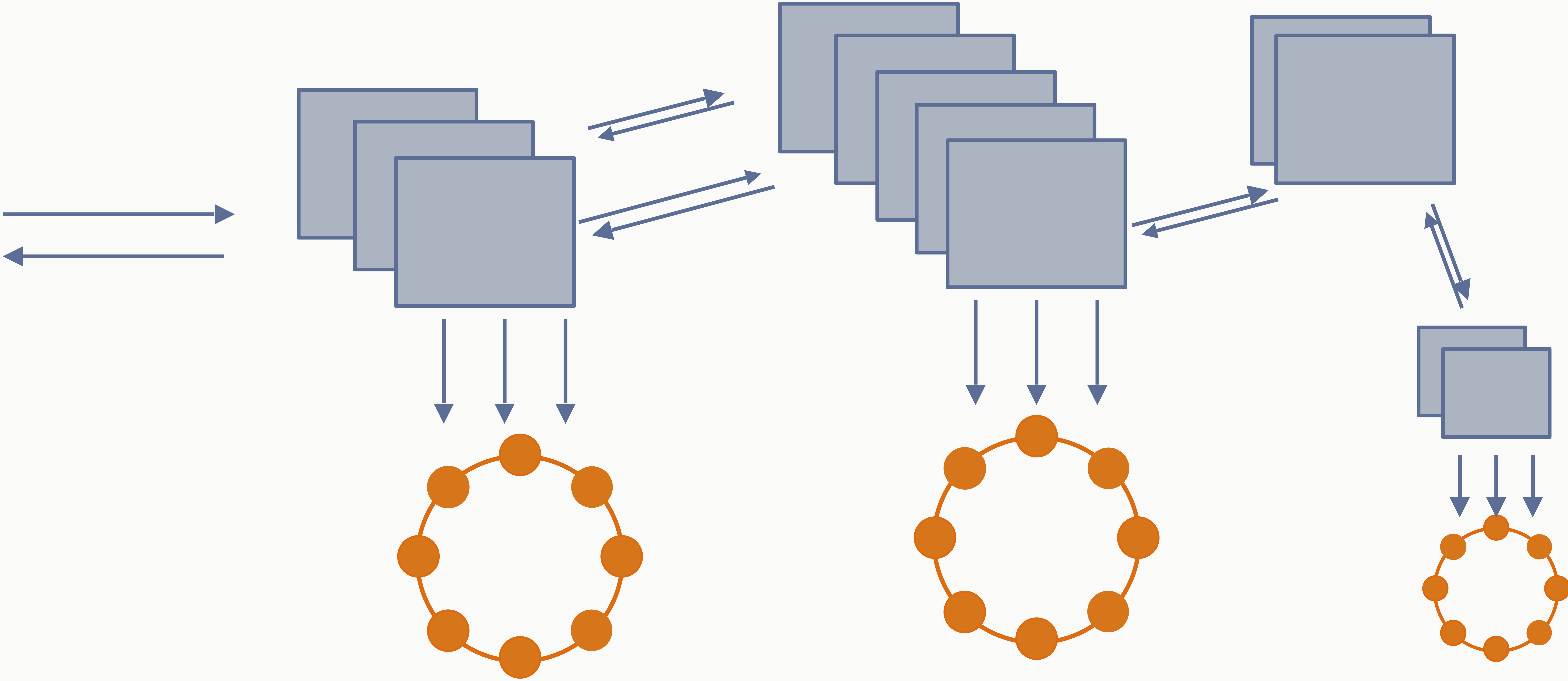
Strangulation





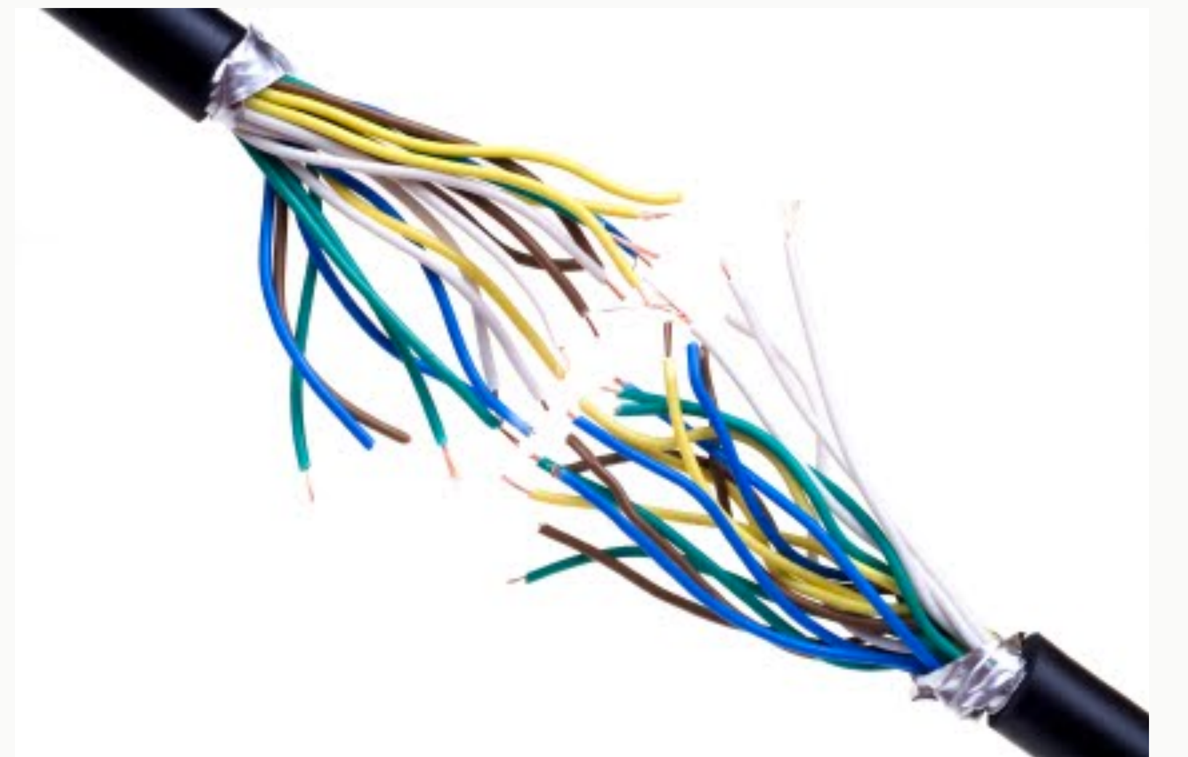
The new

DATASTAX



@chbatey

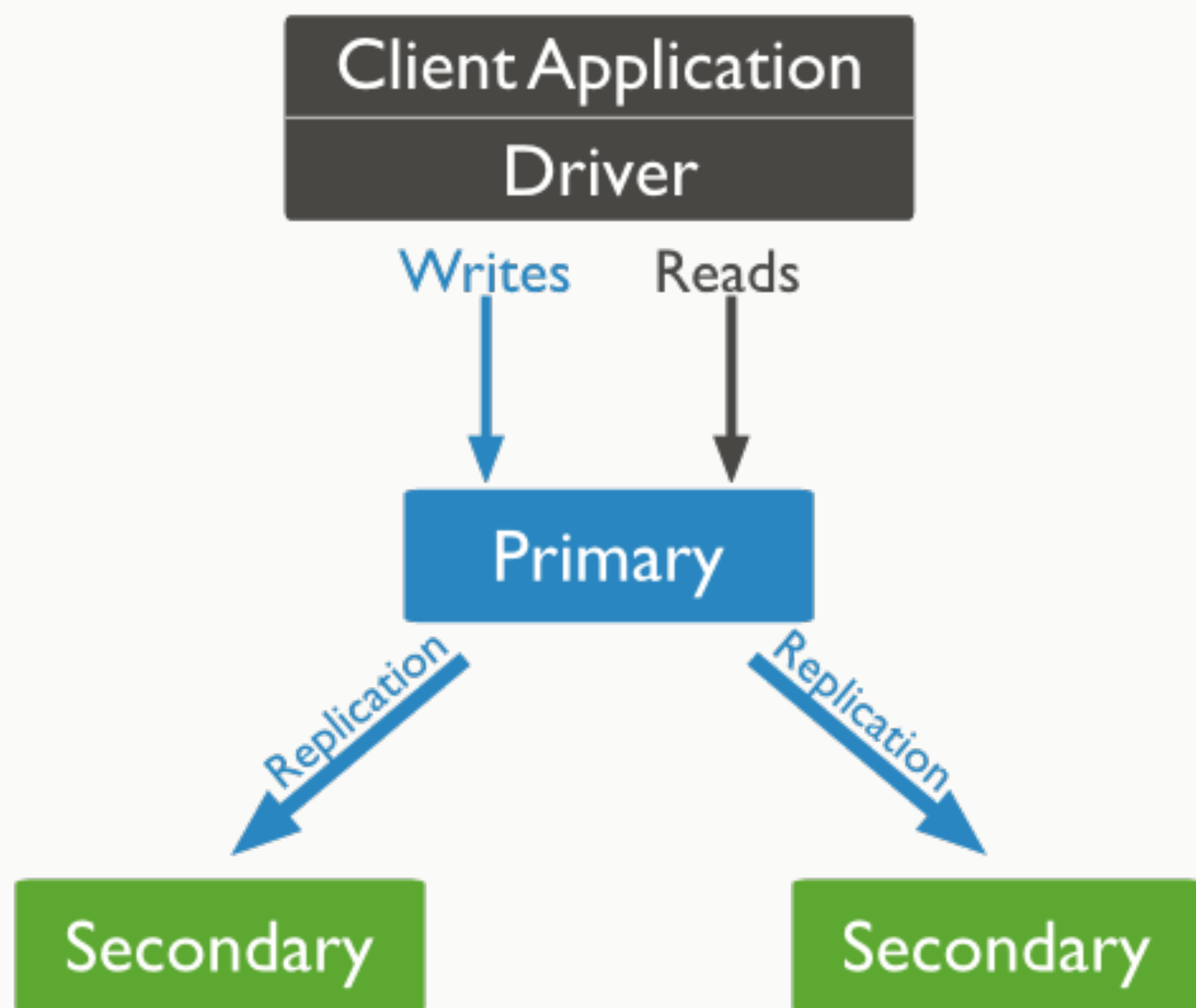
Faults

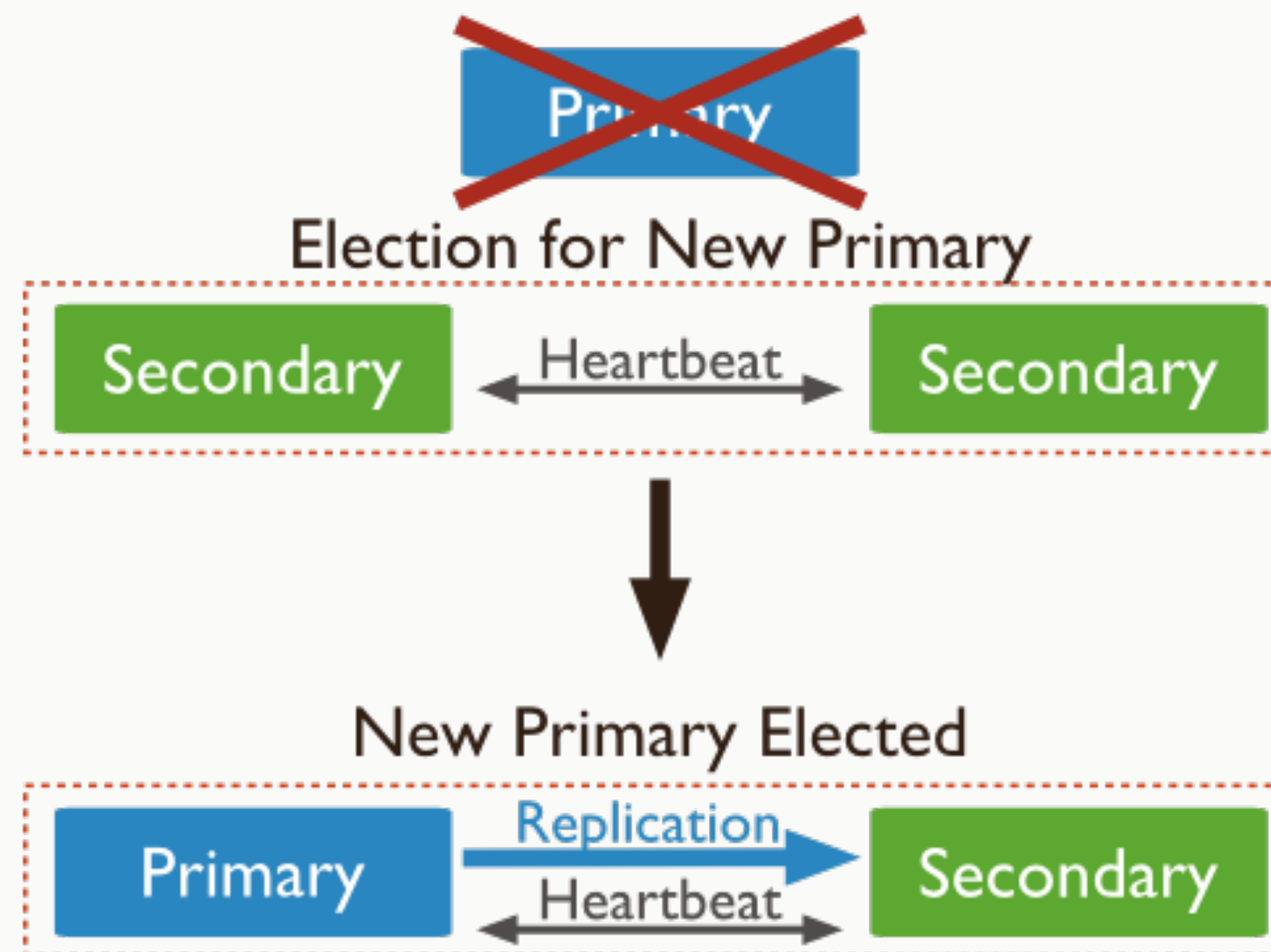


How do you make a Database HA?

DATASTAX

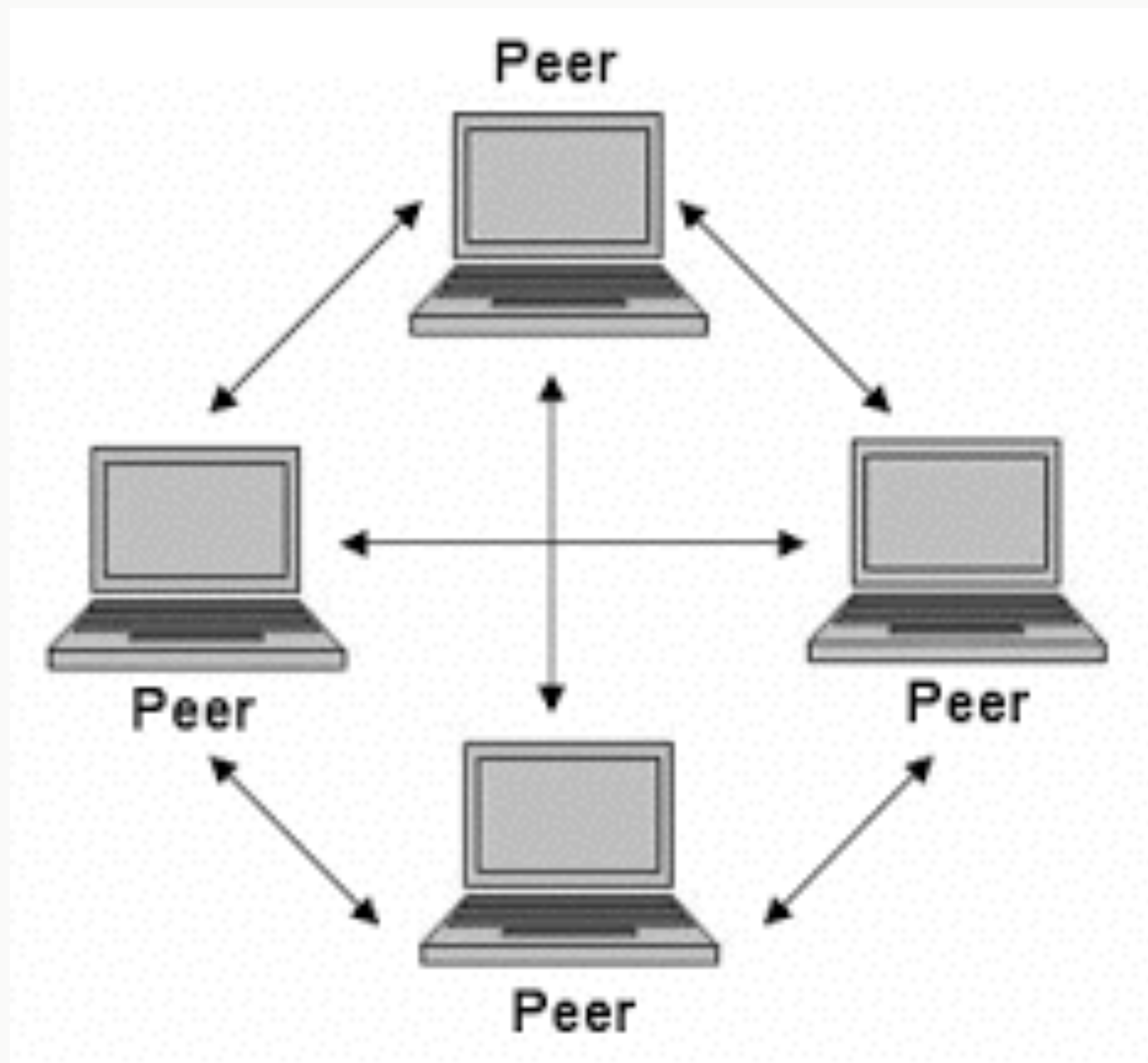
Master/slave





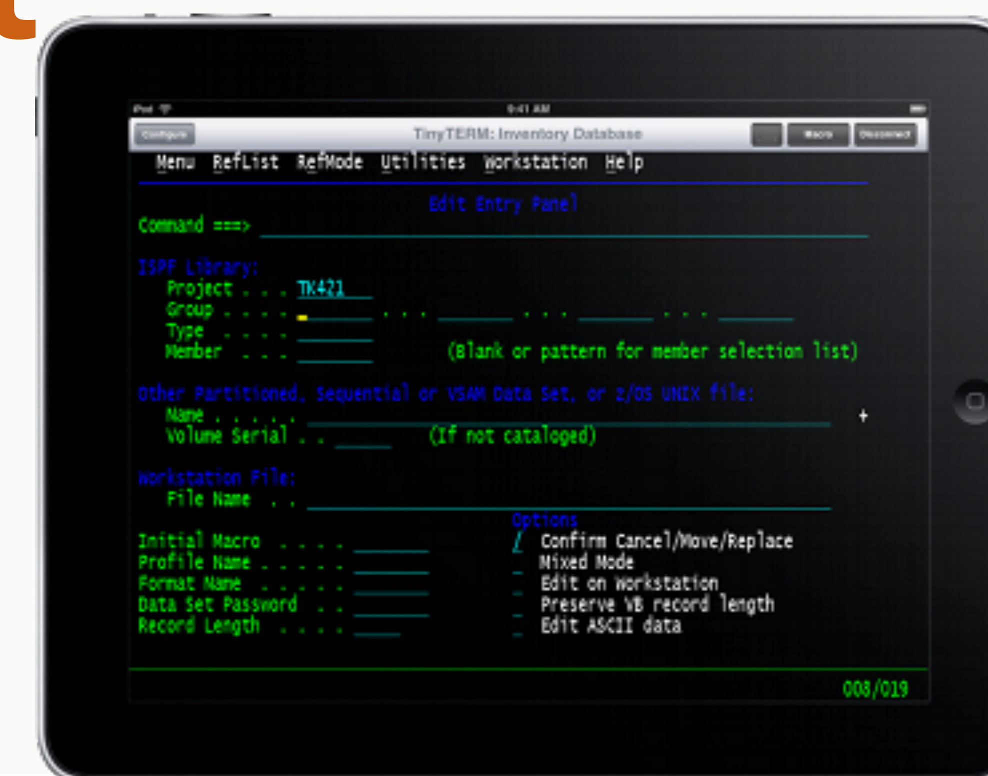
- Master serves all writes
- Read from master and optionally slaves

Peer-to-Peer



- No master
- Read/write to any
- Consistency?

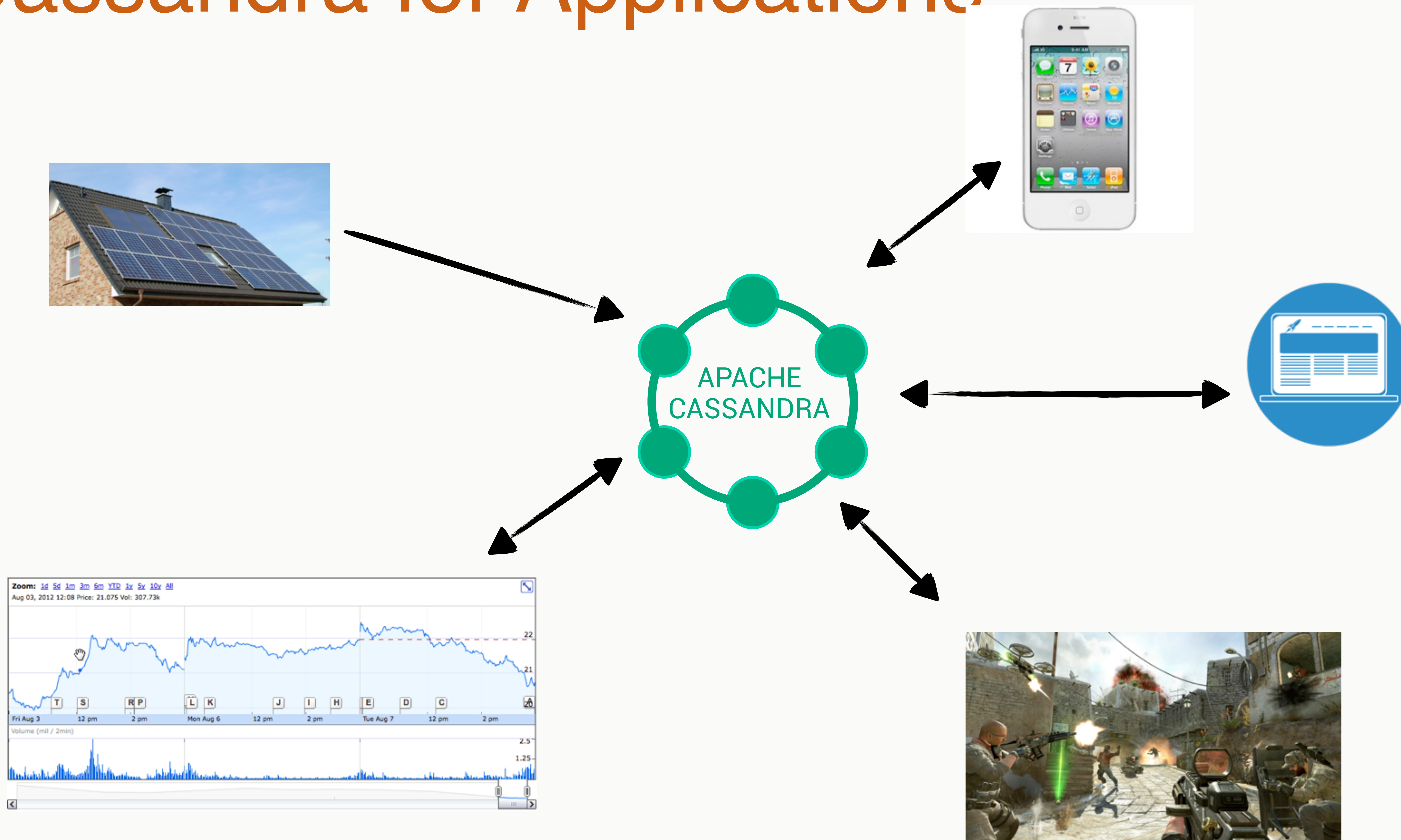
Eventual consistency, deal with it



Cassandra

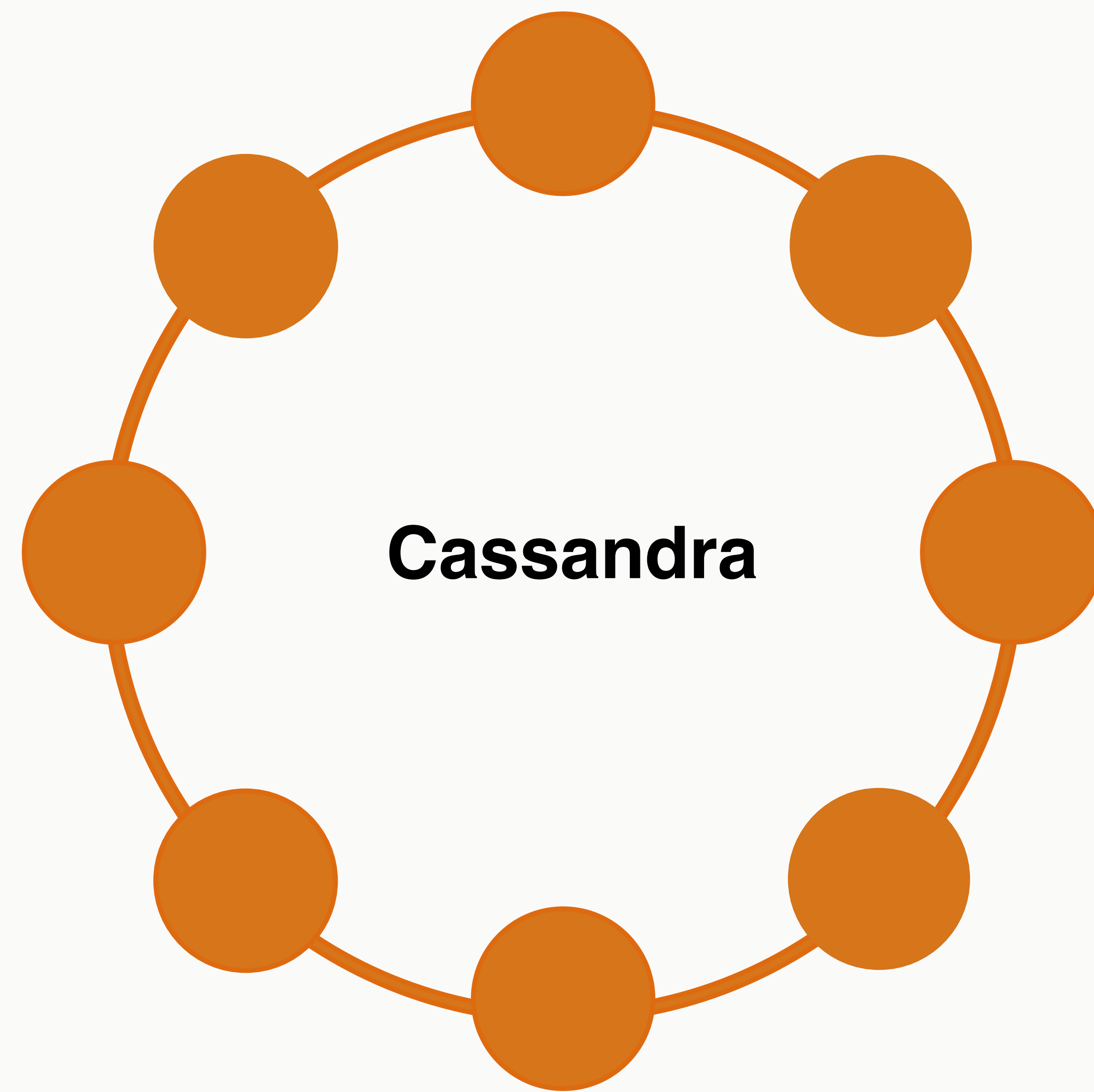
Cassandra for Applications

DATASTAX



@chbatey

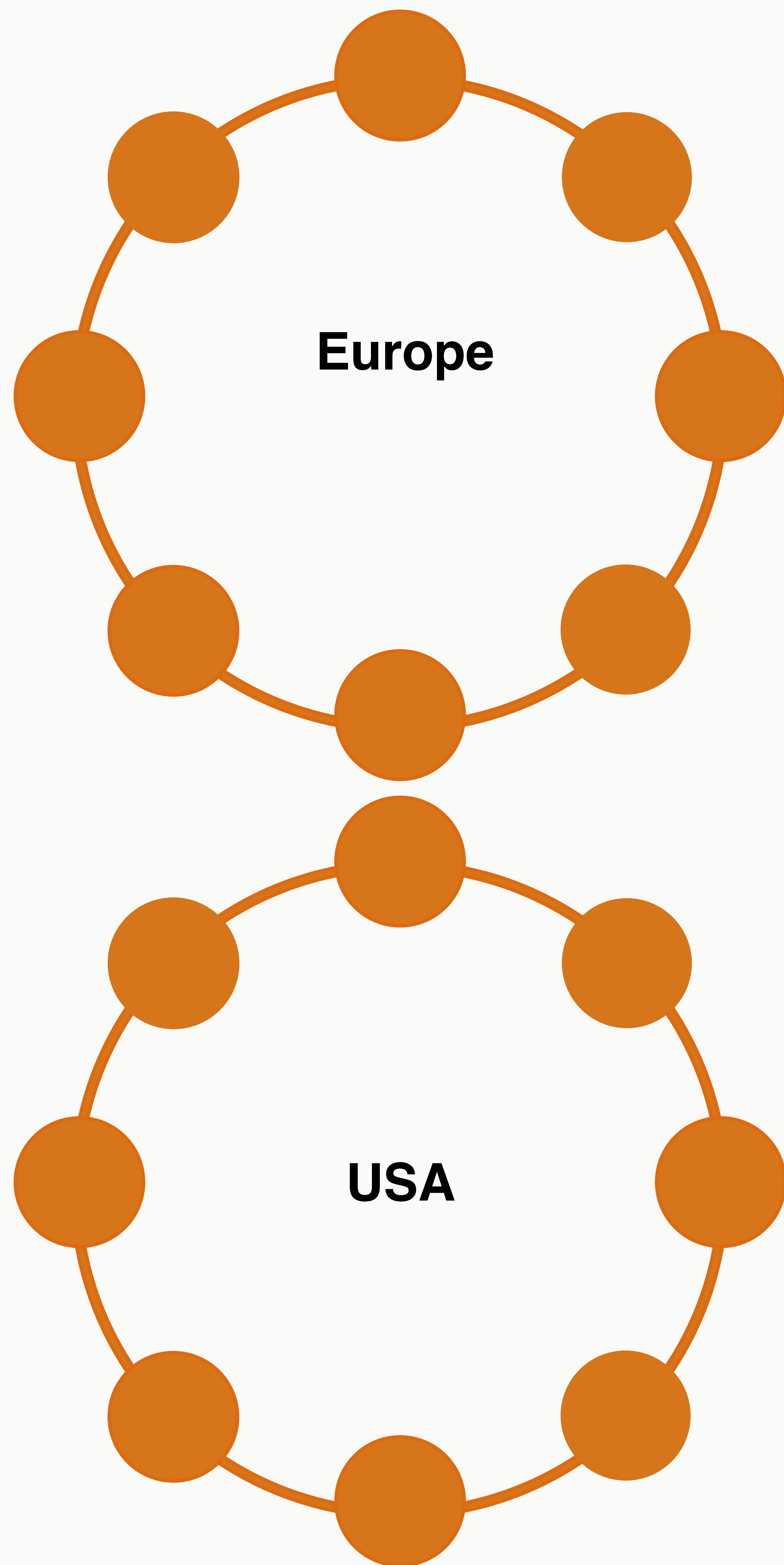
Cassandra



- Distributed masterless database (Dynamo)
- Column family data model (Google BigTable)

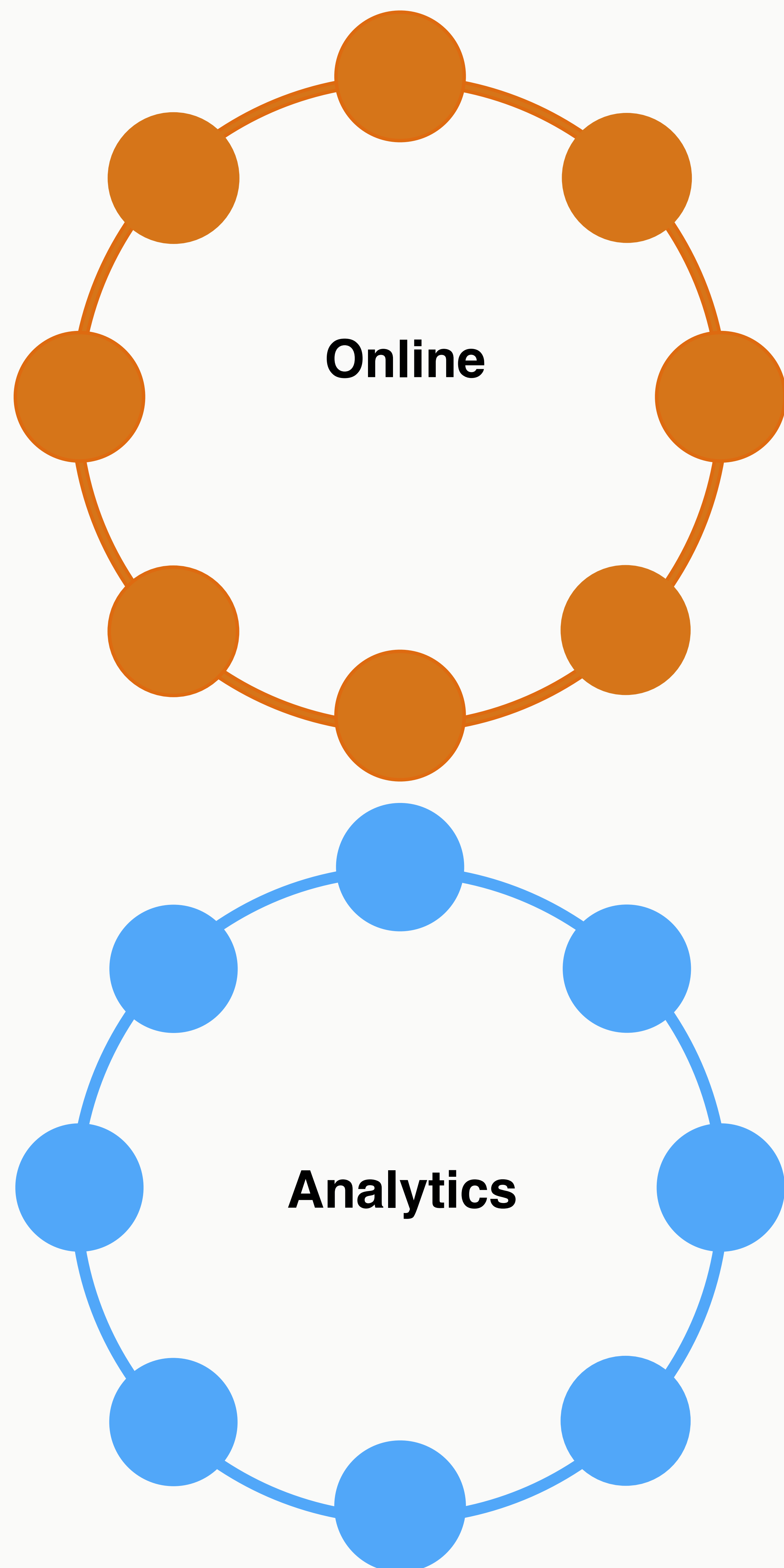
Datacenter and rack aware

DATASTAX



- Distributed master less database (Dynamo)
- Column family data model (Google BigTable)
- Multi data centre replication built in from the start

Cassandra



- Distributed master less database (Dynamo)
- Column family data model (Google BigTable)
- Multi data centre replication built in from the start
- Analytics with Apache Spark

Dynamo 101

DATASTAX

Dynamo: Amazon's Highly Available Key-value Store

Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels

Amazon.com

ABSTRACT

Reliability at massive scale is one of the biggest challenges we face at Amazon.com, one of the largest e-commerce operations in the world; even the slightest outage has significant financial consequences and impacts customer trust. The Amazon.com platform, which provides services for many web sites worldwide, is implemented on top of an infrastructure of tens of thousands of servers and network components located in many datacenters around the world. At this scale, small and large components fail continuously and the way persistent state is managed in the face of these failures drives the reliability and scalability of the software systems.

This paper presents the design and implementation of Dynamo, a highly available key-value storage system that some of Amazon's core services use to provide an "always-on" experience. To achieve this level of availability, Dynamo sacrifices consistency under certain failure scenarios. It makes extensive use of object versioning and application-assisted conflict resolution in a manner that provides a novel interface for developers to use.

One of the lessons our organization has learned from operating Amazon's platform is that the reliability and scalability of a system is dependent on how its application state is managed. Amazon uses a highly decentralized, loosely coupled, service oriented architecture consisting of hundreds of services. In this environment there is a particular need for storage technologies that are always available. For example, customers should be able to view and add items to their shopping cart even if disks are failing, network routes are flapping, or data centers are being destroyed by tornados. Therefore, the service responsible for managing shopping carts requires that it can always write to and read from its data store, and that its data needs to be available across multiple data centers.

Dealing with failures in an infrastructure comprised of millions of components is our standard mode of operation; there are always a small but significant number of server and network components that are failing at any given time. As such Amazon's software systems need to be constructed in a manner that treats failure handling as the normal case without impacting availability or performance.

DATASTAX

Dynamo 101

- The parts Cassandra took
  - Data distribution: consistent hashing
  - Availability + Partition tolerance: replication
  - ~~Gossip~~
  - ~~Hinted handoff~~
  - ~~Anti-entropy repair~~
- And the parts it left behind
  - Key/Value
  - Vector clocks

Picking the right nodes

DATASTAX

- You don't want a full table scan on a 1000 node cluster!
- Dynamo to the rescue: Consistent Hashing

Murmer3 Example

Primary Key

- **Data:**

jim	age: 36	car: ford	gender: M
carol	age: 37	car: bmw	gender: F
johnny	age: 12	gender: M	
suzy:	age: 10	gender: F	

- **Murmer3 Hash Values:**

Primary Key	Murmur3 hash value
jim	350
carol	998
johnny	50
suzy	600

Real hash range: -9223372036854775808 to 9223372036854775807

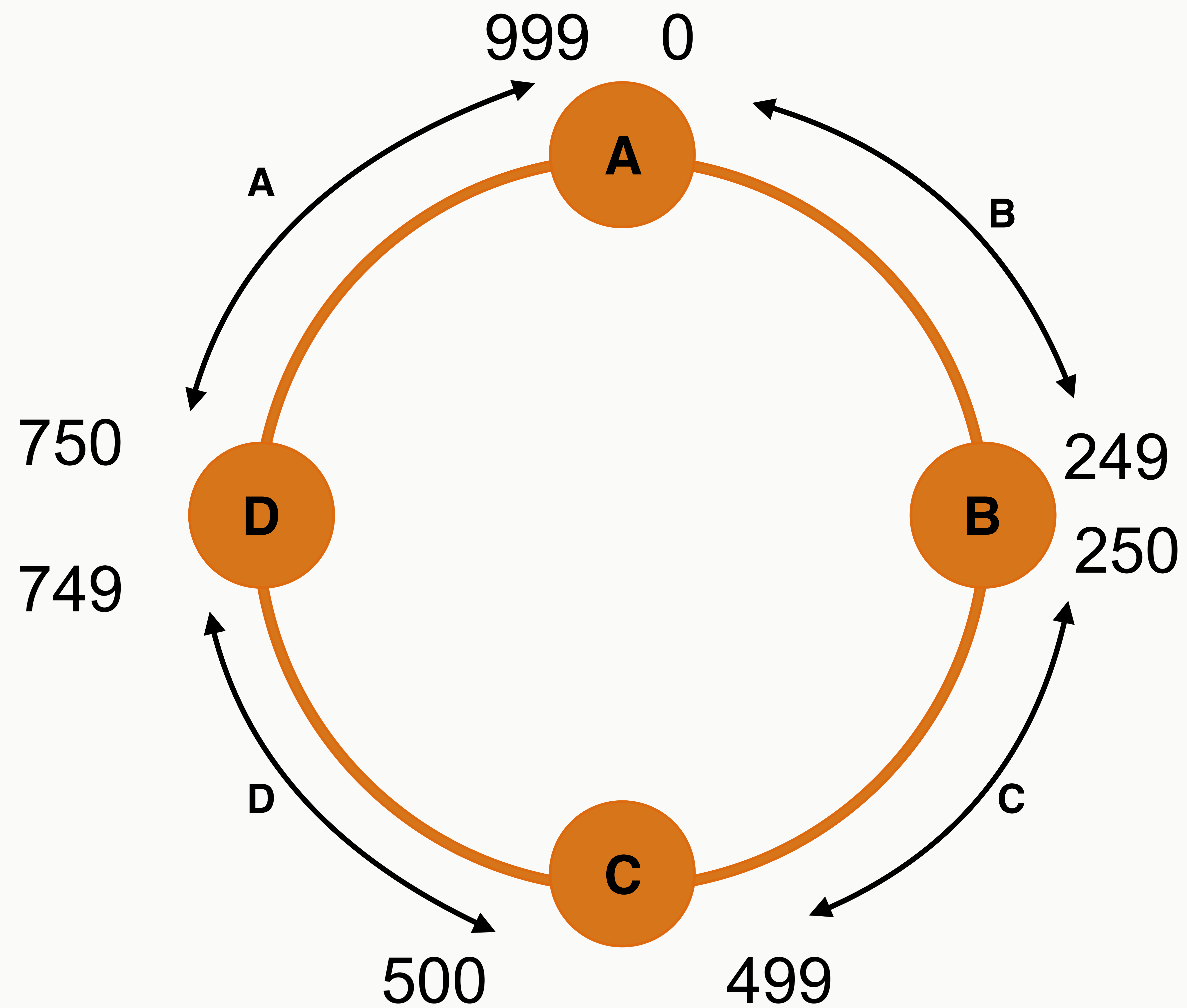
Murmer3 Example

Four node cluster:

Node	Murmur3 start range	Murmur3 end range
A	0	249
B	250	499
C	500	749
D	750	999

Pictures are better

DATASTAX



@chbatey

Murmer3 Example

DATASTAX

Data is distributed as:

Node	Start range	End range	Primary key	Hash value
A	0	249	johnny	50
B	250	499	jim	350
C	500	749	suzy	600
D	750	999	carol	998

DATASTAX

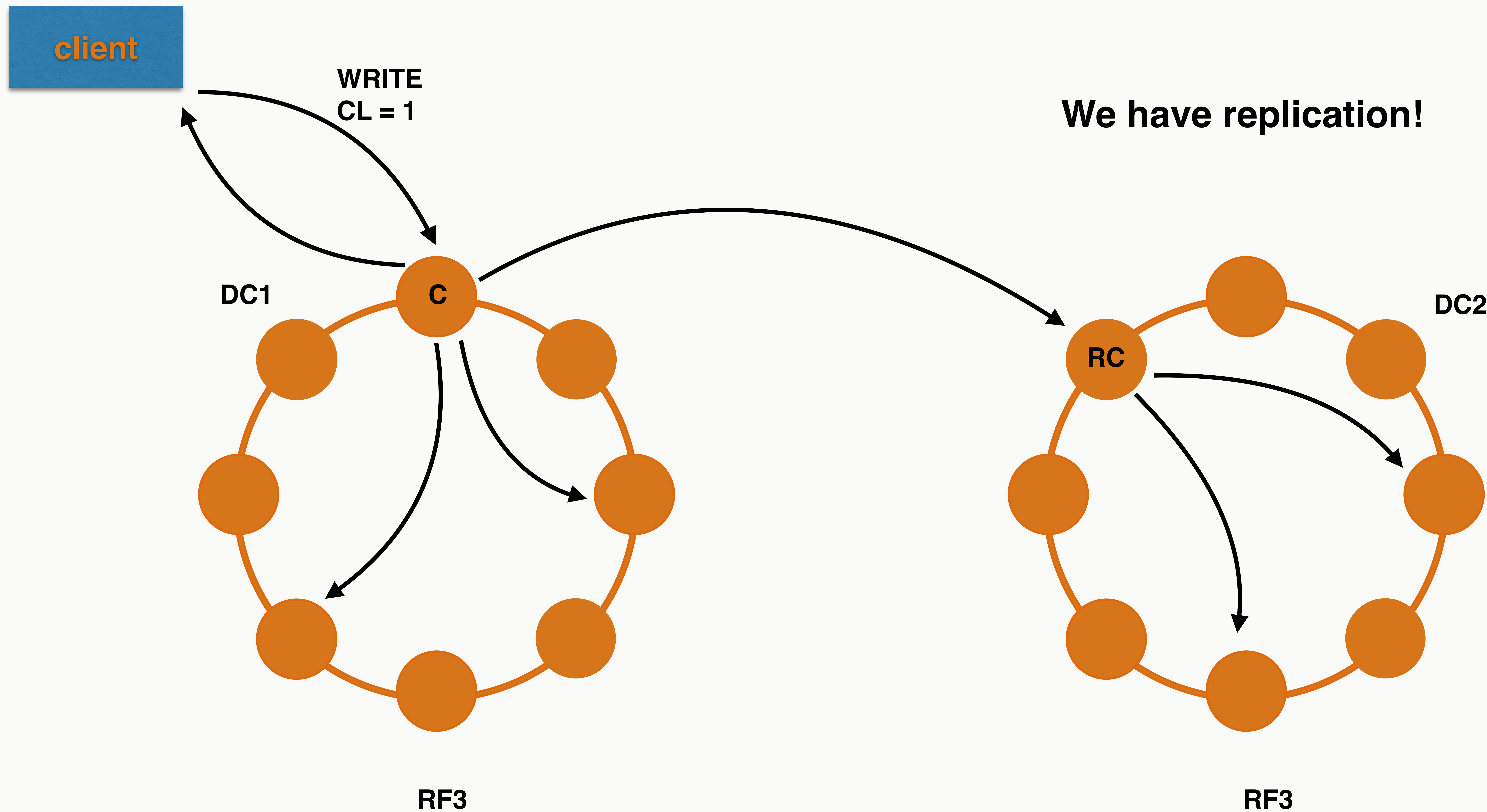
Replication

Replication strategy

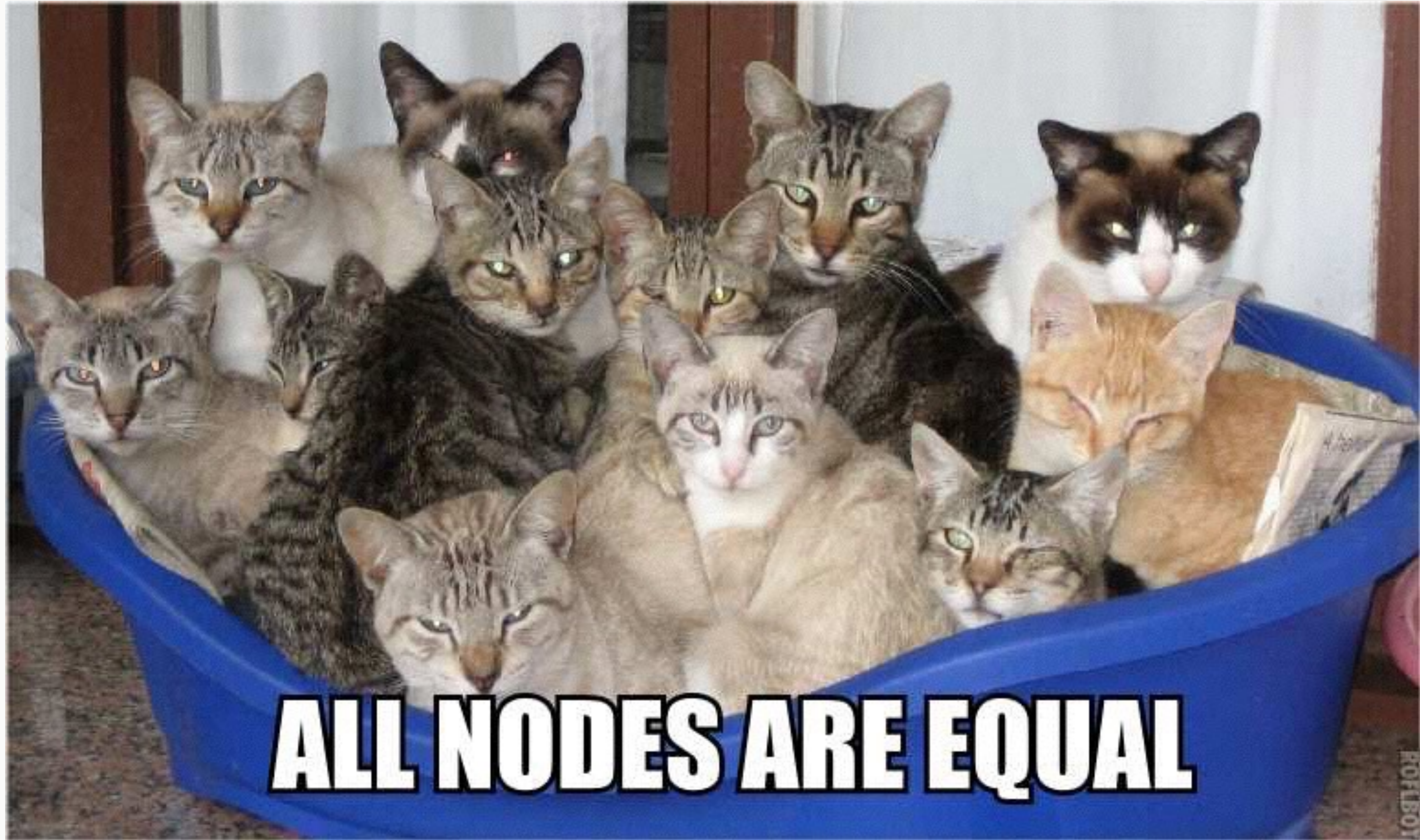
DATASTAX

- Simple
  - Give it to the next node in the ring
  - Don’t use this in production
- NetworkTopology
  - Every Cassandra node knows its DC and Rack
  - Replicas won’t be put on the same rack unless Replication Factor > # of racks
  - Unfortunately Cassandra can’t create servers and racks on the fly to fix this :(

Replication



DATASTAX


ALL NODES ARE EQUAL

Tunable Consistency

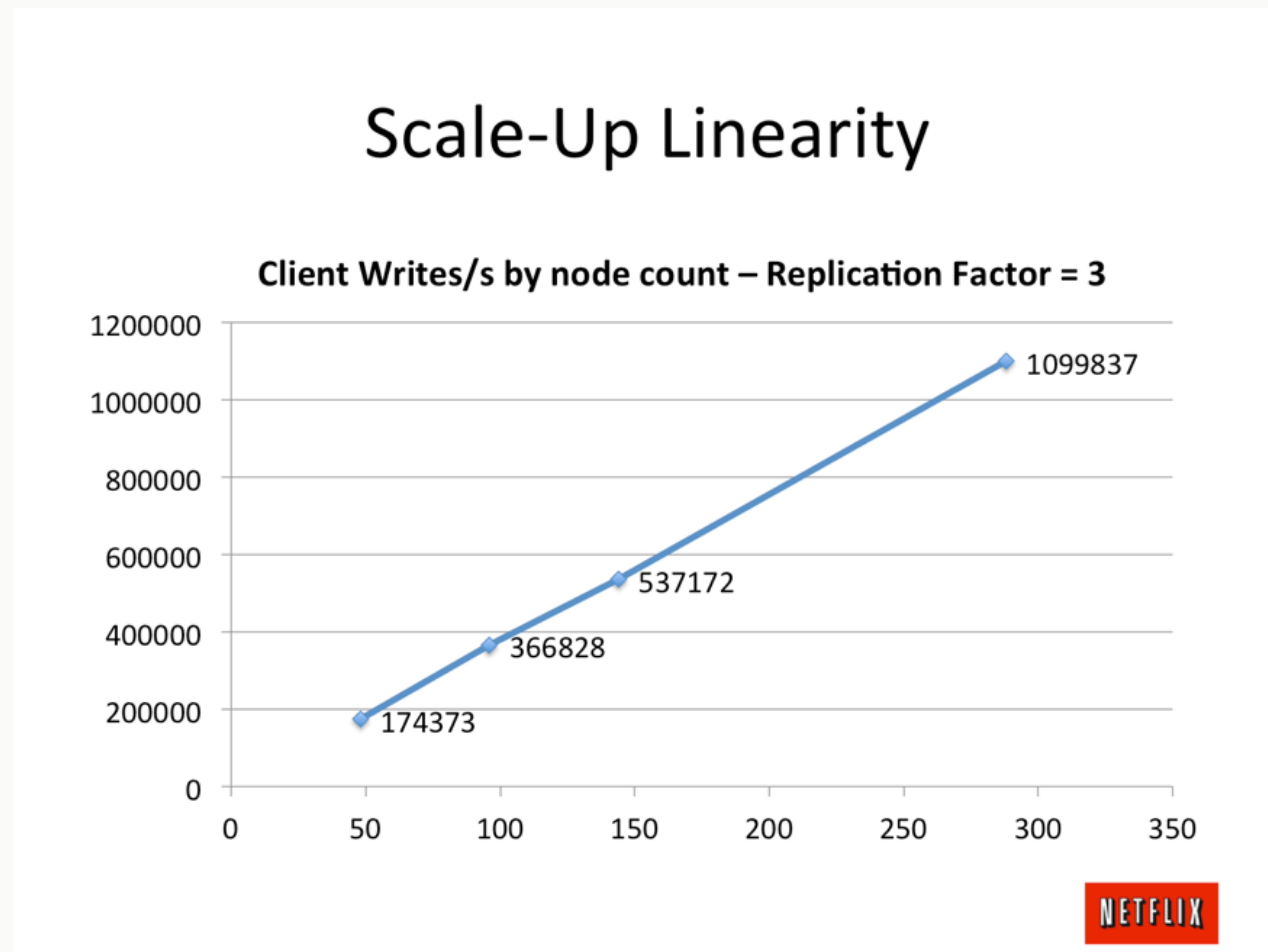
DATASTAX

- Data is replicated N times
- Every query that you execute you give a consistency
  - ALL
  - QUORUM
  - LOCAL_QUORUM
  - ONE
- **Christos Kalantzis** Eventual Consistency != Hopeful Consistency: http://youtu.be/A6qzx_HE3EU?list=PLqcm6qE9lgKJzVvwHprow9h7KMpb5hcUU

Scaling shouldn't be hard

DATASTAX

- Throw more nodes at a cluster
- Bootstrapping + joining the ring
  - For large data sets this can take some time



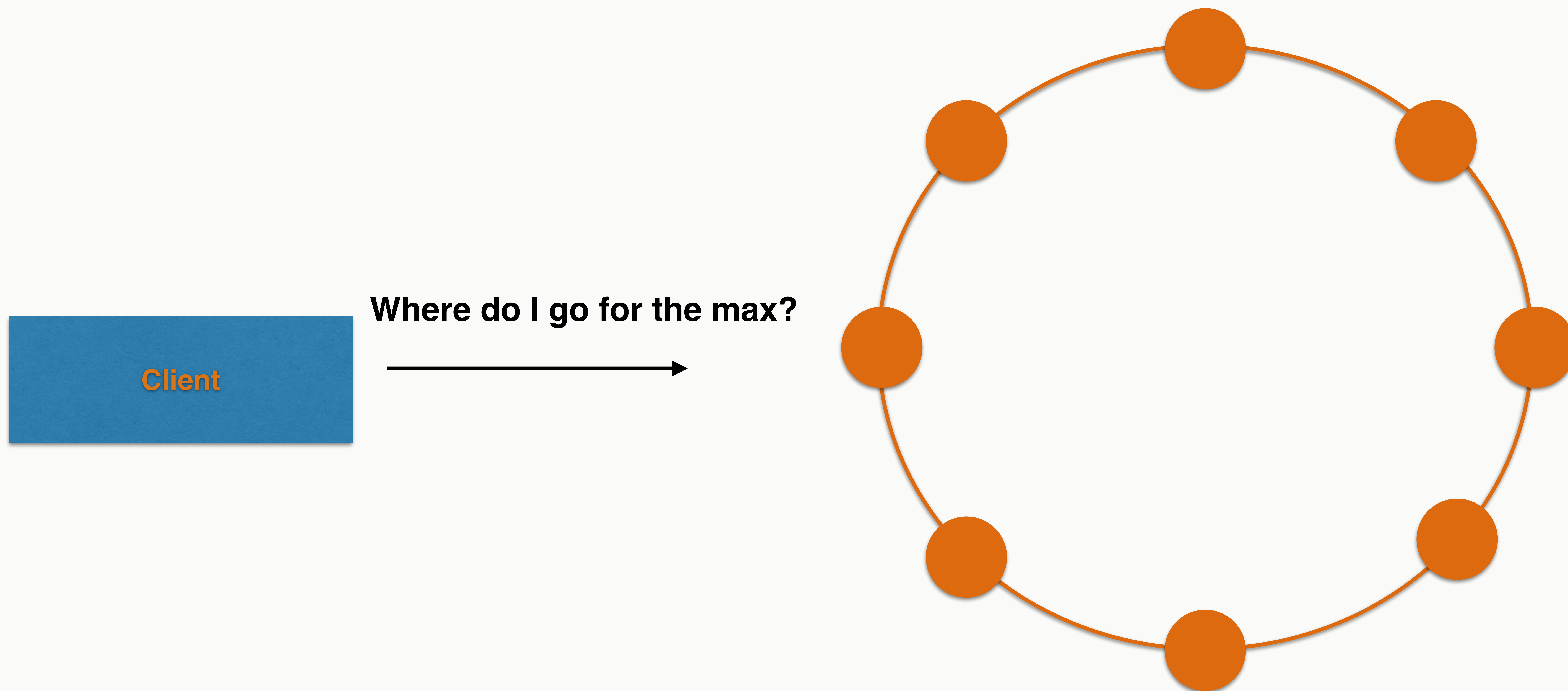
Data modelling

You must denormalise

DATASTAX

Cassandra can not join or aggregate

DATASTAX



@chbatey

CQL

DATASTAX

- Cassandra Query Language
  - SQL like query language
- Keyspace – analogous to a schema
  - The keyspace determines the RF (replication factor)
- Table – looks like a SQL Table

```
CREATE TABLE scores (
  name text,
  score int,
  date timestamp,
  PRIMARY KEY (name, score)
);
```

```
INSERT INTO scores (name, score, date)
VALUES ('bob', 42, '2012-06-24');
INSERT INTO scores (name, score, date)
VALUES ('bob', 47, '2012-06-25');
```

```
SELECT date, score FROM scores WHERE name='bob' AND score >= 40;
```

UUID

- Universal Unique ID
  - 128 bit number represented in character form e.g. 99051fe9-6a9c-46c2-b949-38ef78858dd0
- Easily generated on the client
  - Version 1 has a timestamp component (TIMEUUID)
  - Version 4 has no timestamp component

TIMEUUID

DATASTAX

TIMEUUID data type supports Version 1 UUIDs

Generated using time (60 bits), a clock sequence number (14 bits), and MAC address (48 bits)

– **CQL function ‘now()’ generates a new TIMEUUID**

Time can be extracted from TIMEUUID

– **CQL function dateOf() extracts the timestamp as a date**

TIMEUUID values in clustering columns or in column names are ordered based on time

– **DESC order on TIMEUUID lists most recent data first**

© 2014 DataStax, All Rights Reserved.

Data Model - User Defined Types

DATASTAX

```
CREATE TYPE address (
  street text,
  city text,
  zip_code int,
  country text,
  cross_streets set<text>
);
```

- Complex data in one place
- No multi-gets (multi-partitions)
- Nesting!

Time-to-Live (TTL)

DATASTAX

TTL a row:

```
INSERT INTO users (id, first, last) VALUES (‘abc123’, ‘catherine’, ‘cachart’)
USING TTL 3600; // Expires data in one hour
```

TTL a column:

```
UPDATE users USING TTL 30 SET last = ‘miller’ WHERE id = ‘abc123’
```

- TTL in seconds
- **Can also set default TTL at a table level**
- Expired columns/values automatically deleted
- With no TTL specified, columns/values never expire
- TTL is useful for automatic deletion
- Re-inserting the same row before it expires will overwrite TTL

But isn't Cassandra a columnar store?

DATASTAX

Storing weather data

DATASTAX

```
CREATE TABLE raw_weather_data (
   weather_station text,
   year int,
   month int,
   day int,
   hour int,
   temp double,
   PRIMARY KEY ((weather_station), year, month, day, hour)
) WITH CLUSTERING ORDER BY (year DESC, month DESC, day DESC, hour
DESC);
```

Primary key relationship

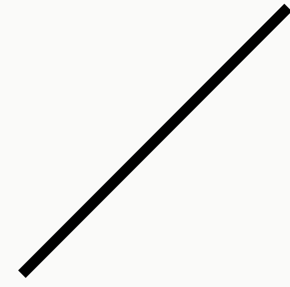
DATASTAX

```
PRIMARY KEY ((weatherstation_id),year,month,day,hour)
```

Primary key relationship

DATASTAX

```
PRIMARY KEY ((weatherstation_id),year,month,day,hour)
```



Partition Key

Primary key relationship

DATASTAX

```
PRIMARY KEY ((weatherstation_id),year,month,day,hour)
```

Partition Key

Clustering Columns

```
WITH CLUSTERING ORDER BY (year DESC, month DESC, day DESC, hour DESC);
```

Primary key relationship

DATASTAX

```
PRIMARY KEY ((weatherstation_id),year,month,day,hour)
```

Partition Key

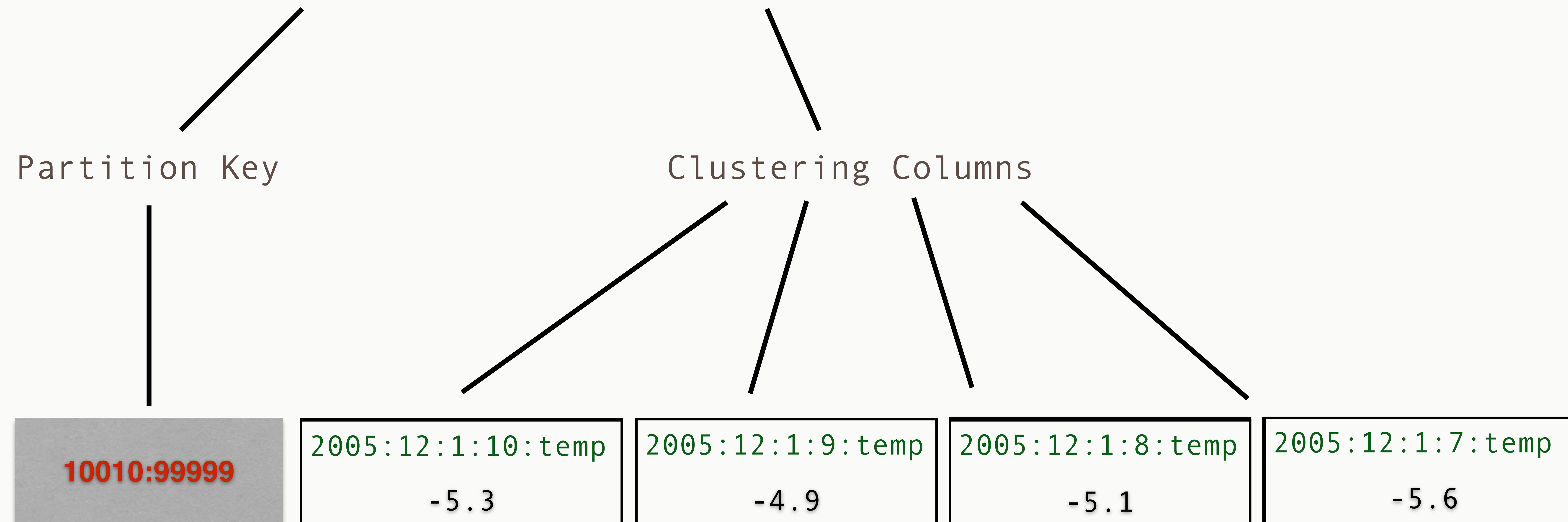
Clustering Columns

10010:99999

```
WITH CLUSTERING ORDER BY (year DESC, month DESC, day DESC, hour DESC);
```

Primary key relationship

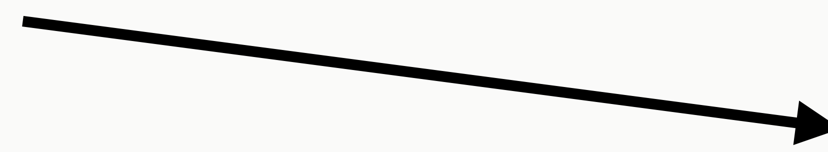
DATASTAX

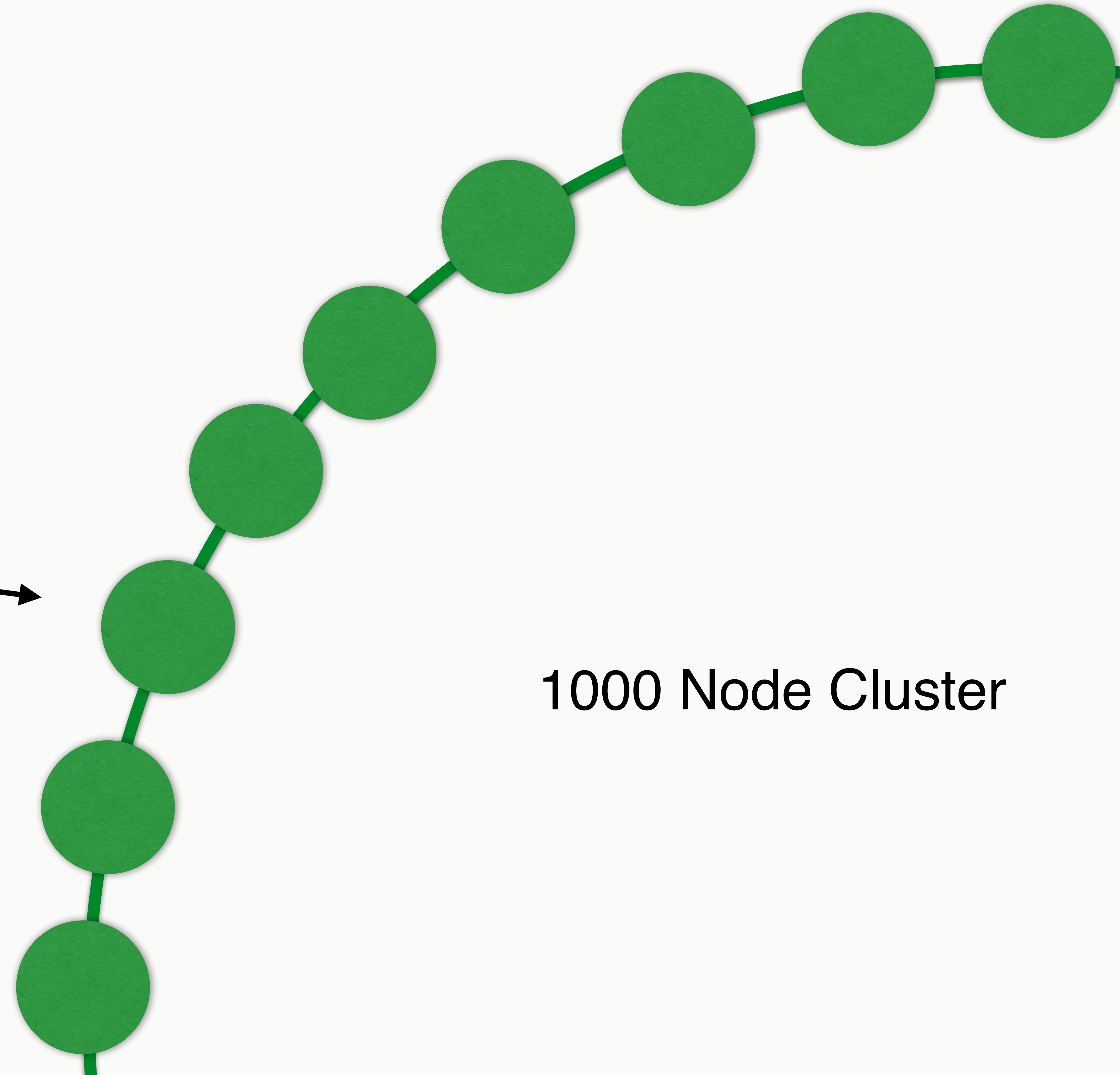



```
WITH CLUSTERING ORDER BY (year DESC, month DESC, day DESC, hour DESC);
```

Data Locality


```
weatherstation_id='10010:99999' ?
```



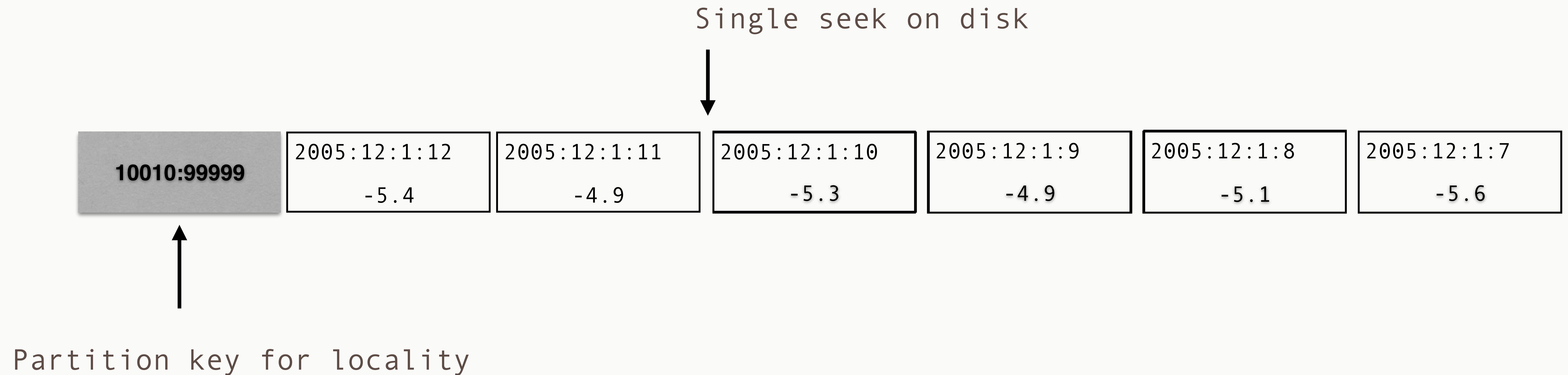


Query patterns

DATASTAX

```
SELECT weatherstation,hour,temperature
FROM raw_weather_data
WHERE weatherstation_id='10010:99999'
AND year = 2005 AND month = 12 AND day = 1
AND hour >= 7 AND hour <= 10;
```

- Range queries
- "Slice" operation on disk



Query patterns

DATASTAX

```
SELECT weatherstation,hour,temperature
FROM raw_weather_data
WHERE weatherstation_id='10010:99999'
AND year = 2005 AND month = 12 AND day = 1
AND hour >= 7 AND hour <= 10;
```

- Range queries
- "Slice" operation on disk

weather_station	hour	temperature
10010:99999	10	-5.3
10010:99999	9	-4.9
10010:99999	8	-5.1
10010:99999	7	-5.6

Summary

DATASTAX

- Cassandra is a shared nothing masterless datastore
- Availability a.k.a up time is king
- Biggest hurdle is learning to model differently
- Modern drivers make it easy to work with

Question time

- Twitter: @chbatey
- Blog: http://christopher-batey.blogspot.co.uk/
- Cassandra resources: http://planetcassandra.org/

DATASTAX

Summary

DATASTAX

- Cassandra is a shared nothing masterless datastore
- Availability a.k.a up time is king
- Biggest hurdle is learning to model differently
- Modern drivers make it easy to work with

Question time

- Twitter: @chbatey
- Blog: http://christopher-batey.blogspot.co.uk/
- Cassandra resources: http://planetcassandra.org/