



DataStax Enterprise Architecture

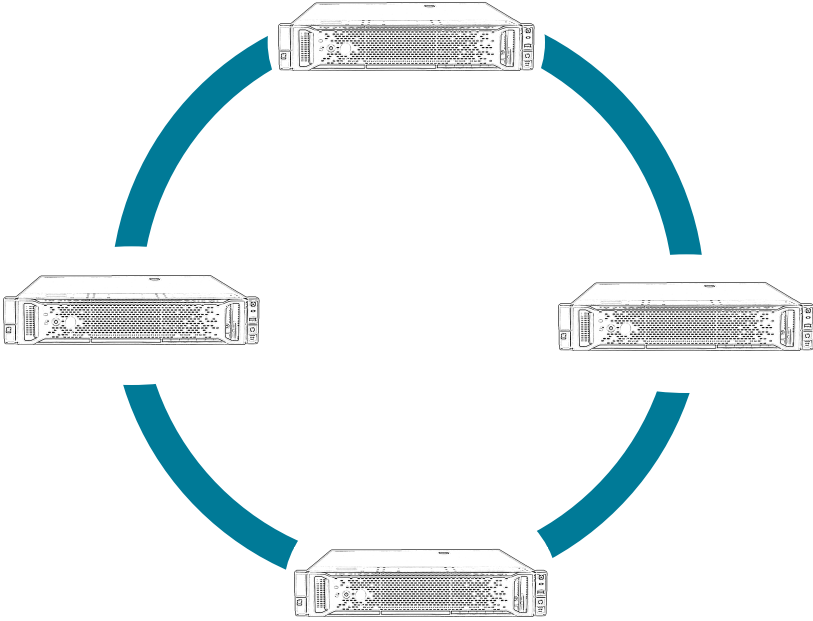
Rob Murphy & Aaron Regis

14th June 2017

Agenda

1	Topology and Data Structure
2	Request Handling
3	Lab1: Cassandra Access and Cassandra Stress

Design Goals and Objectives



- Continuously Available
- Master Less
- Fully Distributed
- Shared-Nothing Architecture
- Build In Replication
- Linear Scalability
- Scale out

Architecture

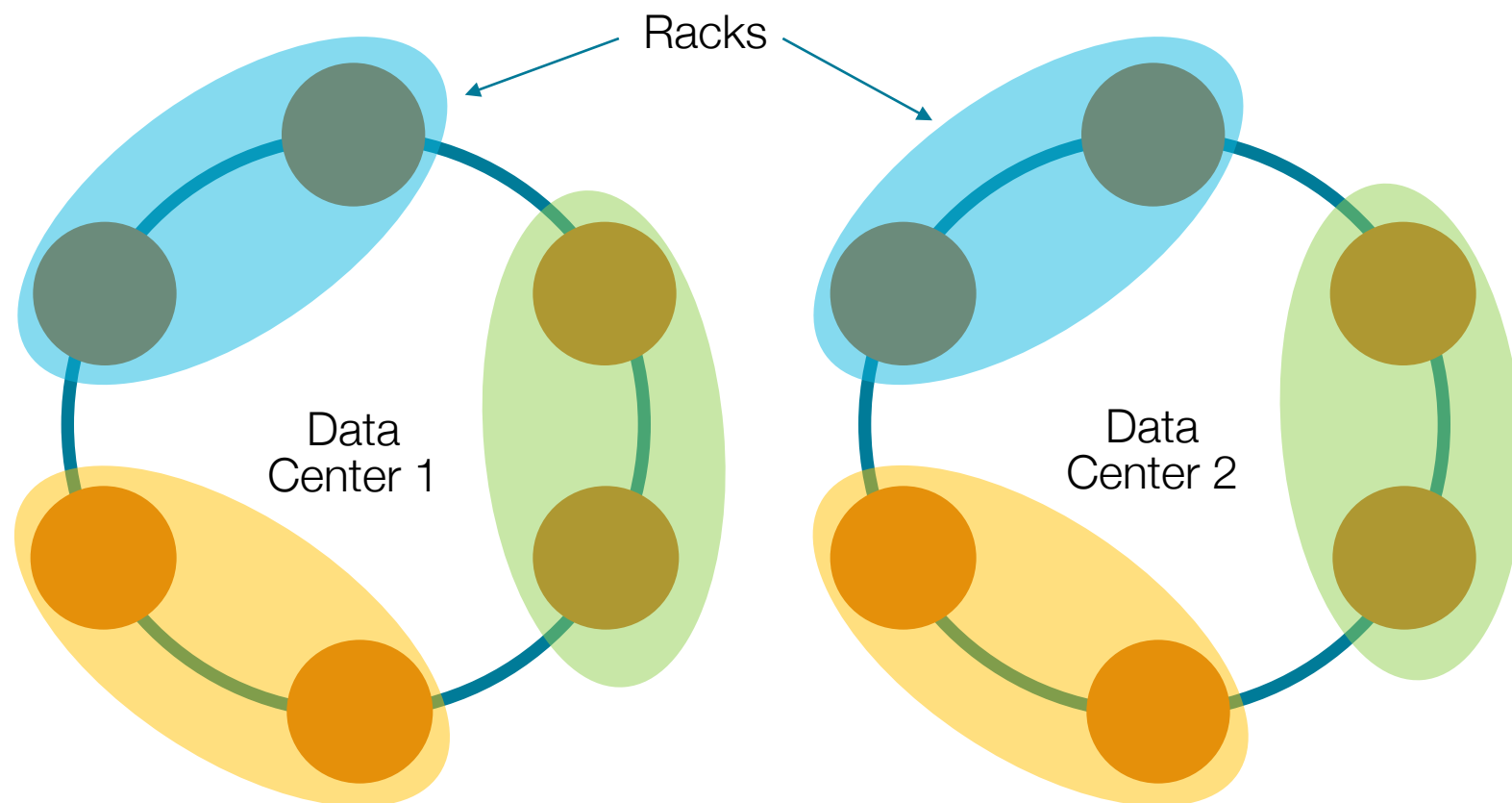
Apache Cassandra™ Architecture

- Cluster layer
 - Amazon DynamoDB paper
 - masterless architecture
- Data-store layer
 - Google Big Table paper
 - Columns/columns family

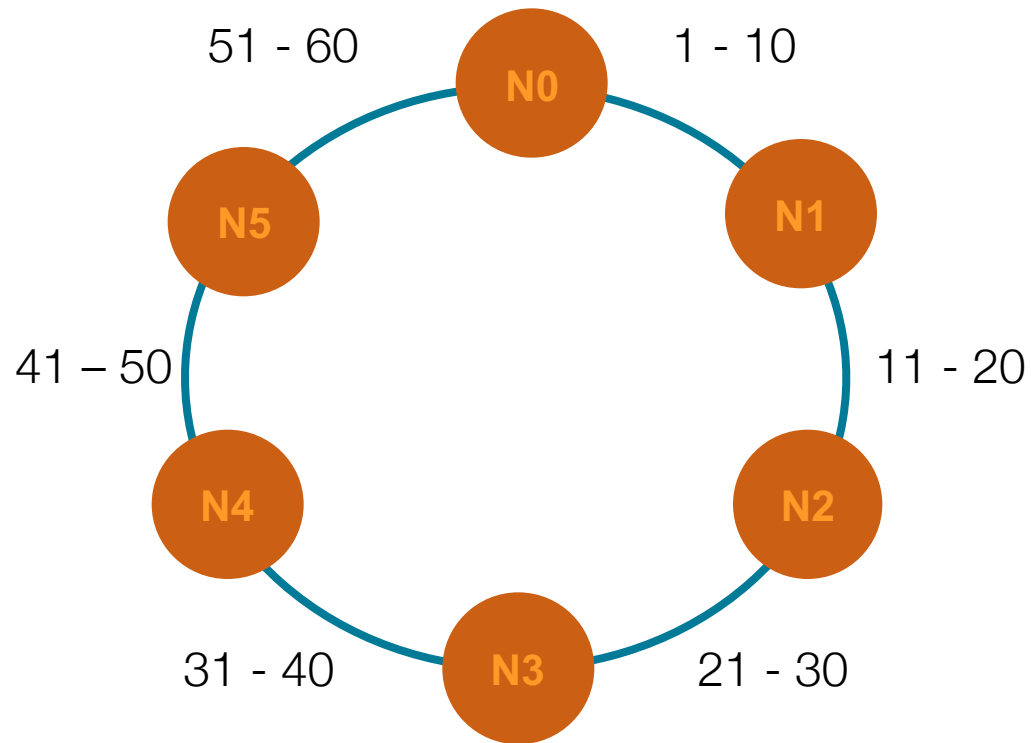


- All nodes are peers
 - Including seed nodes
 - No master
 - Discovery through gossip
- Built-in replication
 - Simplify your architecture!

Cluster



Token Ranges



Token Range : $2^{63} - 2^{63}$

Example with **Replication Factor 3**

N3 will own data for tokens 1 – 30

Token Range : 1-10, owned by N1,N2,N3

Token Range : 11-20, owned by N2,N3,N4

Token Range : 21-30, owned by N3,N4,N5

- Primary key
 - Partition key
 - Clustering columns
- Partitioner
 - Generates unique hash from partition key
- Replication strategy
 - Token hash determines starting point
 - Determines replica placement

Cassandra Query Language

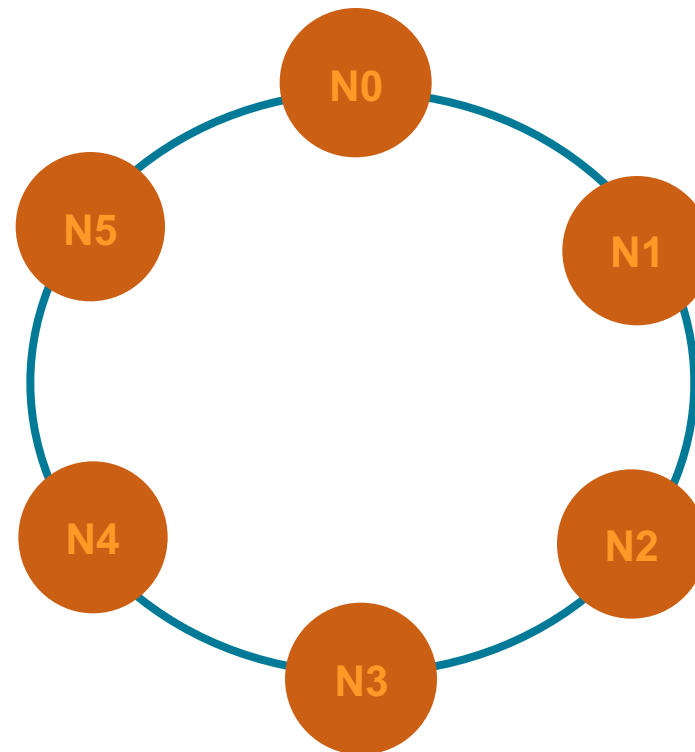
```
CREATE KEYSPACE negib WITH replication =  
{'class': 'NetworkTopologyStrategy', 'DC1': '3'} ← Replication Factor  
AND durable_writes = true;
```

```
CREATE TABLE negib.sales (  
    name text,  
    time int,  
    item text,  
    price double,  
    PRIMARY KEY (name, time) ← Partition Key  
) WITH CLUSTERING ORDER BY (time DESC)
```

```
SELECT * FROM sales where name='Thomas' OR name='Günther' AND time >=20160401;
```

Data Distribution

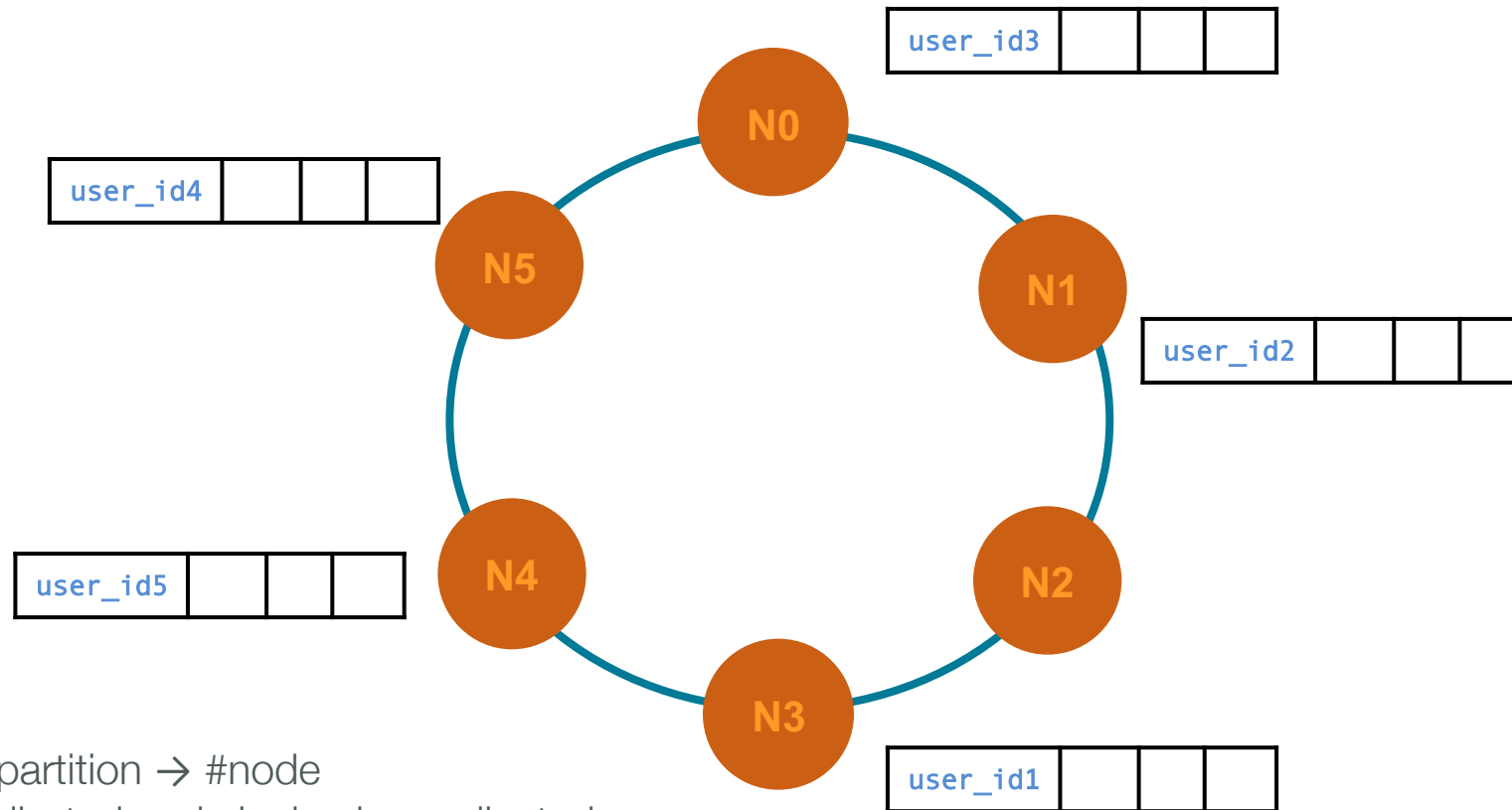
user_id1			
user_id2			
user_id3			
user_id4			
user_id5			



Token = hash of #partition → #node

Data is evenly distributed and clock wise replicated

Data Distribution



Token = hash of #partition \rightarrow #node

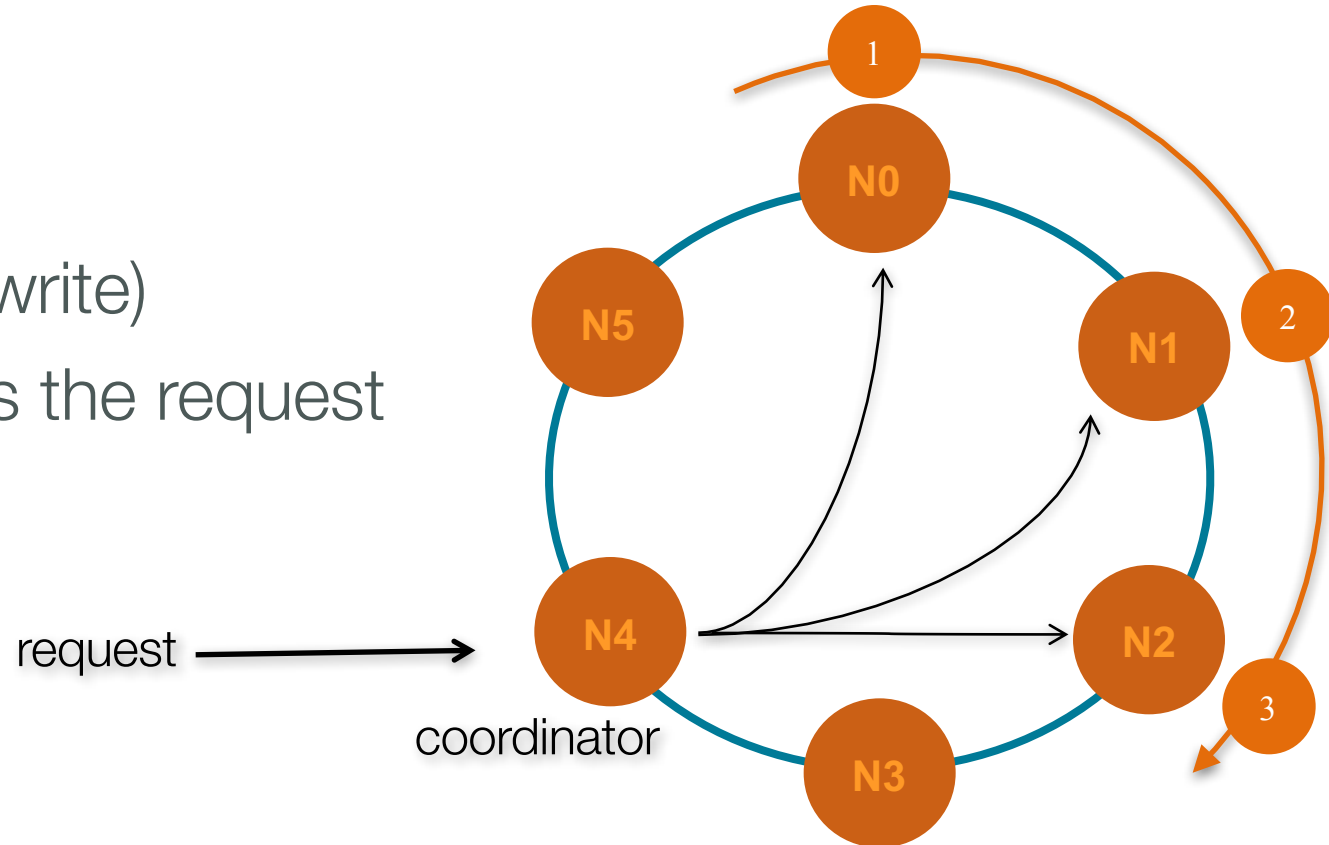
Data is evenly distributed and clock wise replicated

Lab 1 : Accessing the cluster

Read and write request handling

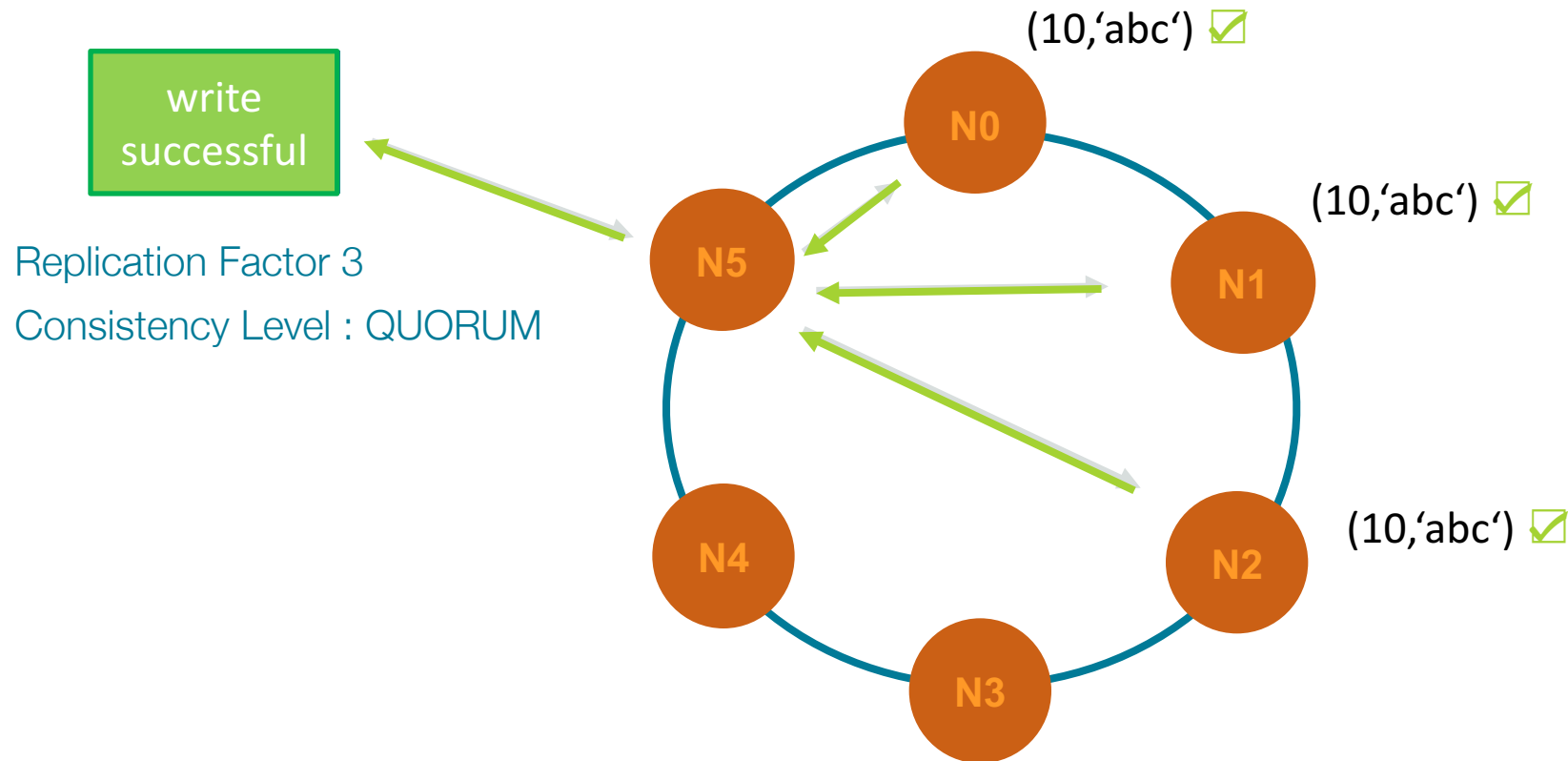
Coordinator node

Incoming requests (read/write)
Coordinator node handles the request



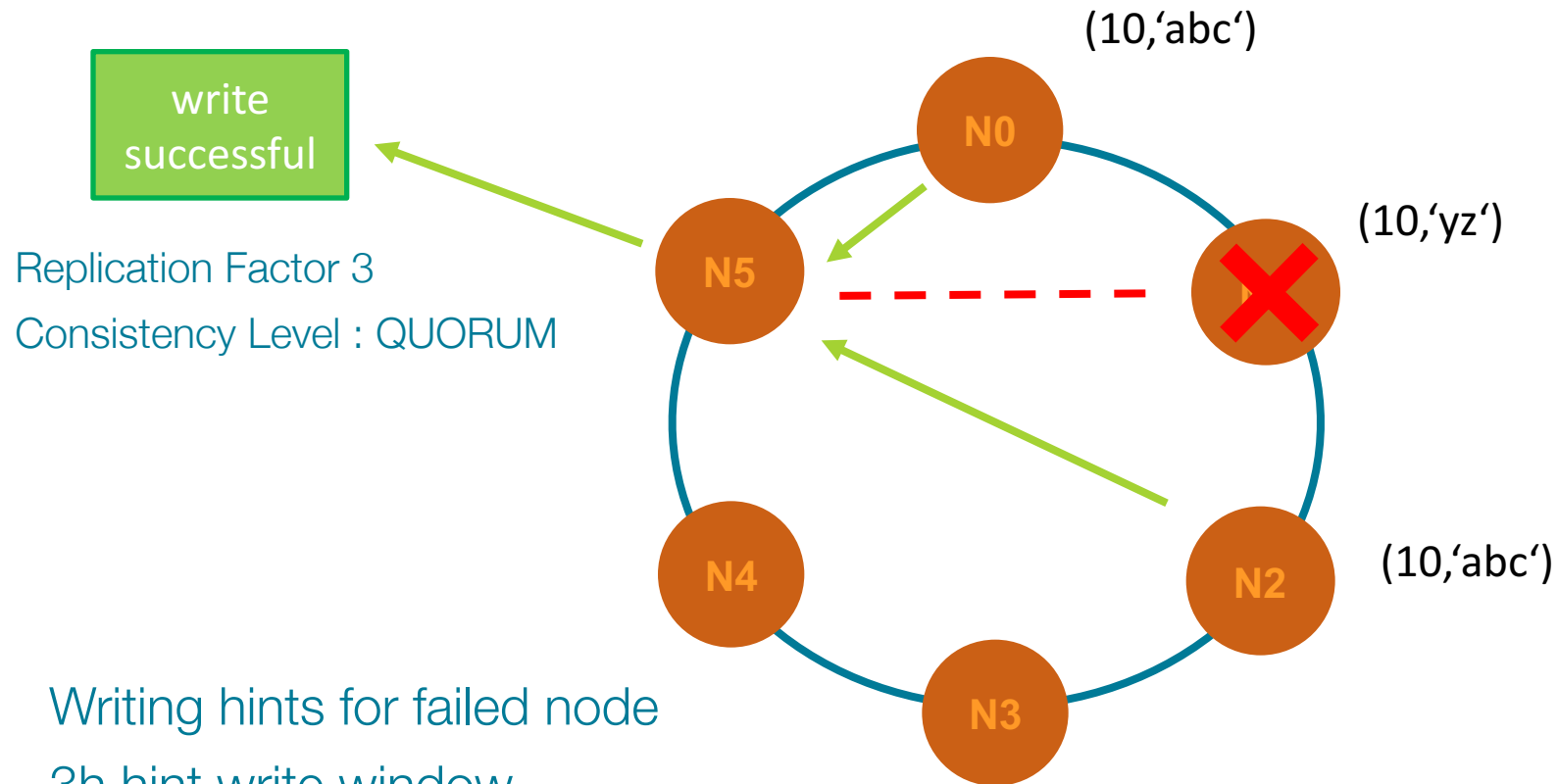
Every node can be coordinator → masterless

Write request handling



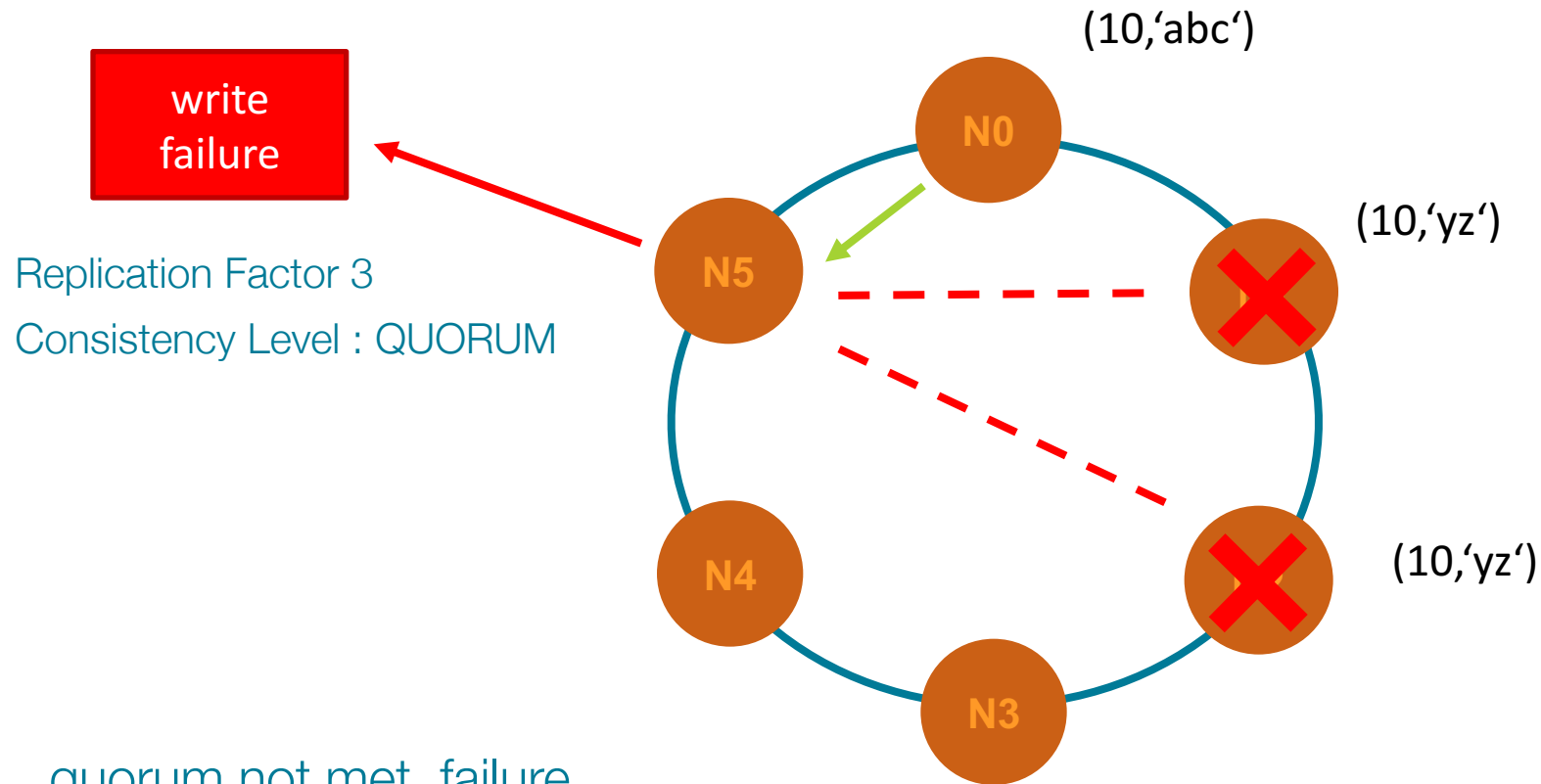
- Background vs. Foreground Read Repair
 - Compare digests
 - If any mismatch
 - re-request to same nodes (full data set)
 - compare full data sets, send update
 - block until out-of-date replicas respond
 - Return merged data set to the client
- Consistency Level
 - one, quorum, all
 - local vs. cluster wide

Write request handling



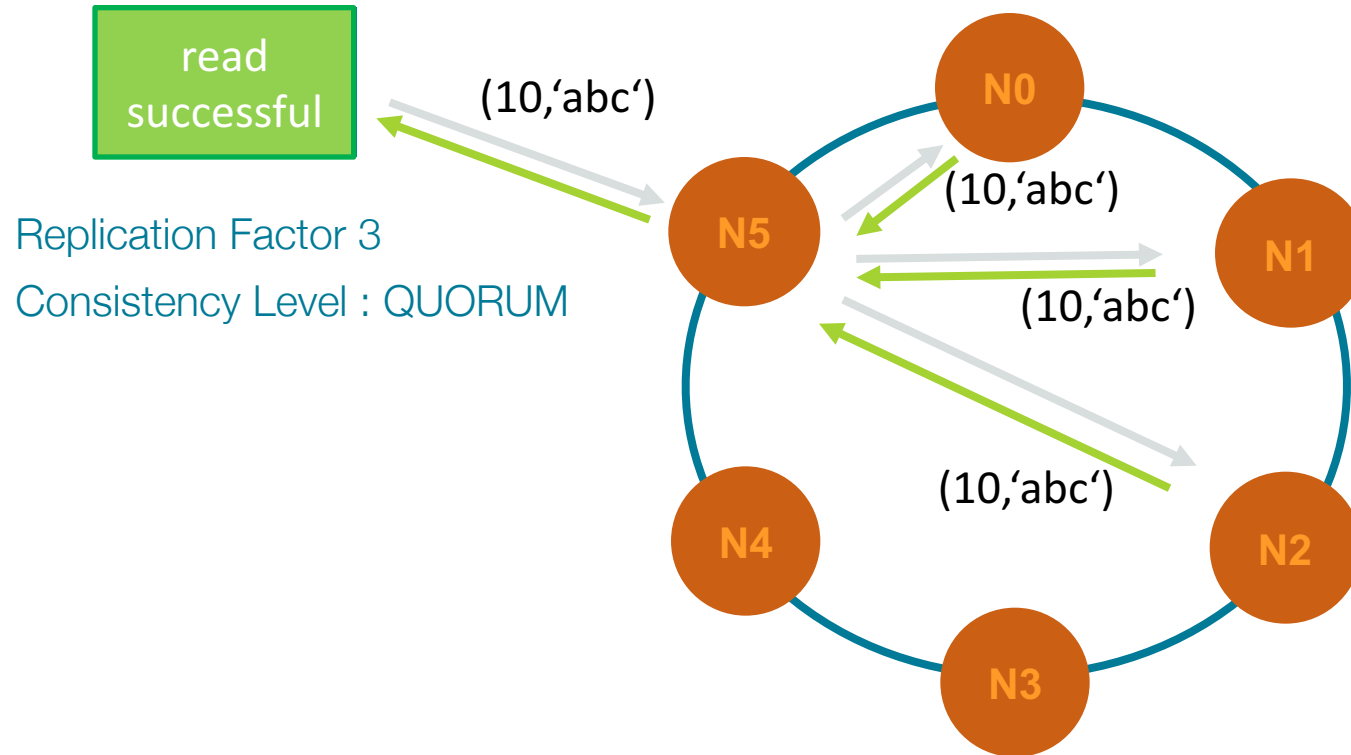
- Writing hints for failed node
- 3h hint write window
- CL: LOCAL_ALL will fail

Write request handling

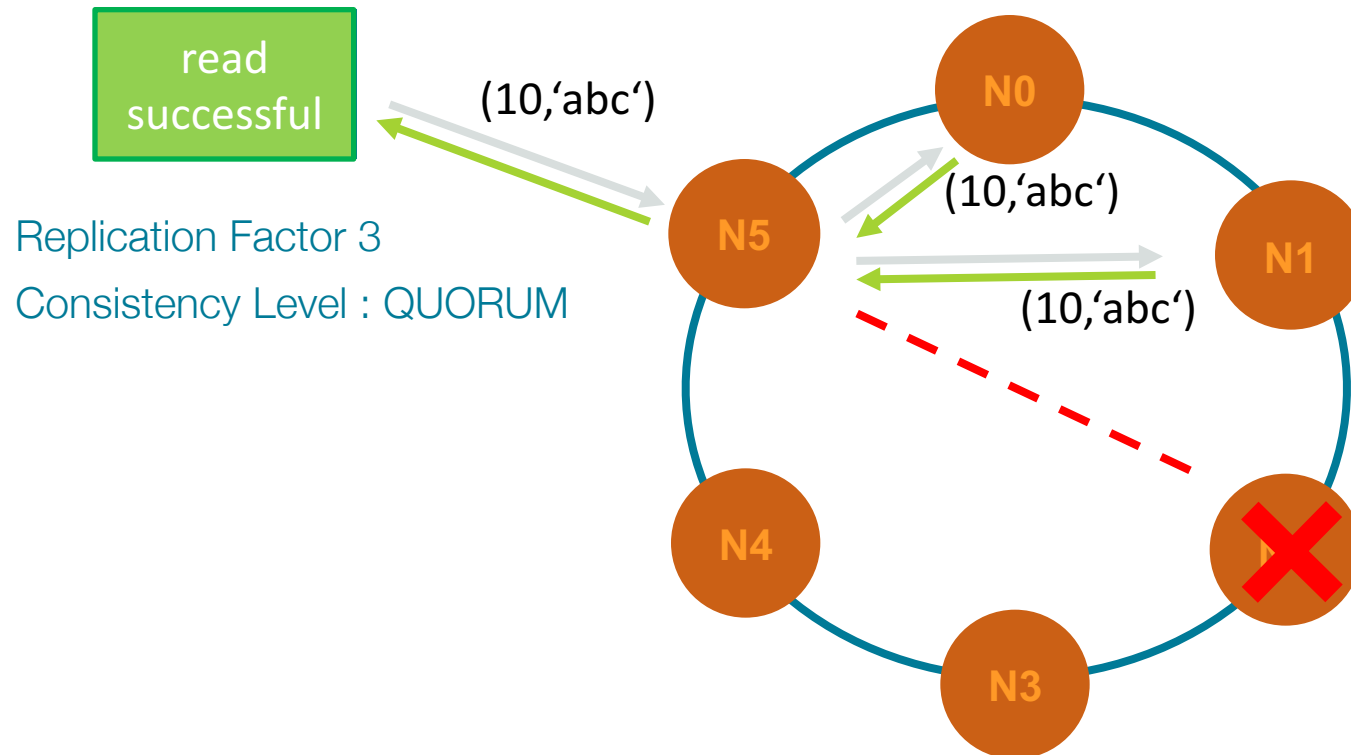


- quorum not met, failure
- CL: LOCAL_ONE will succeed

Read request handling

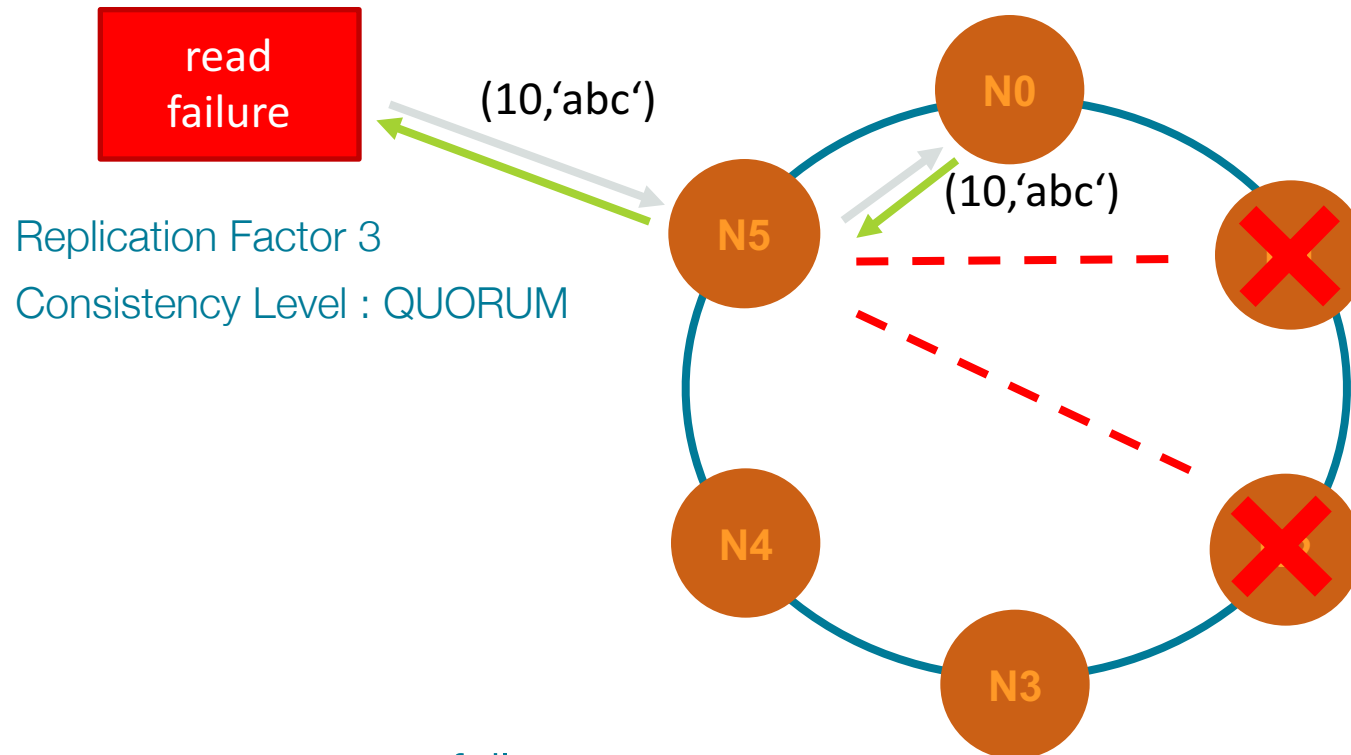


Read request handling



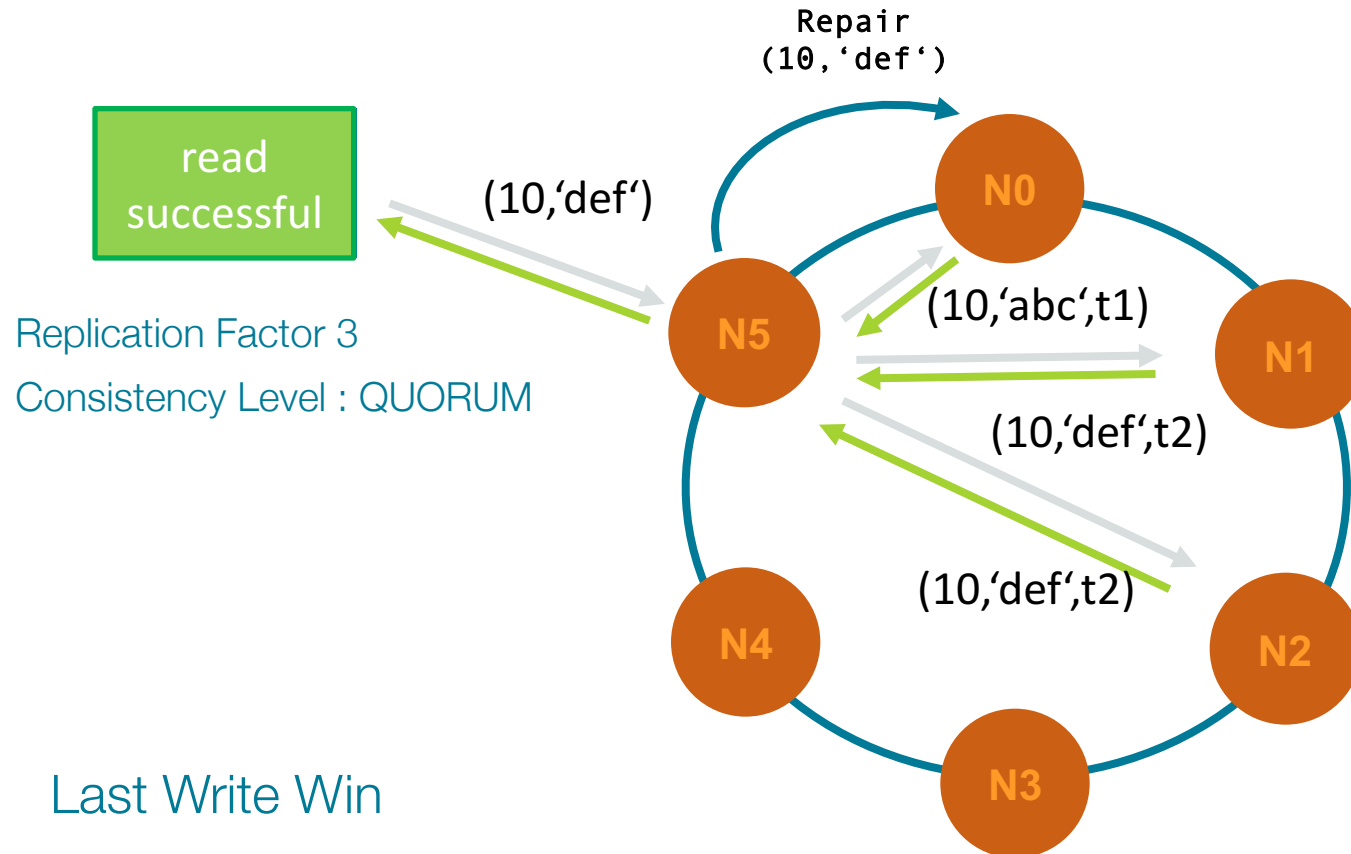
- Reading LOCAL_QUORUM succeeds
CL: LOCAL_ALL will fail

Read request handling



- quorum not met, failure
- CL: LOCAL_ONE will succeed. [See more...](#)

Read request handling – Read Repair



- Last Write Win
- $R+W > RF$ = immediate consistency
- Background vs. foreground Read Repair. [See more...](#)

Driver Code

```
Cluster cluster = Cluster.builder()  
    .addContactPoint("127.0.0.1")  
    .withLoadBalancingPolicy(new TokenAwarePolicy(DCAwareRoundRobinPolicy.builder()  
        .withLocalDc("myLocalDC")  
        .build()))  
    .build();
```

```
PreparedStatement prepared = session.prepare  
    ( "insert into sales(id, name) values (?, ?)");
```

```
BoundStatement bound = prepared.bind("4", "Gregg");
```

```
session.execute(bound); // Throws UnavailableException If consistency doesn't met, downgrade is  
                           possible with corresponding RetryPolicy. Read More...
```


Take away

- Data distribution (hash, tokens)
- Data replication (RF)
- All nodes are peer nodes , master less
- Background Read Repairs
- RetryPolicy in driver

Lab 2 : Hands-on DSE CQL

Thank You!

Eventuell Bootstrap, ReBalance, Num Tokens VNodes

- All nodes are peers
 - Including seed nodes
 - No master
 - Discovery through gossip
- Built-in replication
 - Simplify your architecture!