CASSANDRA @ INSTAGRAM 2019

Dikang Gu -- Instagram

ABOUT ME

- Apache Cassandra Committer
- Engineering manager @ IG
- HDFS developer @ FB

AGENDA

1 Cassandra usage at Instagram

2 Improvements

3 Challenges

INSTAGRAM

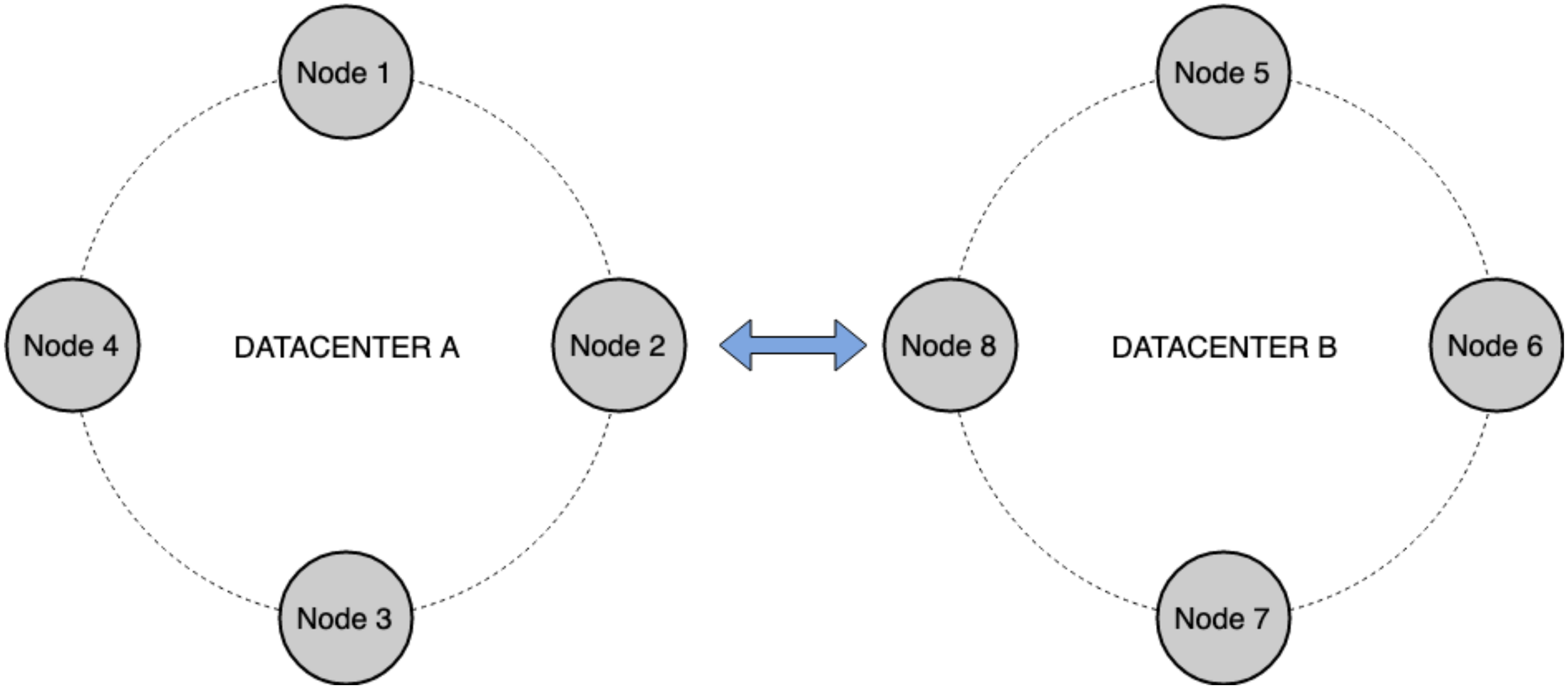
- Launched in 2010
- 1B+ Monthly Active Accounts
- 500M+ Daily Active Accounts of Stories
- IGTV
- IG Shopping

CASSANDRA IN A NUTSHELL

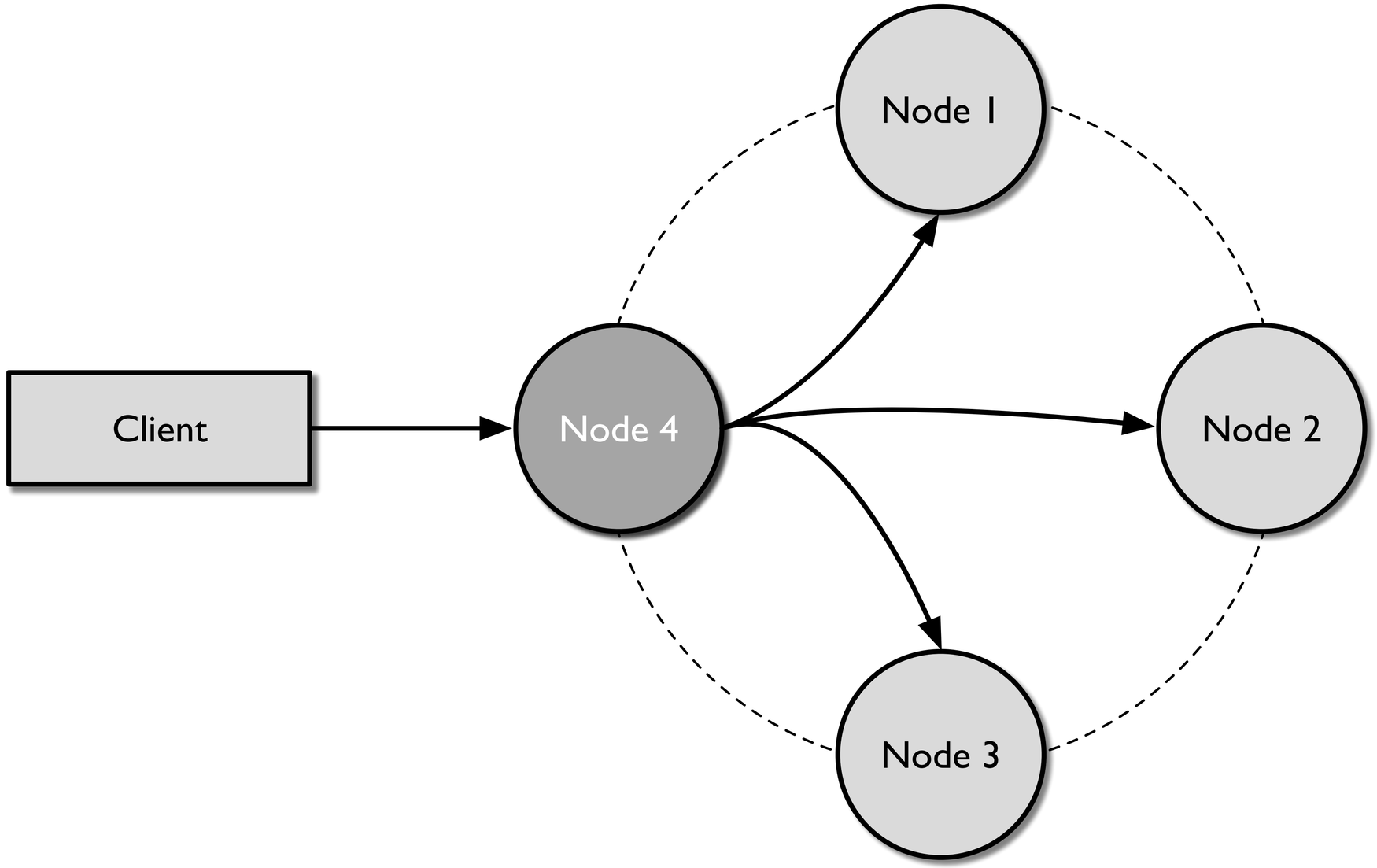
WHAT IS CASSANDRA

Apache Cassandra is robust, high-performance distributed database.

CASSANDRA

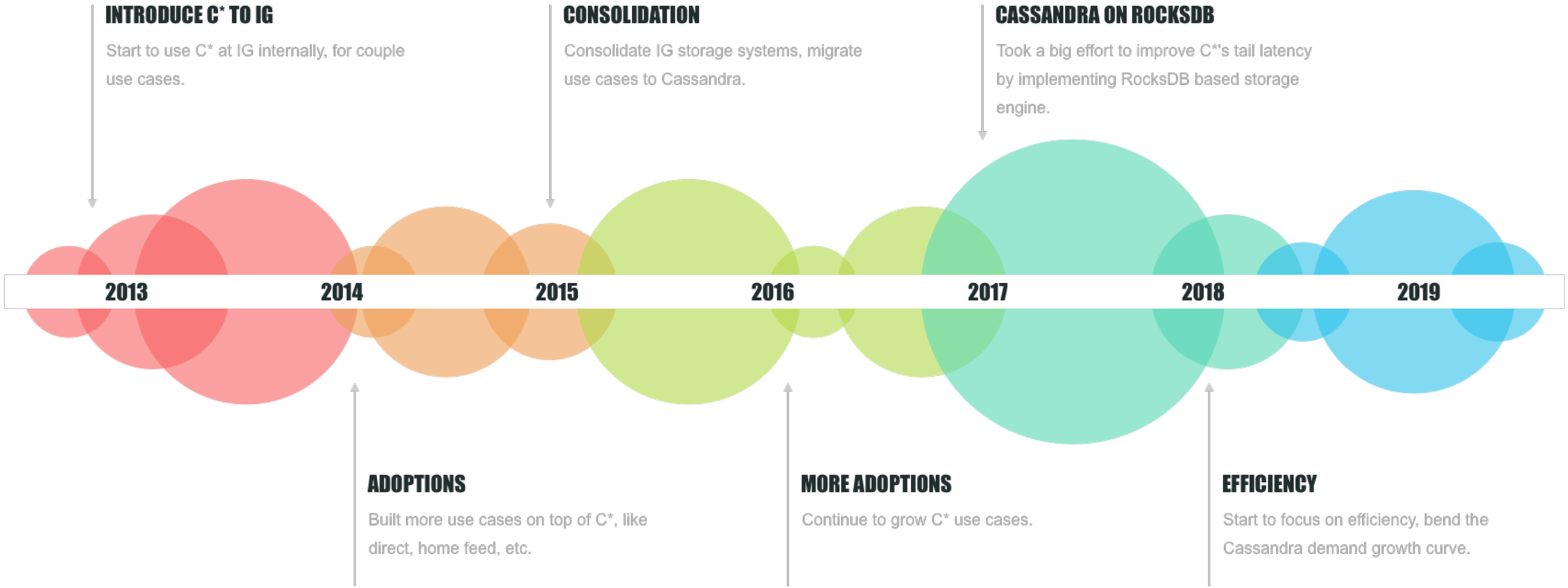

Node 1
Node 5
Node 4
DATACENTER A
Node 2
Node 8
DATACENTER B
Node 6
Node 3
Node 7

CASSANDRA



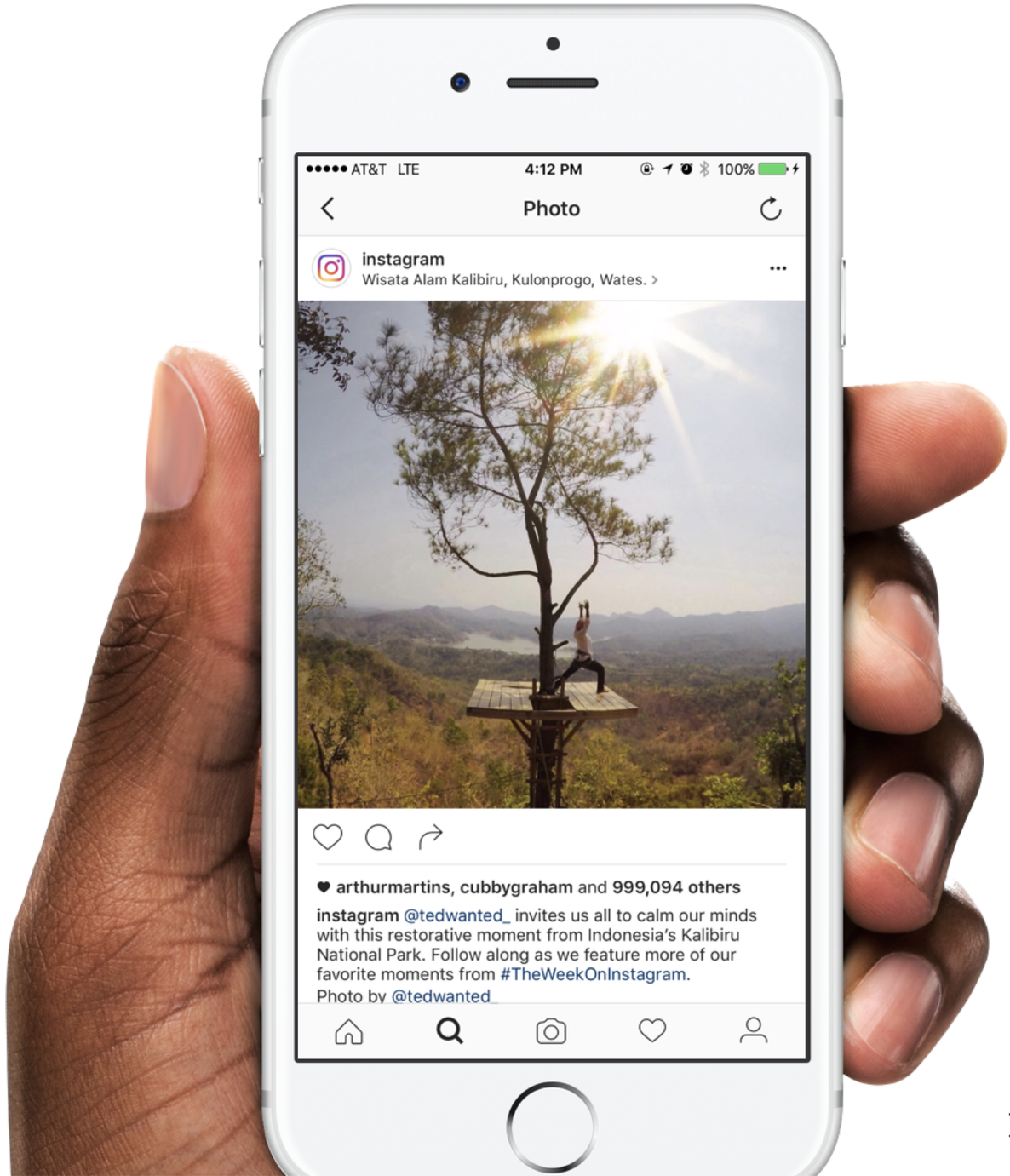
CASSANDRA USAGE @ IG

CASSANDRA HISTORY @ IG



INSTAGRAM DEPLOYMENT

- 1000s of Apache Cassandra instances
- 10s of millions of QPS
- 100s of production use cases
- Petabytes of data
- 5+ Datacenters
- Version 3.0
- Custom storage engine based on RocksDB



USE CASES

USE CASE A

Metadata store

Application uses Cassandra as persisted metadata storage, they store a list of metadata blobs associated with a key, and do point query or range query during the read time.

```
CREATE TABLE keyspace.t1
  (
     key    BIGINT,
     t_id   INT,
     value  BLOB,
     PRIMARY KEY (key, t_id)
  )
```

USE CASE B

Time series store

This type of applications use Cassandra as time series storage, they store a list of activities into Cassandra, sorted by timestamp. This class of use cases usually have high write throughput, which fits well into Cassandra's strength.

```
CREATE TABLE keyspace.t2
  (
     key    BIGINT,
     ts     TIMEUUID,
     value  BLOB,
     PRIMARY KEY (key, ts)
  )
```

USE CASE C

Counter store

This type of applications stores distributed Counters into Cassandra. They issue bump or get requests against the storage.

```
CREATE TABLE keyspace.t3
  (
     key   BIGINT,
     value COUNTER,
     PRIMARY KEY (key)
  )
```

ABSTRACTIONS

CQL

```
SELECT * FROM keyspace.t1 WHERE key = 1 AND t_id = 2;

INSERT INTO keyspace.t1 (key, t_id, value) VALUES (1, 2, "metadata")
```

STORAGE API

Put

MultiGet

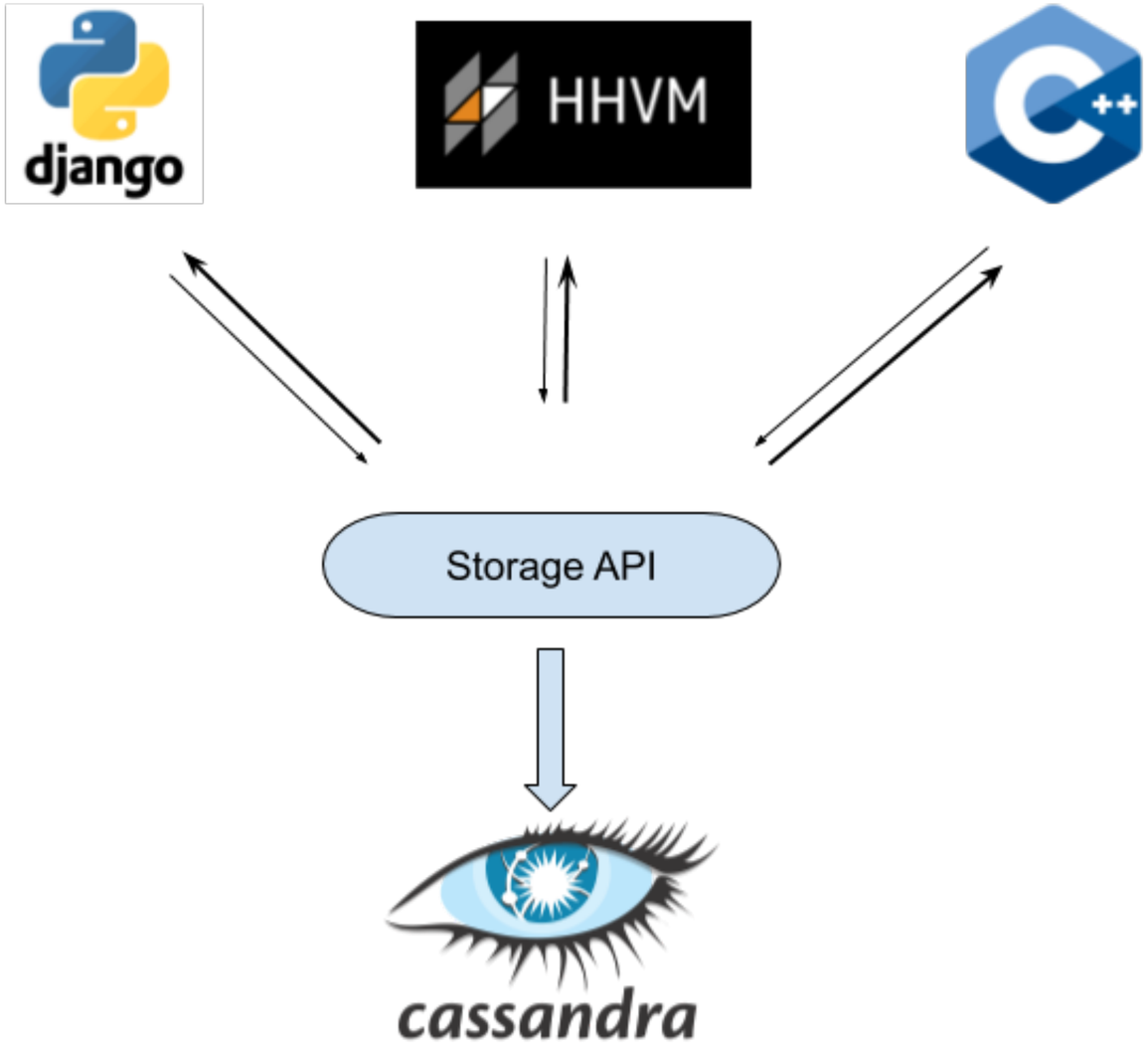
Get

BatchMutate

GetRange

Delete

CLIENTS



RECAP

Cassandra usage at Instagram

- Cassandra in a nutshell
- C* history and deployments within IG
- Typical use cases
- Abstractions

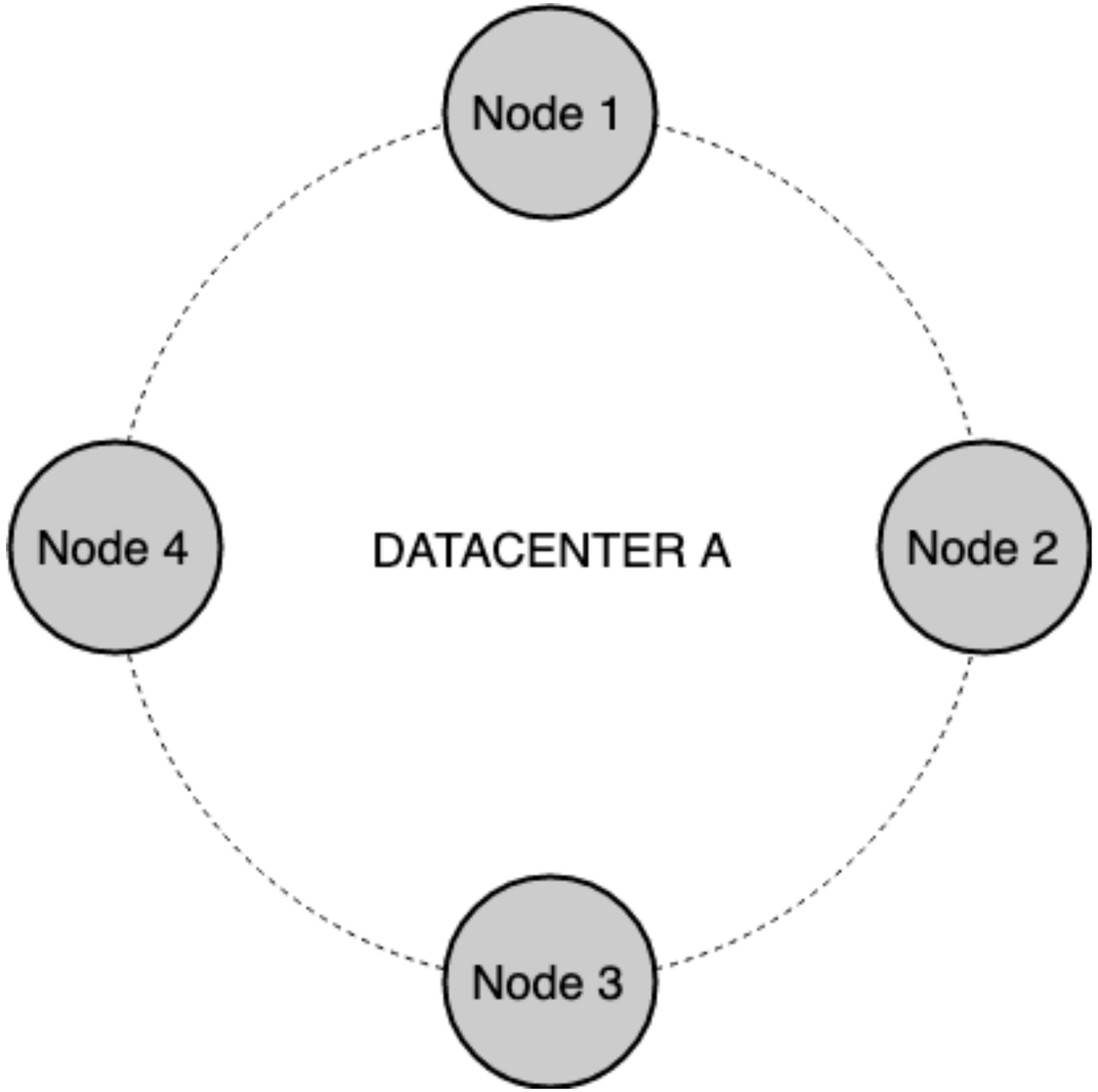
IMPROVEMENTS

IMPROVEMENTS

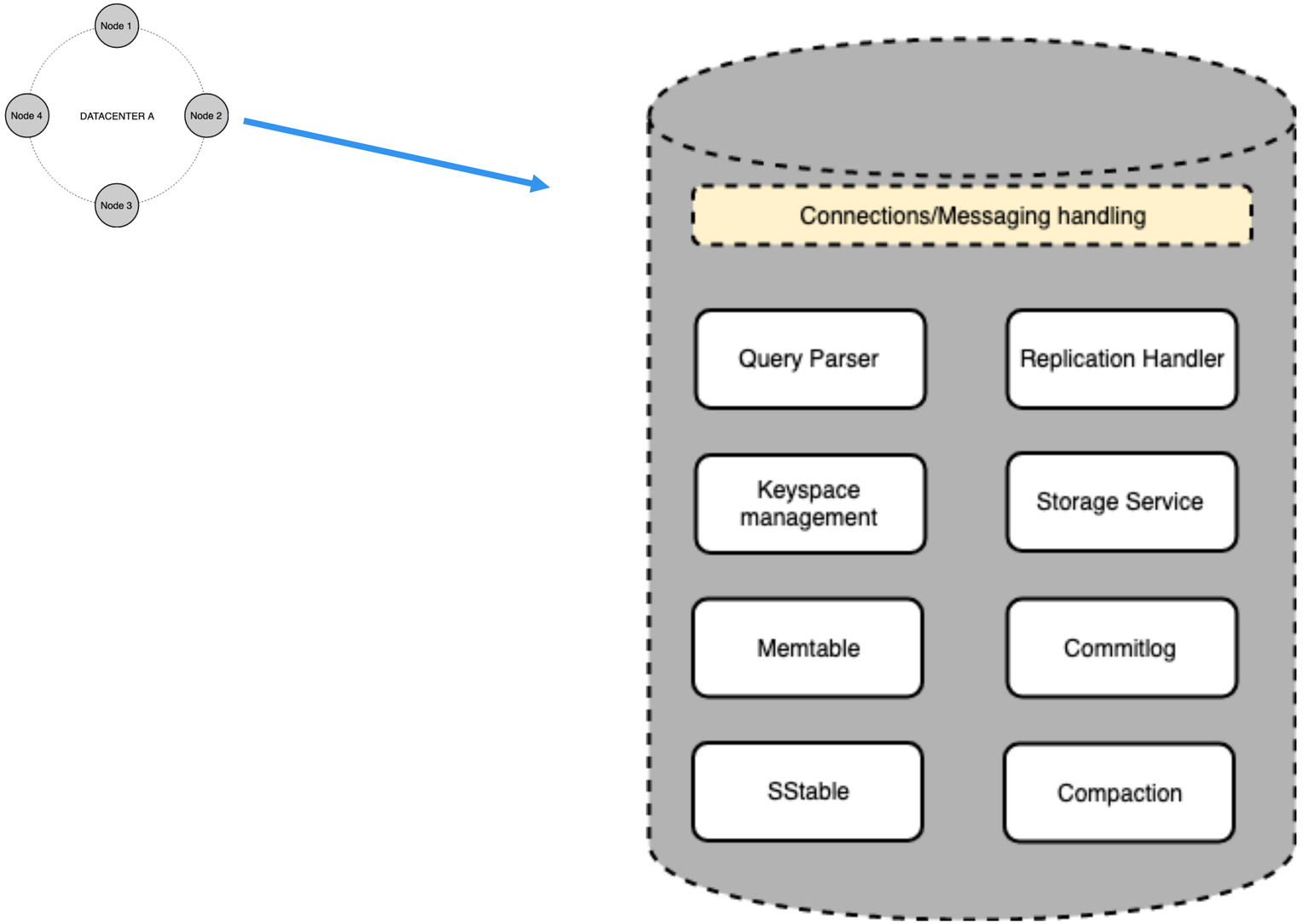
- Pluggable Storage Engine
- Global Data Partition
- Large Scale C* Cluster
- Gateway
- Manageability

PLUGGABLE STORAGE ENGINE

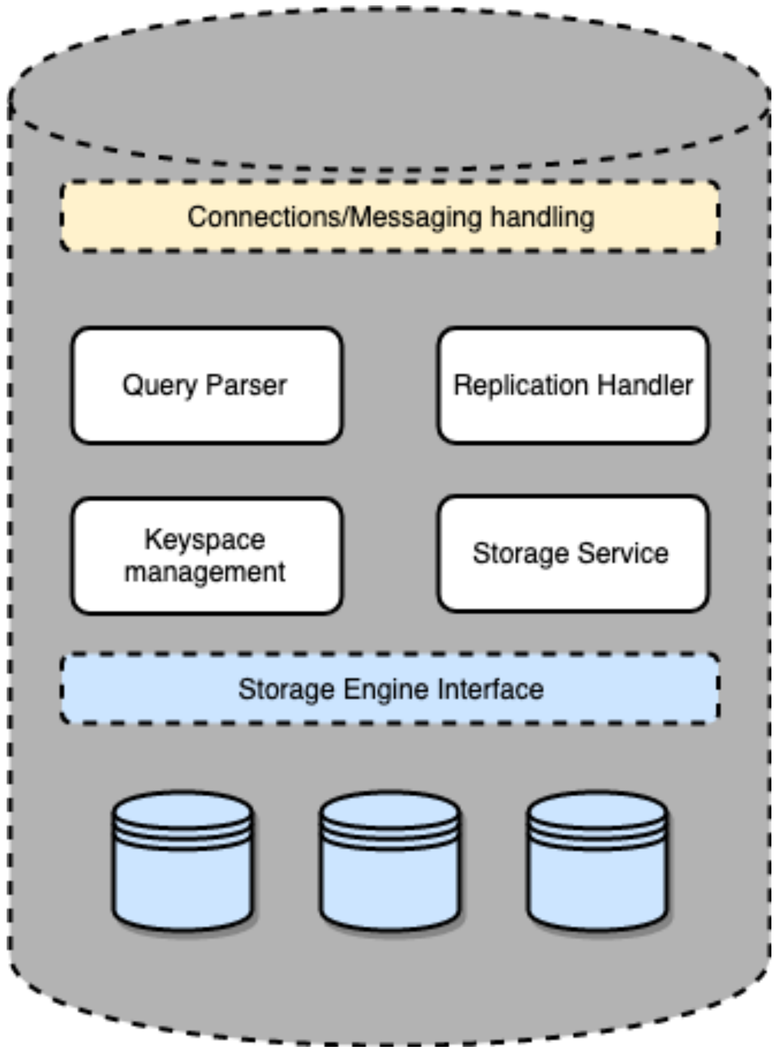
CASSANDRA



CASSANDRA SINGLE NODE



CASSANDRA STORAGE ENGINE LAYER



CASSANDRA-13474

Cassandra / CASSANDRA-13474

Cassandra pluggable storage engine

Details

Type:	New Feature	Status:	OPEN
Priority:	Normal	Resolution:	Unresolved
Component/s:	Legacy/Core	Fix Version/s:	None
Labels:	None		

Description

Instagram is working on a project to significantly reduce Cassandra's tail latency, by implementing a new storage engine on top of RocksDB, named Rocksandra.

We started a prototype of single column (key-value) use case, and then implemented a full design to support most of the data types and data models in Cassandra, as well as streaming.

After a year of development and testing, we have rolled out the Rocksandra project to our internal deployments, and observed 3-4X reduction on P99 read latency in general, even more than 10 times reduction for some use cases.

We published a blog post about the wins and the benchmark metrics on AWS environment. https://engineering.instagram.com/open-sourcing-a-10x-reduction-in-apache-cassandra-tail-latency-d64f86b43589

I think the biggest performance win comes from we get rid of most Java garbages created by current read/write path and compactions, which reduces the JVM overhead and makes the latency to be more predictable.

We are very excited about the potential performance gain. As the next step, I propose to make the Cassandra storage engine to be pluggable (like Mysql and MongoDB), and we are very interested in providing RocksDB as one storage option with more predictable performance, together with community.

Design doc for pluggable storage engine: https://docs.google.com/document/d/1suZlvhzgB6NIyBNpM9nxoHxz_Ri7qAm-UEO8v8AIFsc/edit

Issue Links

relates to CASSANDRA-13476 RocksDB based storage engine OPEN

Sub-Tasks

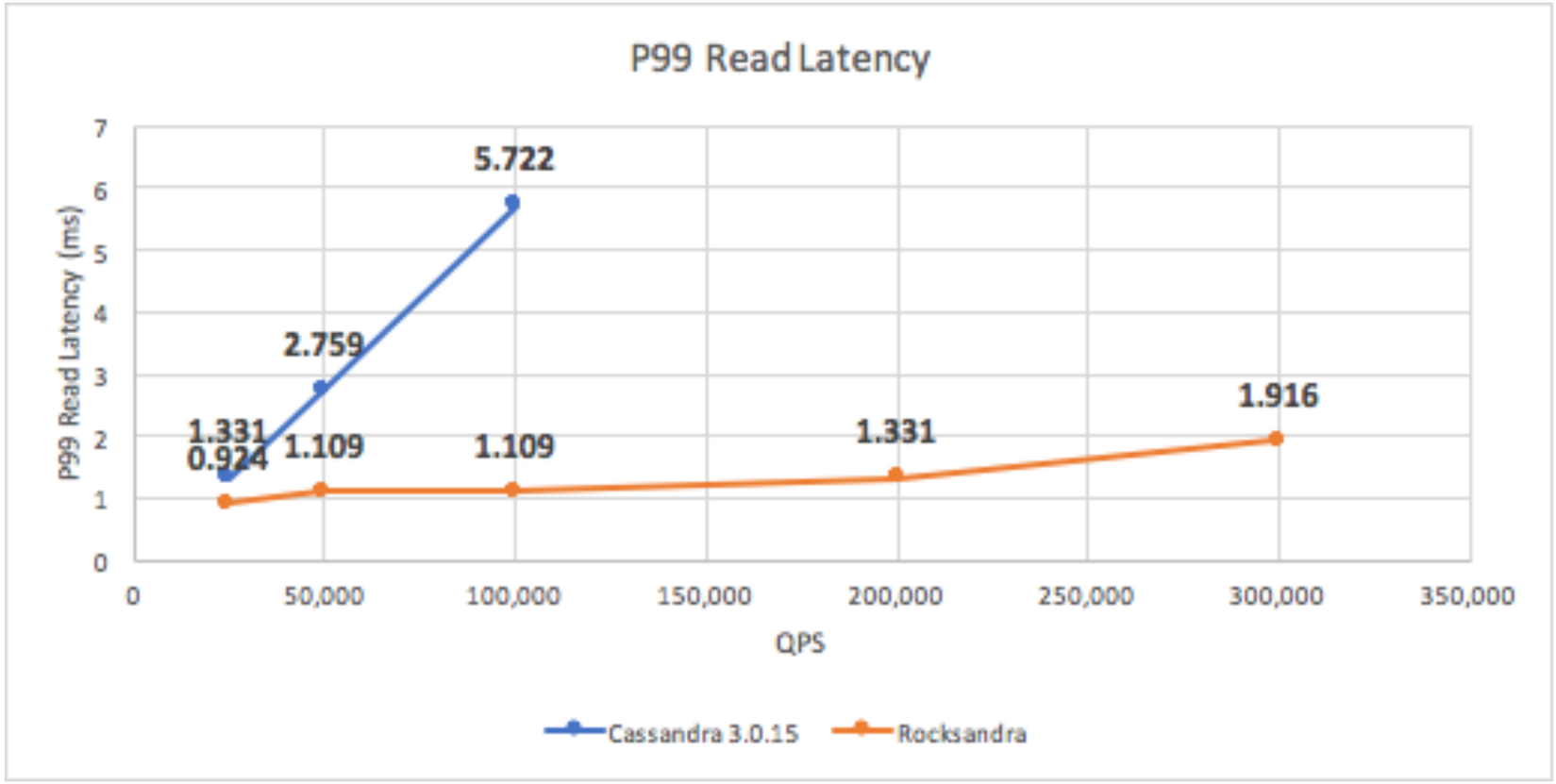
1.	Pluggable storage engine design	OPEN	Dikang Gu
2.	Refactor streaming	RESOLVED	Blake Eggleston
3.	Refactor repair	RESOLVED	Blake Eggleston
4.	Refactor read path	PATCH AVAIL...	Dikang Gu
5.	Refactor write path	RESOLVED	Blake Eggleston
6.	Refactor compaction	OPEN	*Unassigned*
7.	Refactor Keyspace/CFS operations	OPEN	*Unassigned*
8.	Refactor metrics	OPEN	*Unassigned*
9.	Refactor Indexes	OPEN	*Unassigned*
10.	Abstract storage engine API from Keyspace/CFS	OPEN	*Unassigned*
11.	Refactor Schema/Metadata	OPEN	*Unassigned*
12.	Refactor commitlog	OPEN	*Unassigned*

NEW HIGH PERFORMANCE ENGINE

Rocksandra

https://github.com/Instagram/cassandra/tree/rocks_3.0

ROCKSANDRA

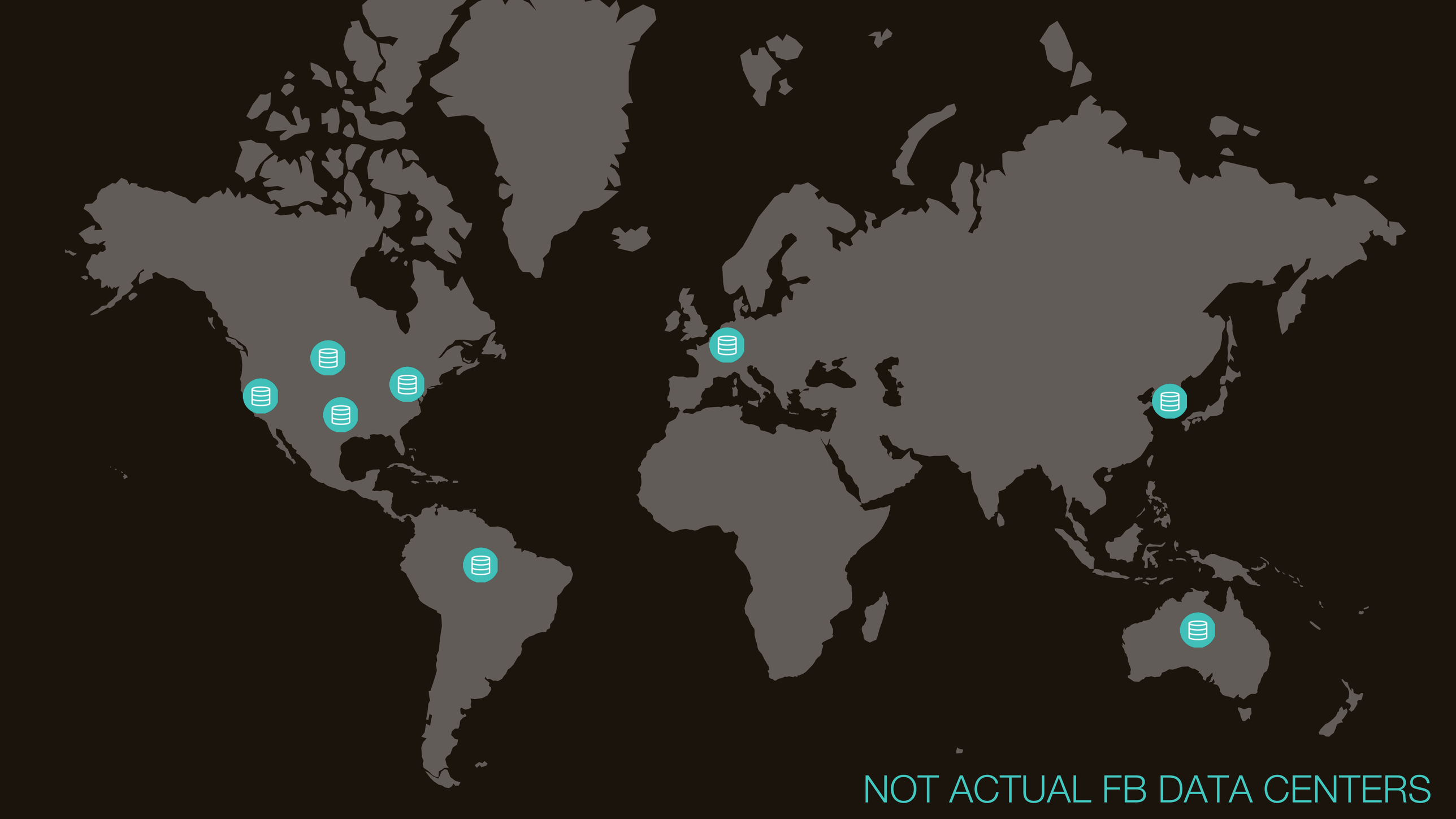


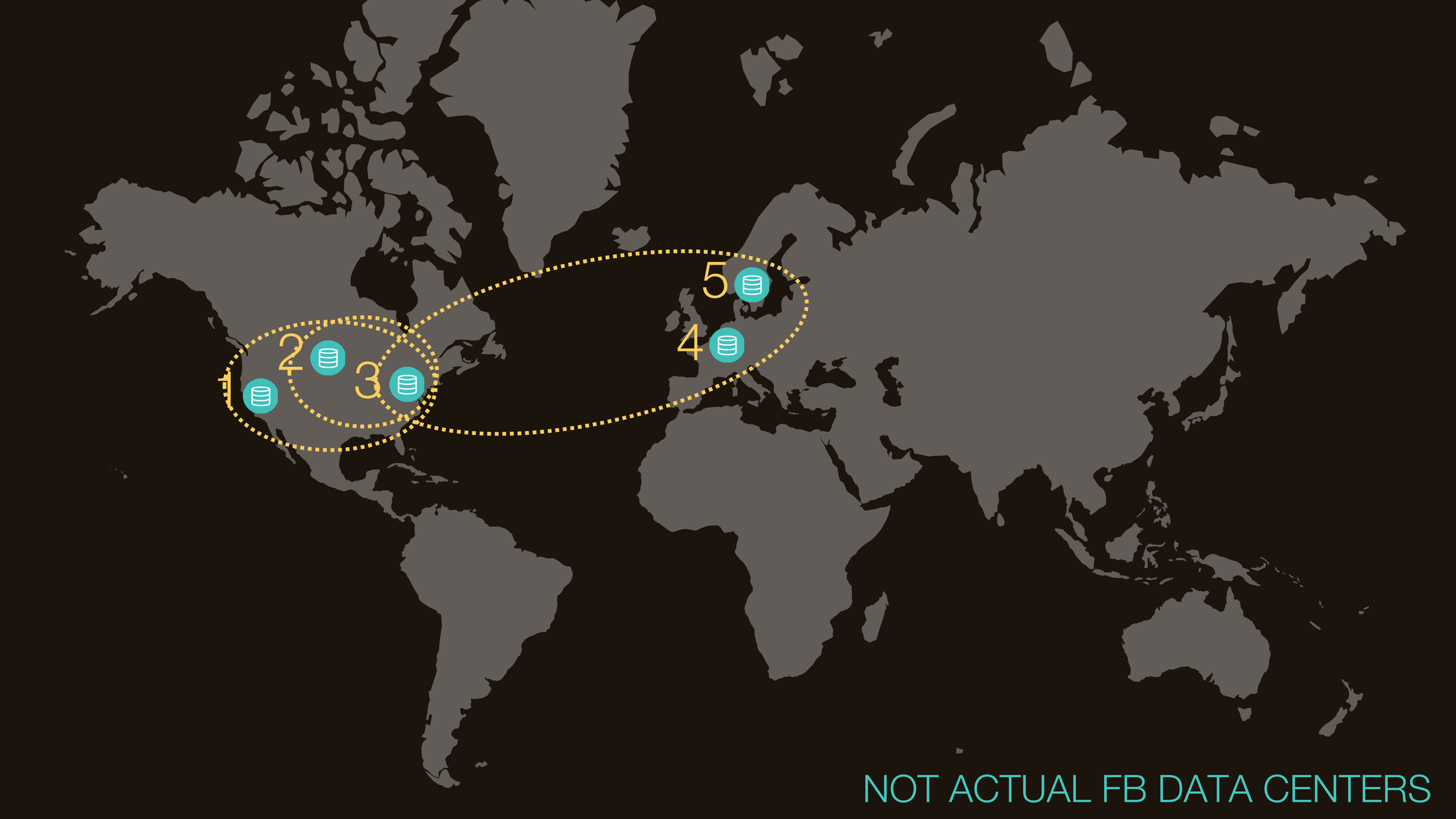
https://instagram-engineering.com/open-sourcing-a-10x-reduction-in-apache-cassandra-tail-latency-d64f86b43589

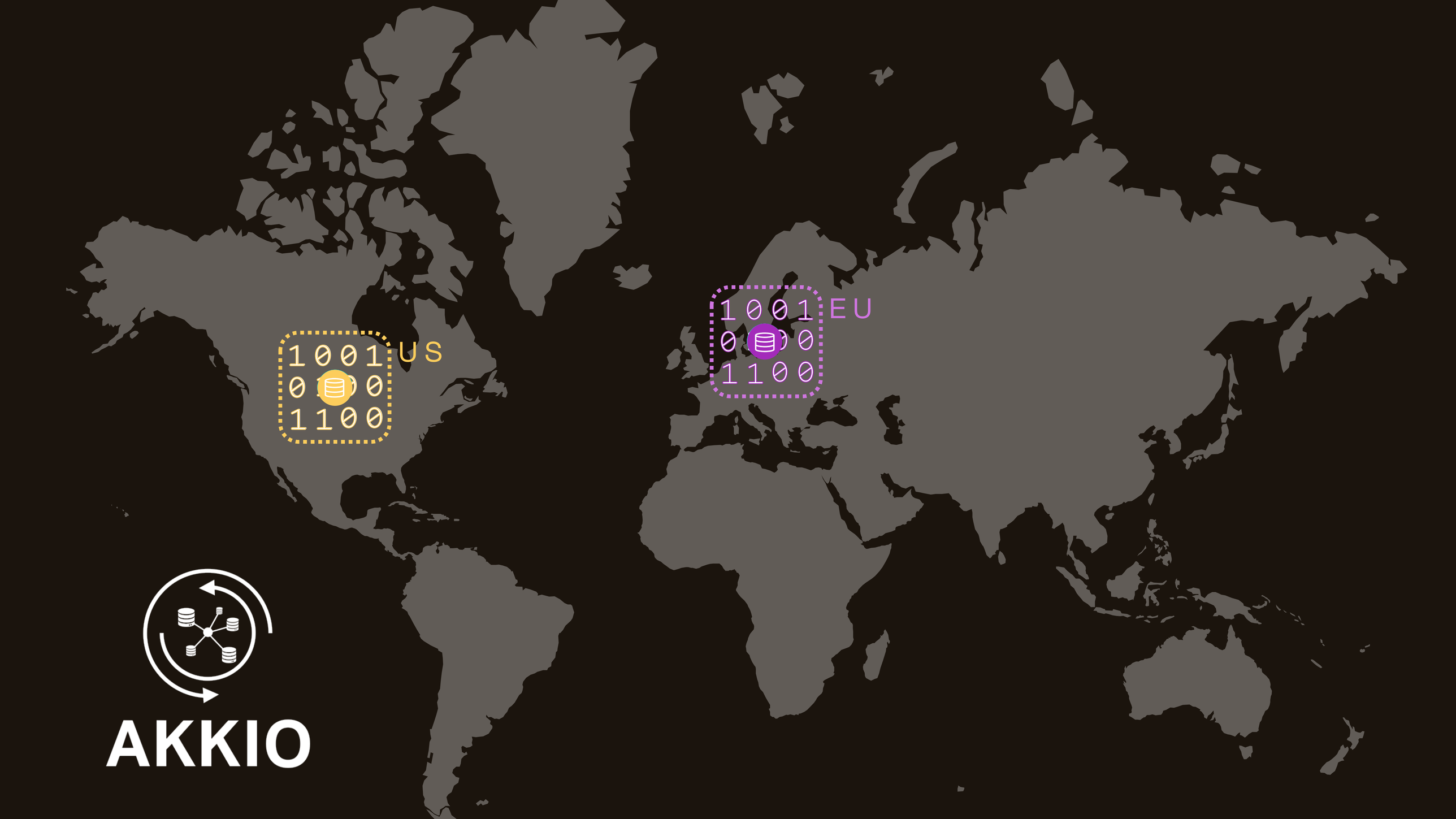
Talk: “Rocksandra Update”

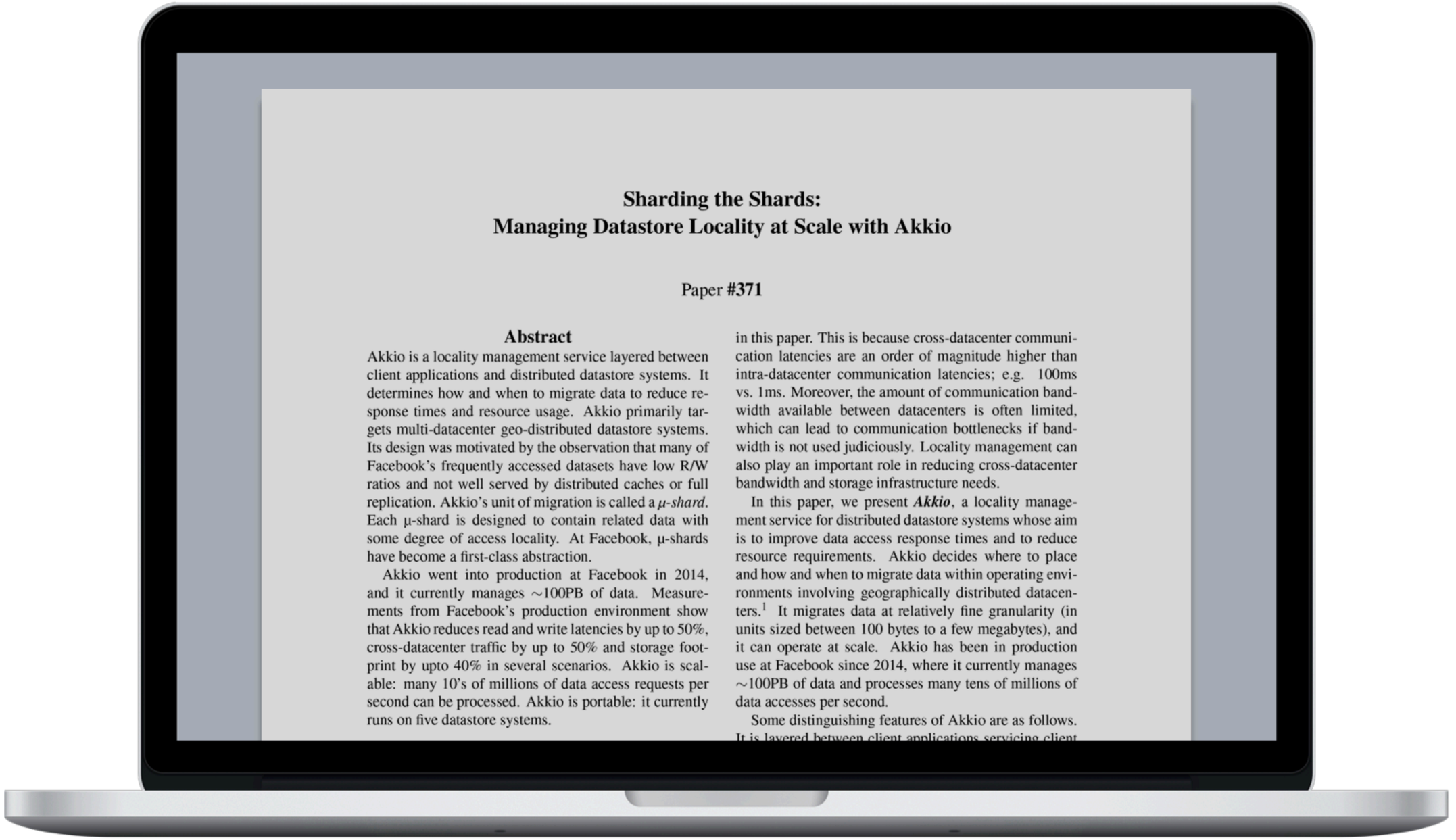
By Pengchao Wang @ Instagram

GLOBAL DATA PARTITION


NOT ACTUAL FB DATA CENTERS


1
2
3
4
5
NOT ACTUAL FB DATA CENTERS


1001
1100
US
1001
1100
EU
AKKIO



Sharding the Shards: Managing Datastore Locality at Scale with Akkio

Paper **#371**

Abstract

Akkio is a locality management service layered between client applications and distributed datastore systems. It determines how and when to migrate data to reduce response times and resource usage. Akkio primarily targets multi-datacenter geo-distributed datastore systems. Its design was motivated by the observation that many of Facebook's frequently accessed datasets have low R/W ratios and not well served by distributed caches or full replication. Akkio's unit of migration is called a *μ-shard*. Each μ-shard is designed to contain related data with some degree of access locality. At Facebook, μ-shards have become a first-class abstraction.

Akkio went into production at Facebook in 2014, and it currently manages ∼100PB of data. Measurements from Facebook's production environment show that Akkio reduces read and write latencies by up to 50%, cross-datacenter traffic by up to 50% and storage footprint by upto 40% in several scenarios. Akkio is scalable: many 10's of millions of data access requests per second can be processed. Akkio is portable: it currently runs on five datastore systems.

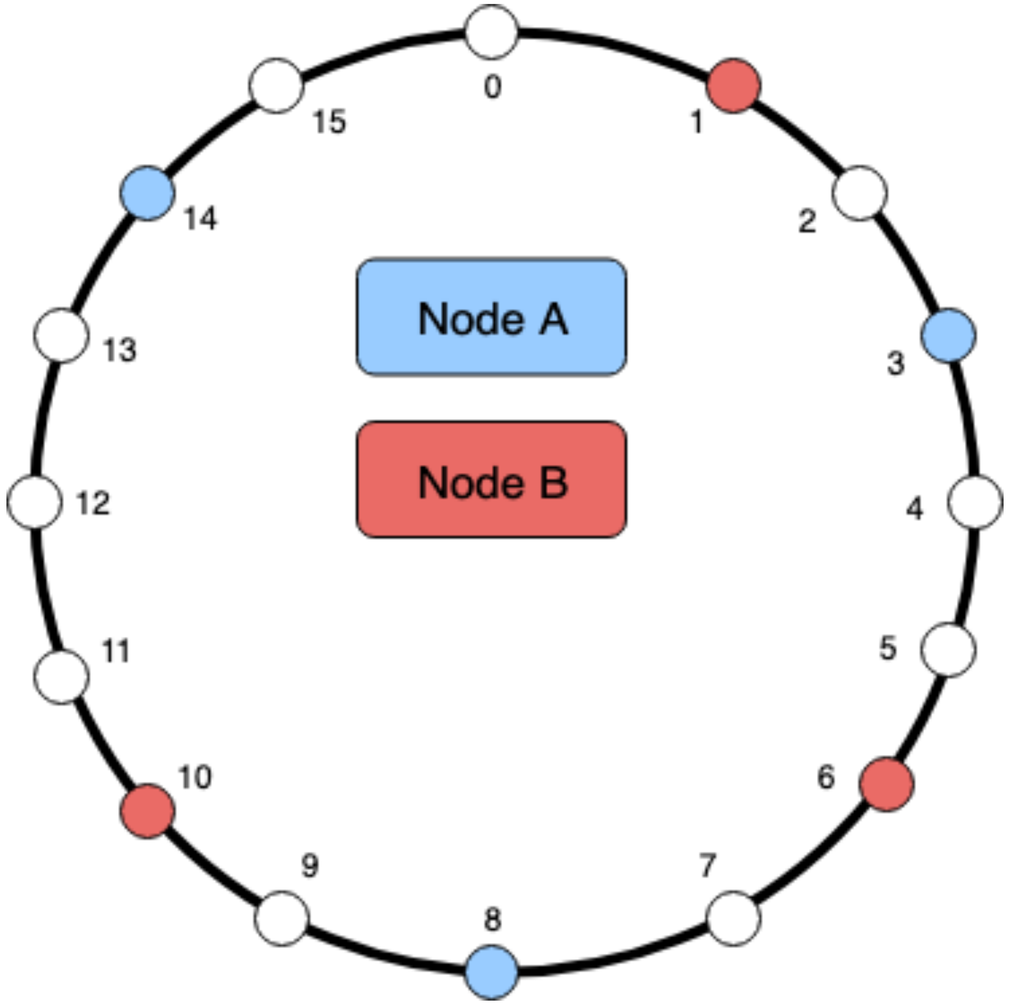
in this paper. This is because cross-datacenter communication latencies are an order of magnitude higher than intra-datacenter communication latencies; e.g. 100ms vs. 1ms. Moreover, the amount of communication bandwidth available between datacenters is often limited, which can lead to communication bottlenecks if bandwidth is not used judiciously. Locality management can also play an important role in reducing cross-datacenter bandwidth and storage infrastructure needs.

In this paper, we present ***Akkio***, a locality management service for distributed datastore systems whose aim is to improve data access response times and to reduce resource requirements. Akkio decides where to place and how and when to migrate data within operating environments involving geographically distributed datacenters.[1] It migrates data at relatively fine granularity (in units sized between 100 bytes to a few megabytes), and it can operate at scale. Akkio has been in production use at Facebook since 2014, where it currently manages ∼100PB of data and processes many tens of millions of data accesses per second.

Some distinguishing features of Akkio are as follows. It is layered between client applications servicing client

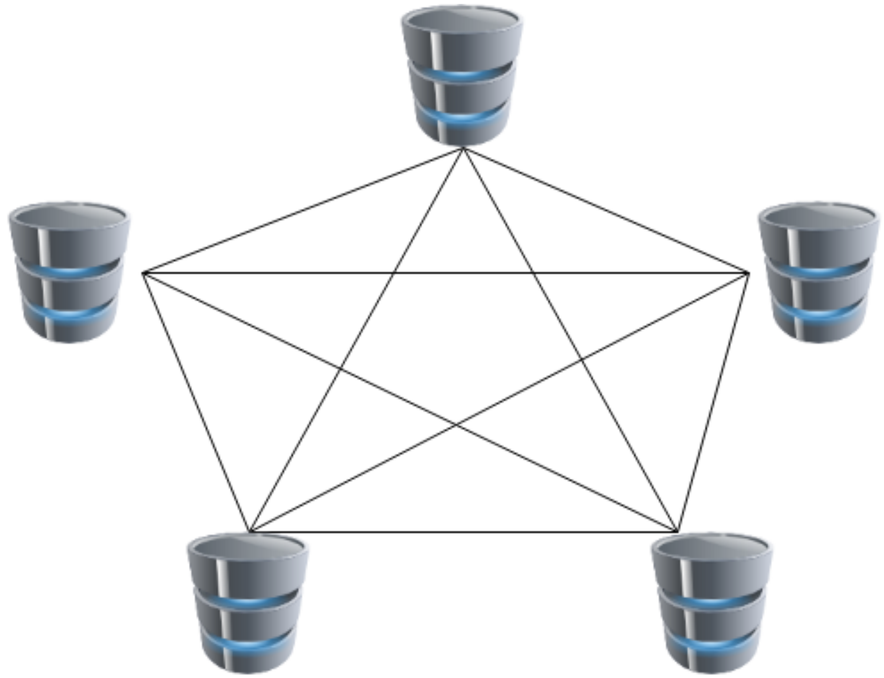
LARGE SCALE C* CLUSTER

CONSISTENT HASHING

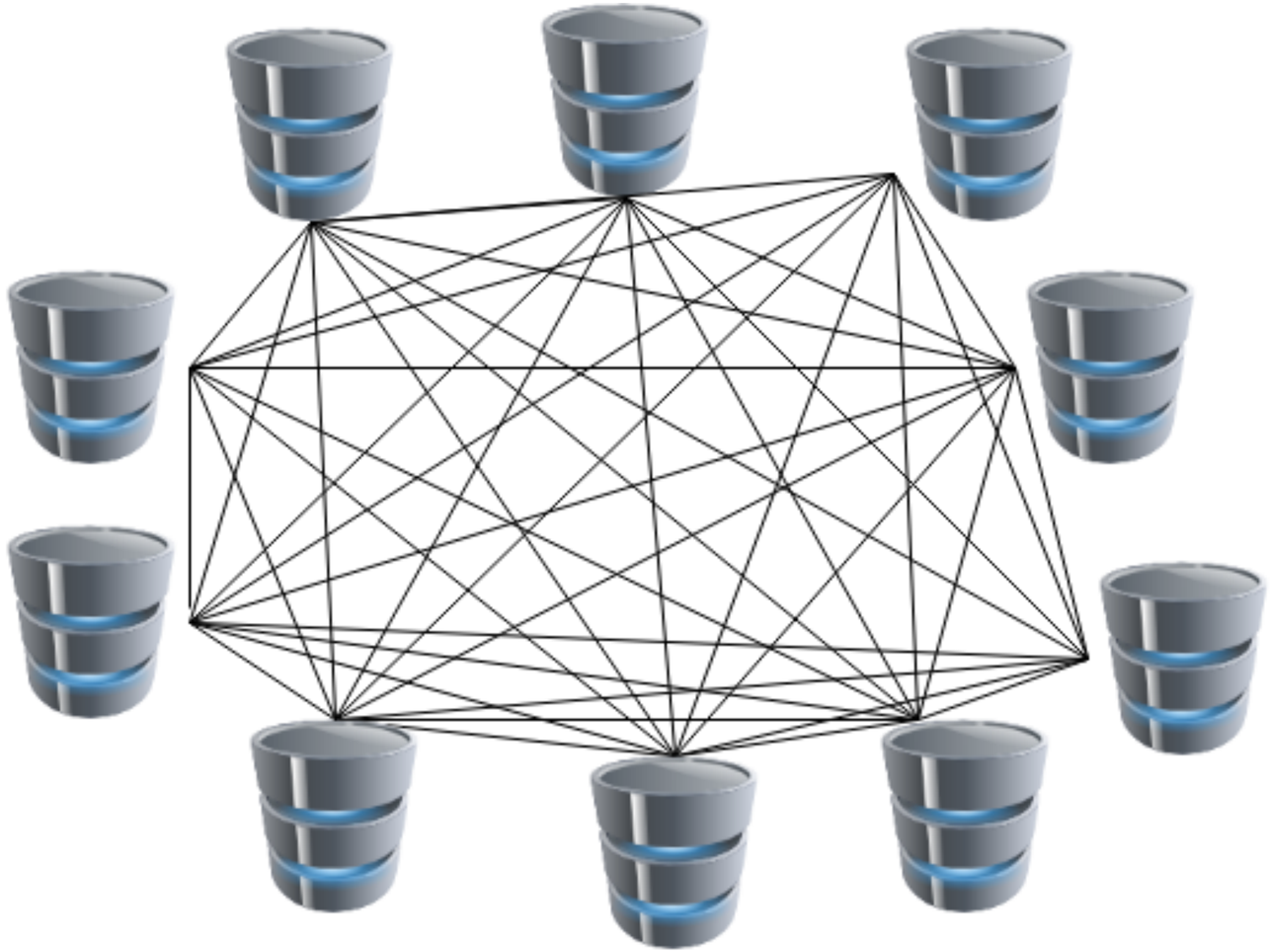


GOSSIP

- Peer-to-Peer communication
- Exchange state information about themselves and other nodes
  - Membership
  - Ownership
  - Healthiness



LARGE CASSANDRA CLUSTER



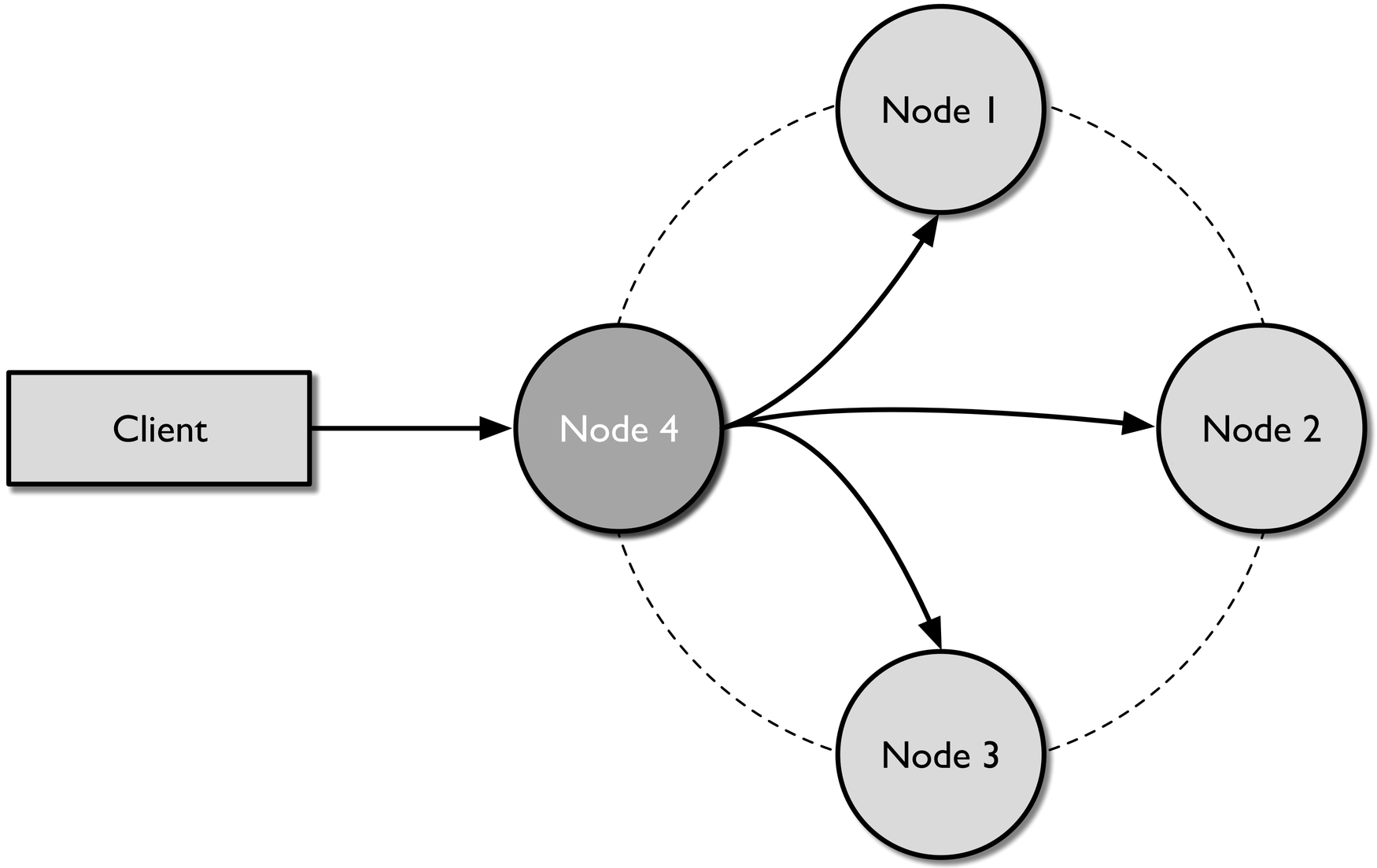
GOSSIP IMPROVEMENTS

- Avoid Updating unchanged gossip state (CASSANDRA-15097)
- Endpoints no longer owning tokens are not removed for vnode (CASSANDRA-15098)
- Node restart causes unnecessary token metadata update (CASSANDRA-15133)
- RemoveNode takes long time and blocks gossip stage (CASSANDRA-15141)

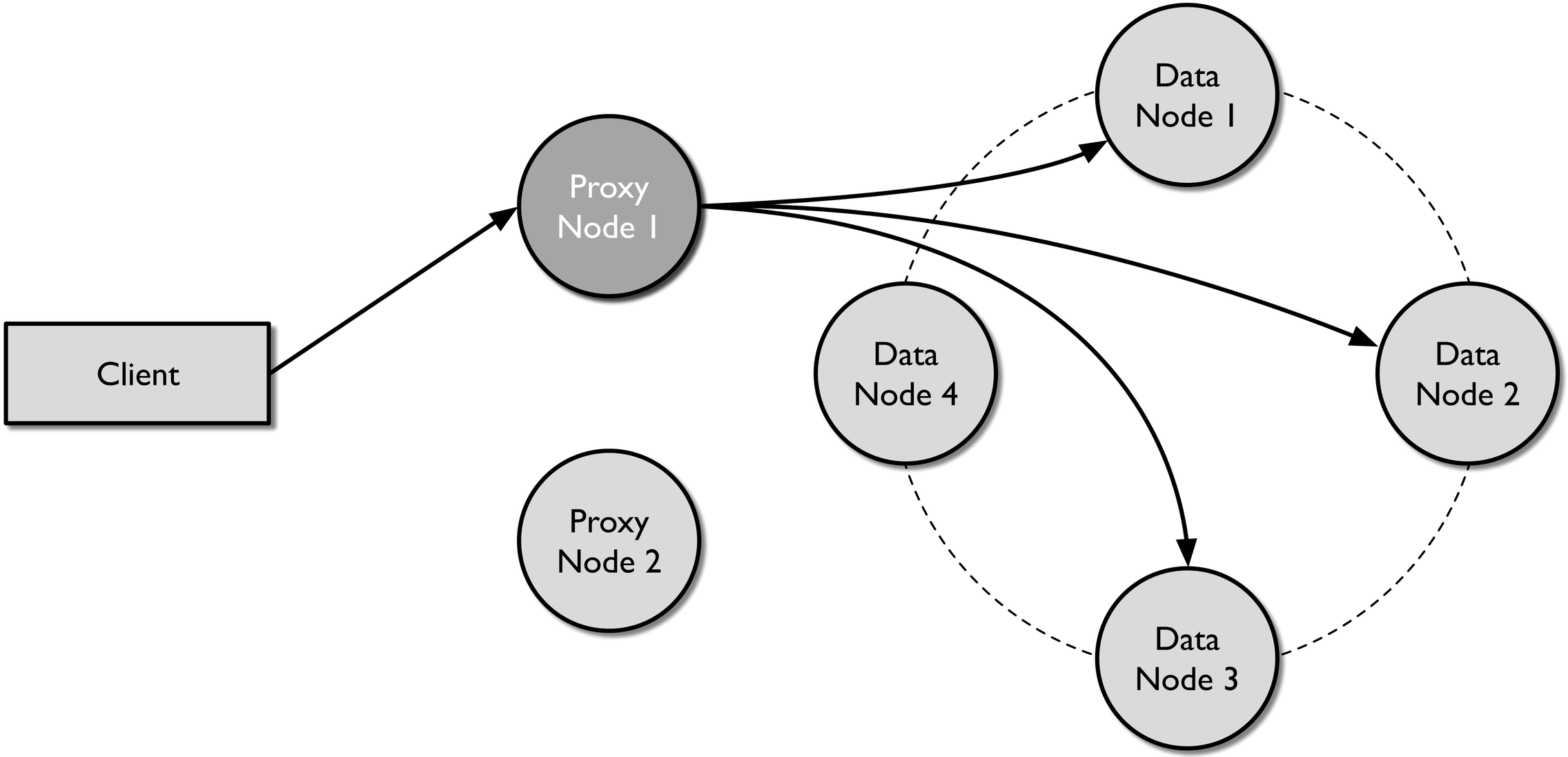
10x Cassandra server start-up!

GATEWAY

QUERY PATH



QUERY PATH WITH PROXY NODES



PRESENT CASSANDRA INFRASTRUCTURE

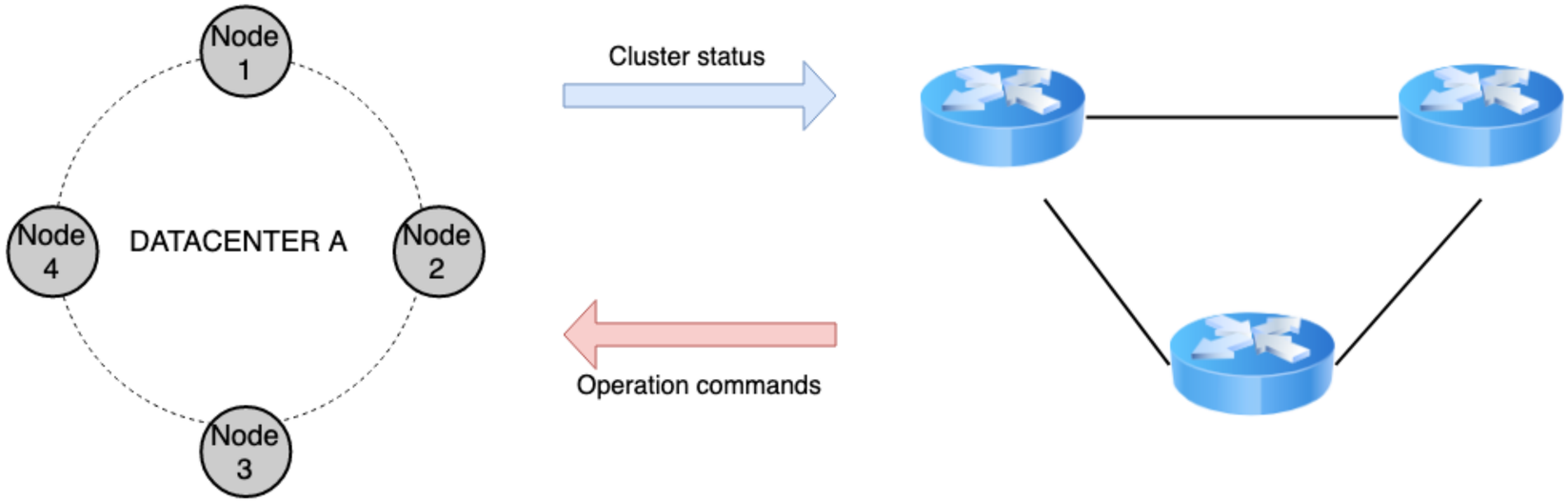
TRAFFIC SHADOWING

Talk: “Cassandra Traffic Management at Instagram”

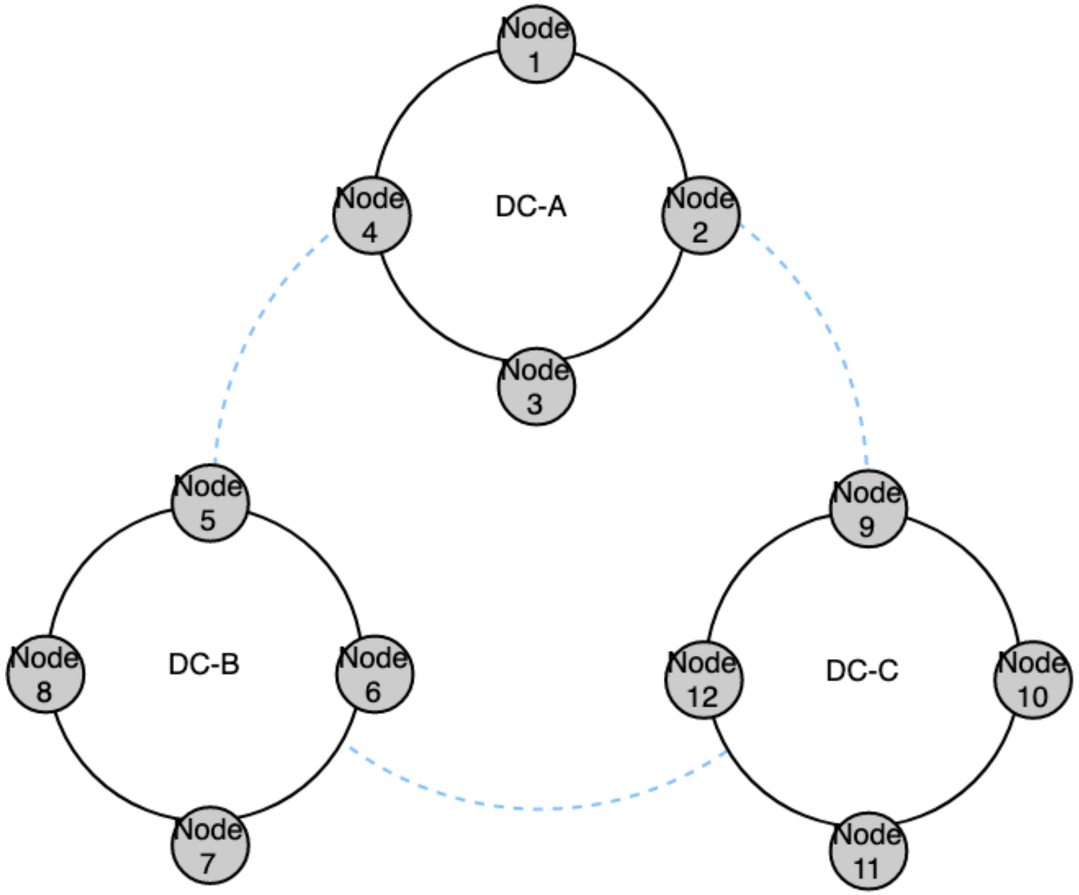
By Michaël Figuière @ Instagram

MANAGEABILITY

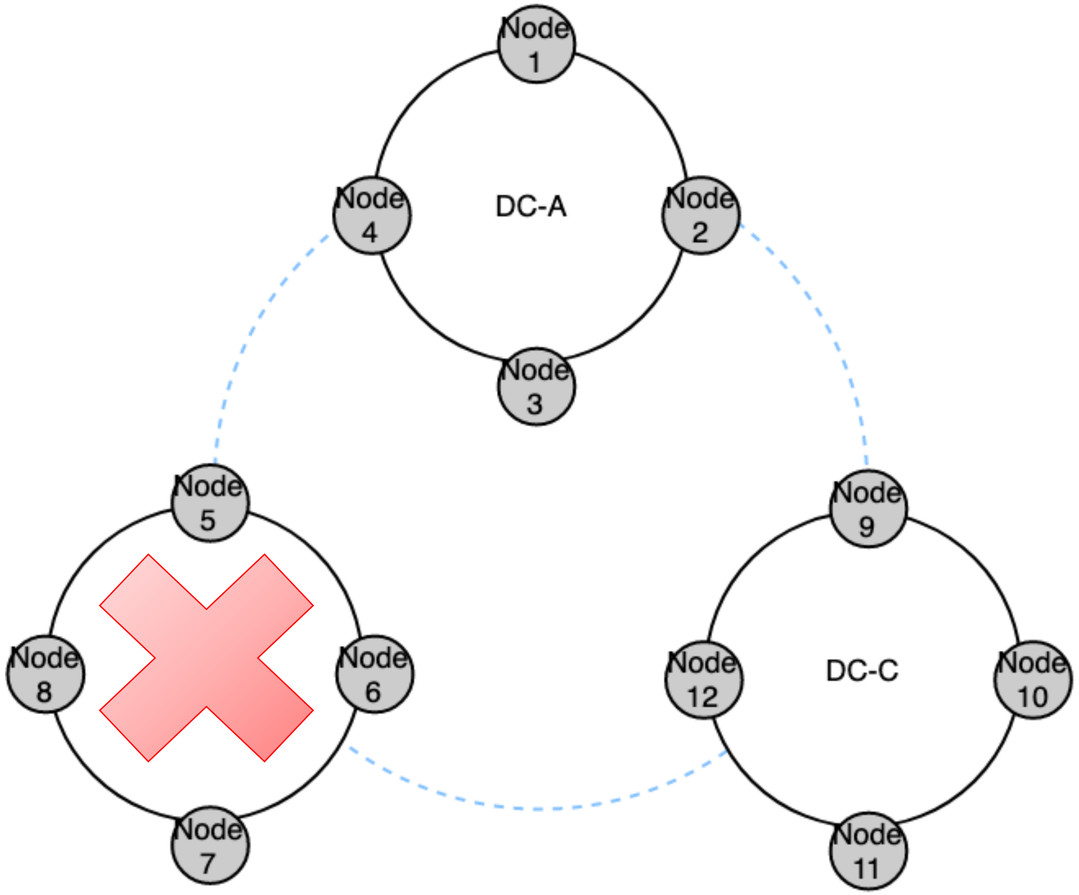
CONTROL PLANE



DISASTER READINESS



DISASTER READINESS



RECAP

Improvements

- Pluggable Storage Engine
- Global Data Partition
- Large scale Cassandra cluster
- Gateway
- Manageability

CHALLENGES

STRONG CONSISTENT MEMBERSHIP

PROBLEMS

- Tokens are allocated with no coordination, caused unevenly balanced clusters or even duplicated tokens
- Violate consistency during token movements
- Block running operations in parallel

CASSANDRA-9667

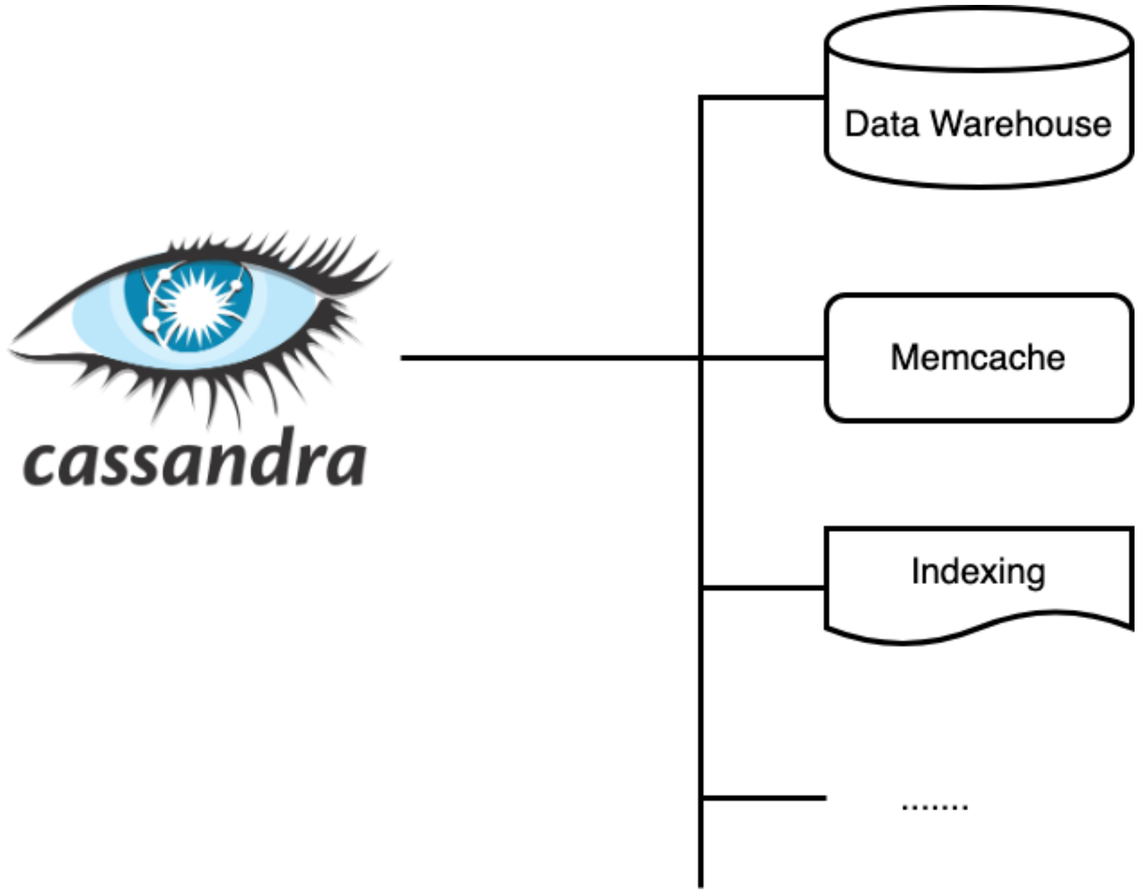
Strongly consistent membership and ownership

Talk: “Cassandra Consistent Token Management”

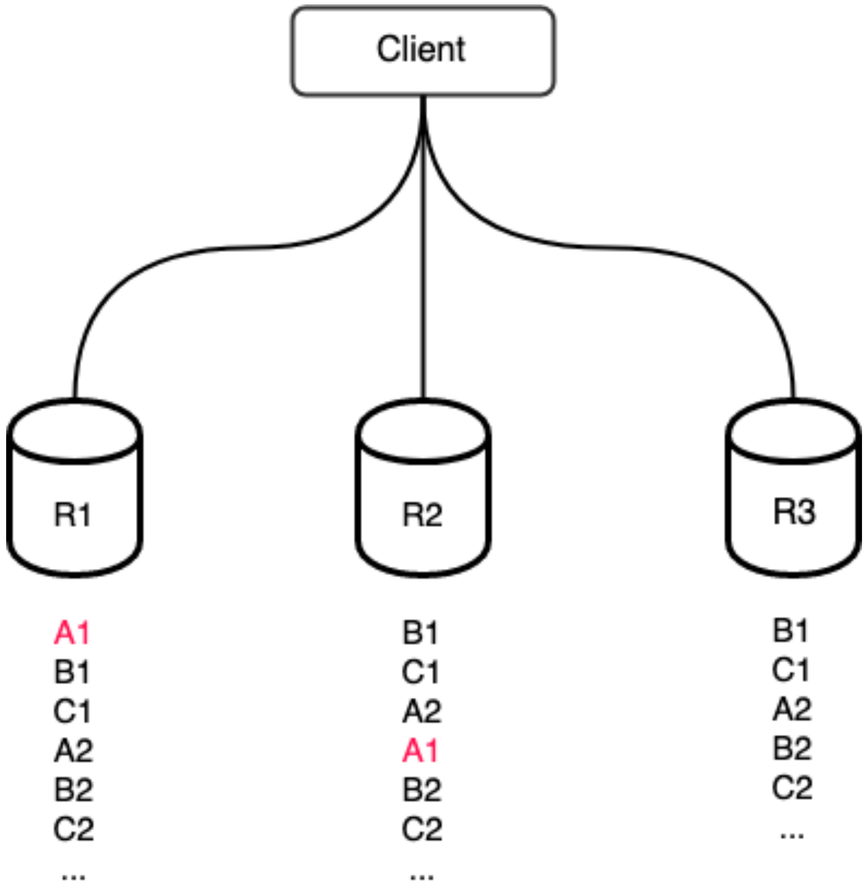
By Jay Zhuang @ Instagram

GLOBAL DATA ORDERING

DATA PIPELINES



INCONSISTENT REPLICAS



RECAP

- Cassandra usage at Instagram
- Improvements
  - Pluggable Storage Engine
  - Global Data Partition
  - Large scale Cassandra cluster
  - Gateway
  - Manageability
- Challenges
  - Storage consistent membership
  - Global data ordering