DataStax Enterprise Architecture

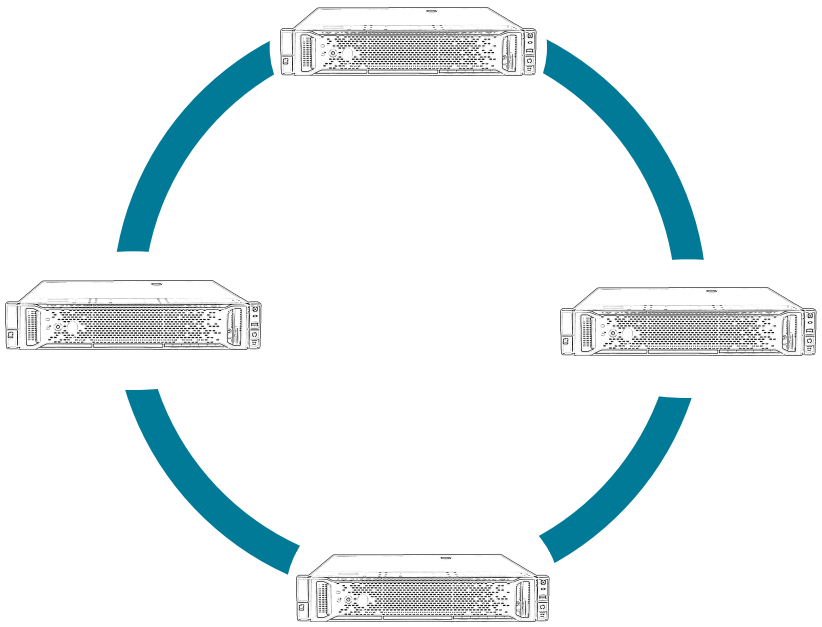
Negib Marhoul, Solution Engineer, DataStax

5. October 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

DATASTAX

Architecture

Apache Cassandra™ Architecture

- Cluster layer
- Amazon DynamoDB paper
- masterless architecture

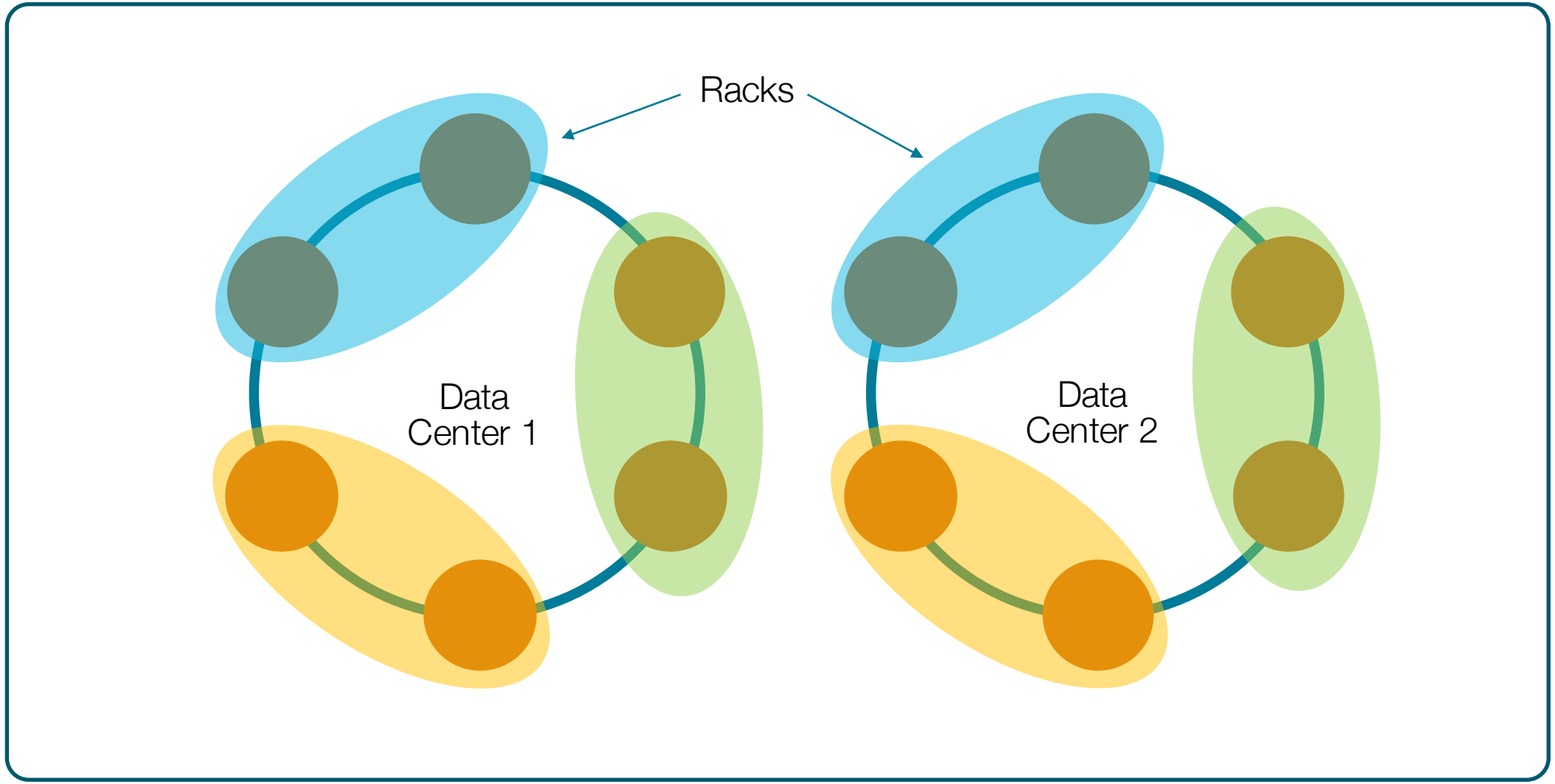
- Data-store layer
- Google Big Table paper
- Columns/columns family

- All nodes are peers
  - Including seed nodes
  - No master
  - Discovery through gossip

- Built-in replication
  - Simplify your architecture!

DATASTAX

Cluster



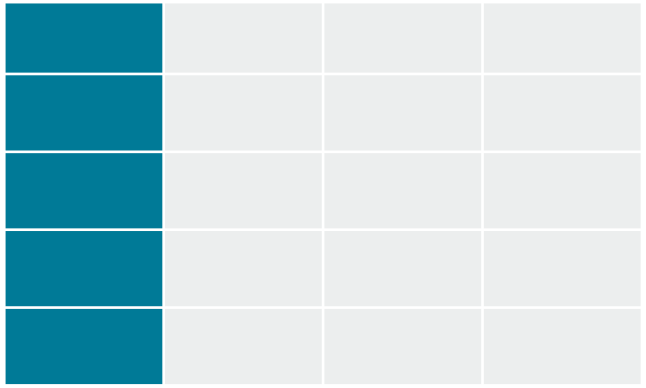
DATASTAX

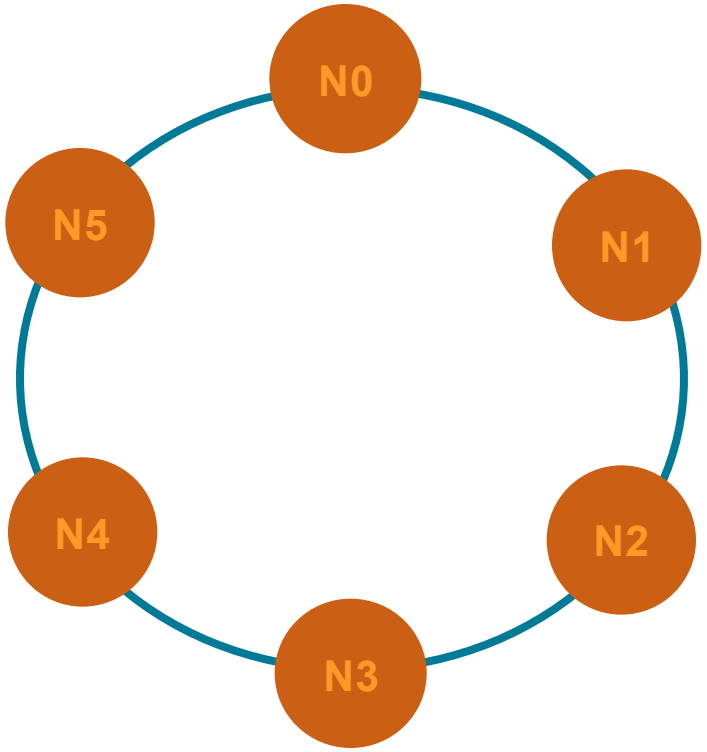
Tokens

Data is partitioned after its partition key

A unique token is allocated to a partition

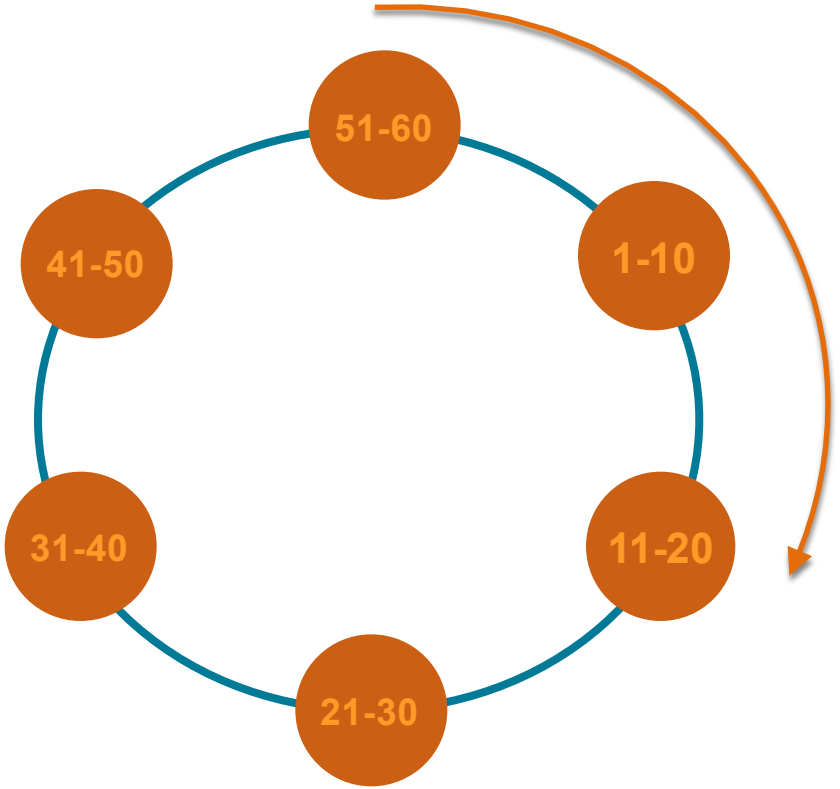
Token = random hash of #partition





DATASTAX

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
```

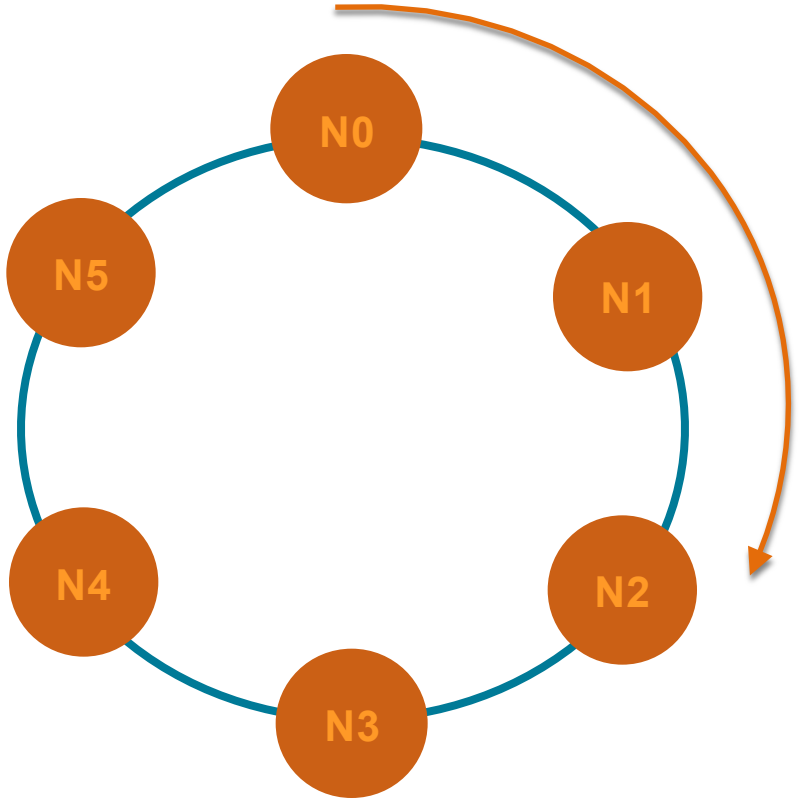

DATASTAX

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

Data Distribution

Token = hash of #partition → #node

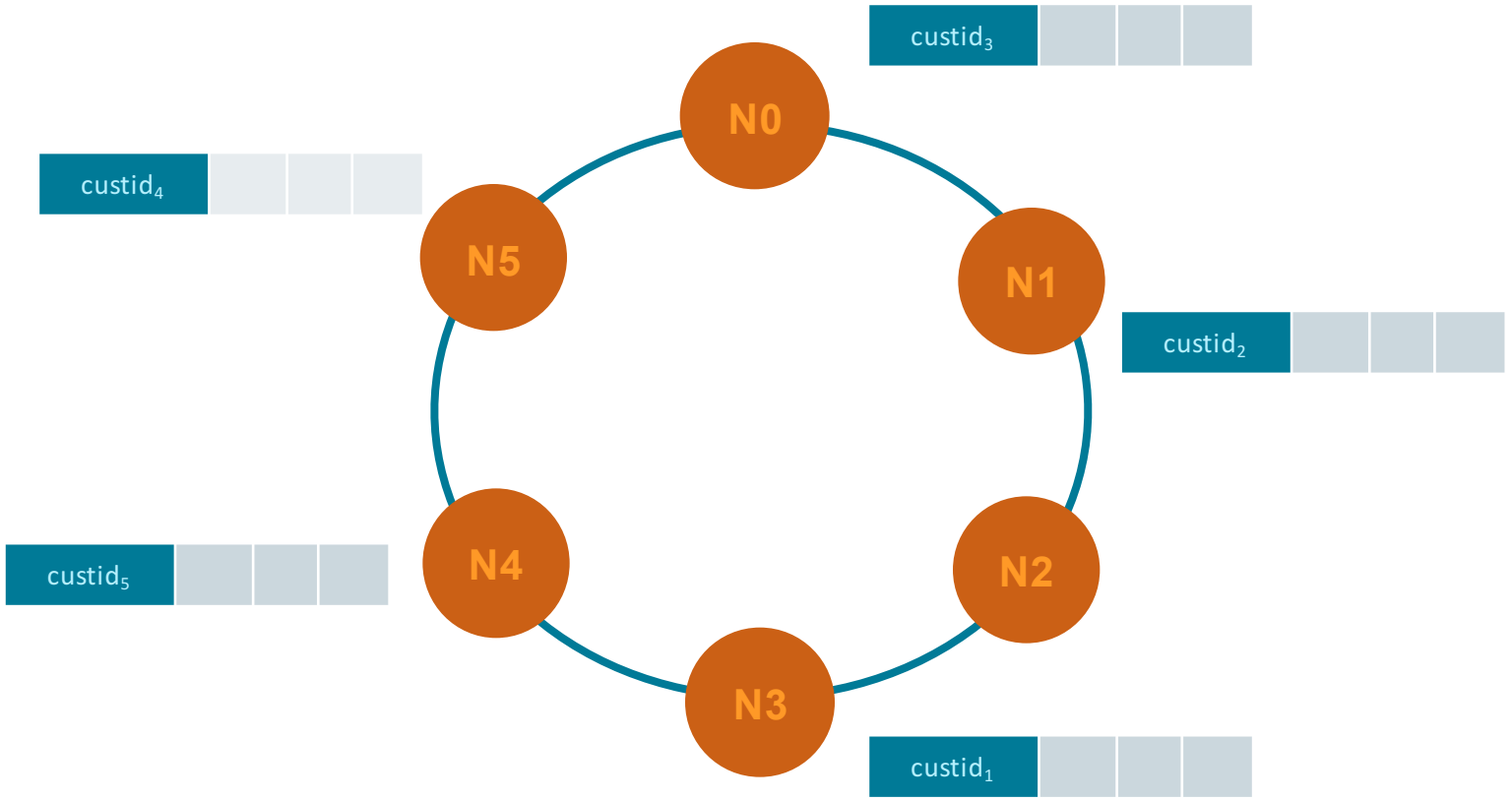
Token1	custid 1			
Token2	curstid 2			
Token3	curstid 3			
Token4	curstid 4			
Token4	curstid 5			



Data is evenly distributed and clock wise replicated

DATASTAX

Data Distribution



DATASTAX

Cassandra Query Language DDL / DML

```
CREATE KEYSPACE retailer WITH replication =
{'class': 'NetworkTopologyStrategy', 'DC1': '3'}
AND durable_writes = true;
```

Replication Factor

```
CREATE TABLE retailer.sales_by_customer (
    custid int,
    salesdt text,
    comment text,
    discount double,
    revenue double,
    PRIMARY KEY (custid, salesdt));
```

Partition Key

```
SELECT * FROM sales_by_customer where custid=1 OR custid=2 AND salesdt >=20160401;
```


DATASTAX

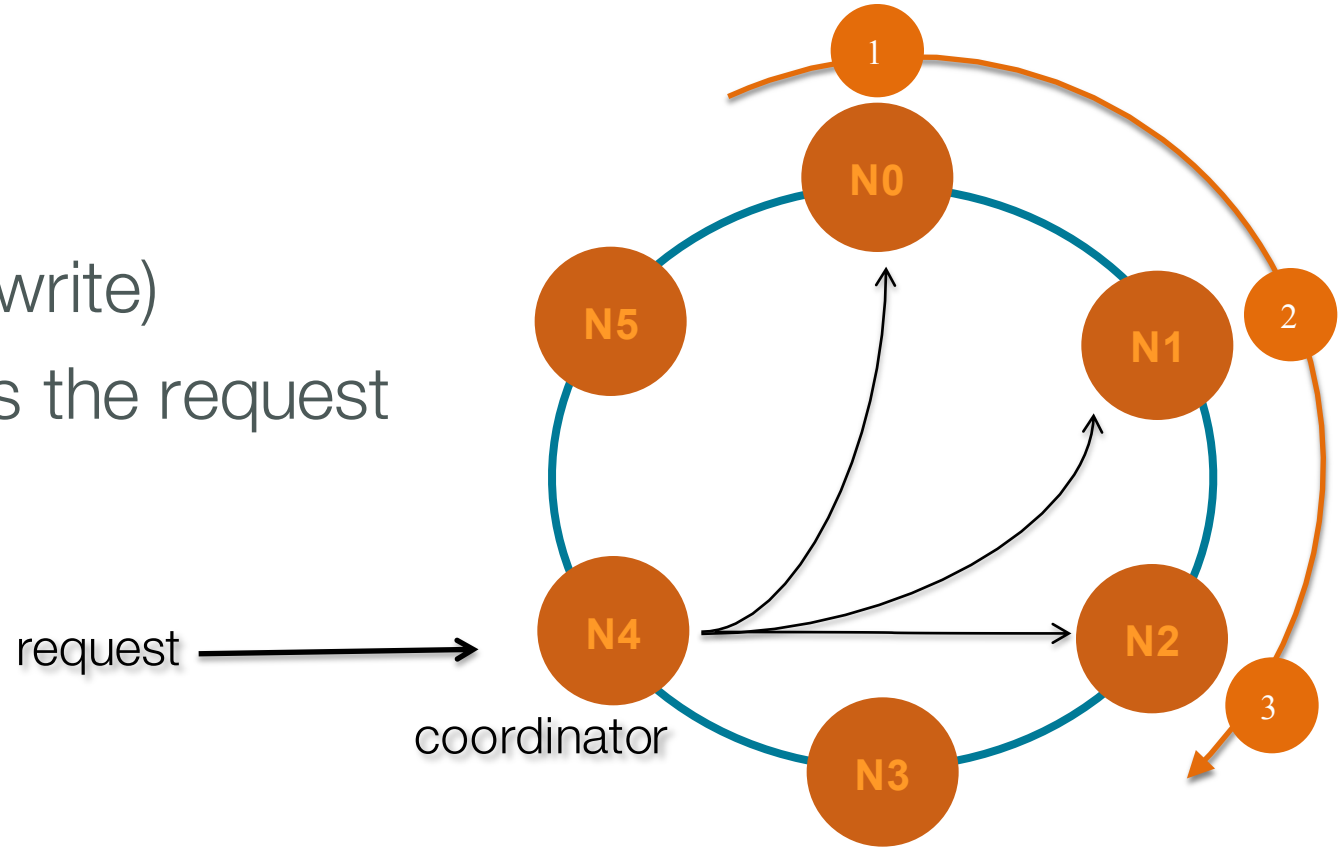
Lab 1 : Accessing the cluster

Tunable Consistency
Read and write request handling

Coordinator node

Incoming requests (read/write)

Coordinator node handles the request



Every node can be coordinator → masterless

Consistency

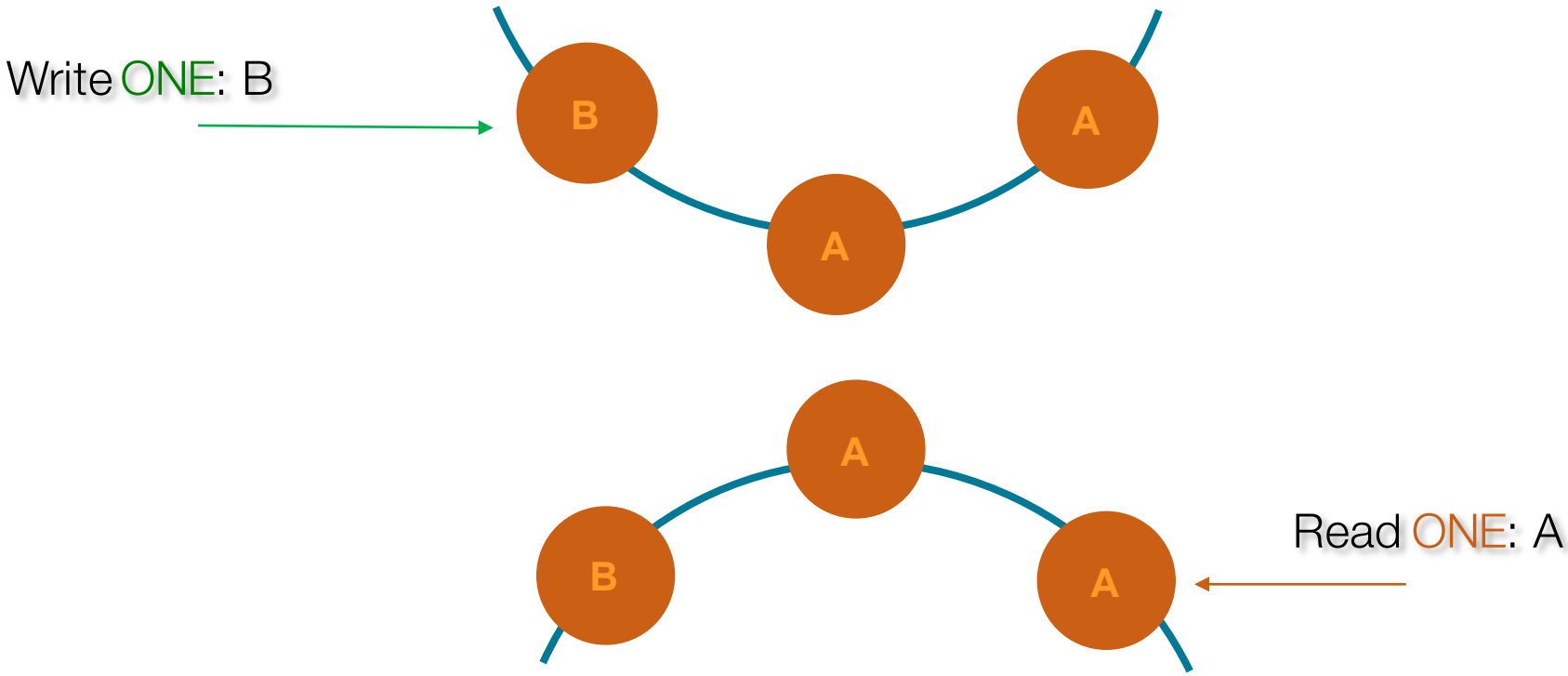
Tunable at runtime

- ONE
- QUORUM (strict majority w.r.t. RF)
- ALL

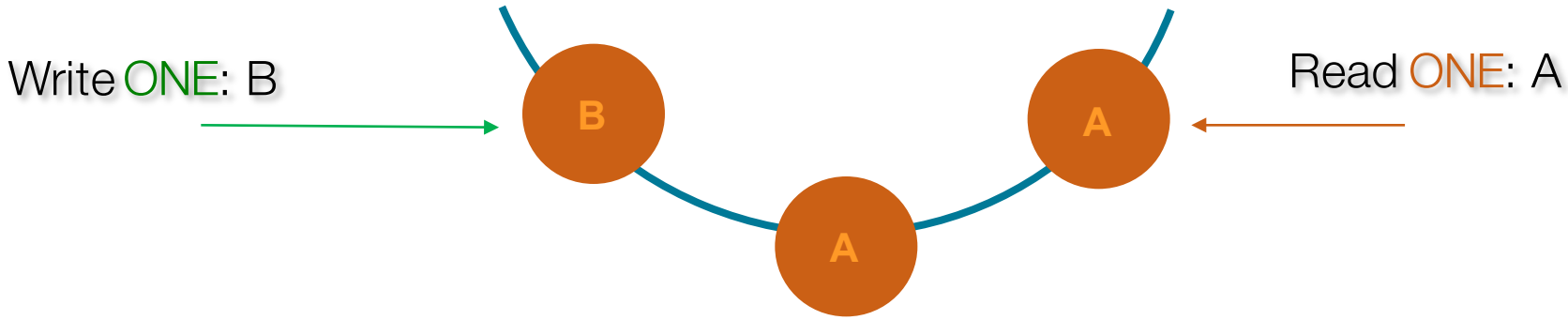
Apply both to read & write

DATASTAX

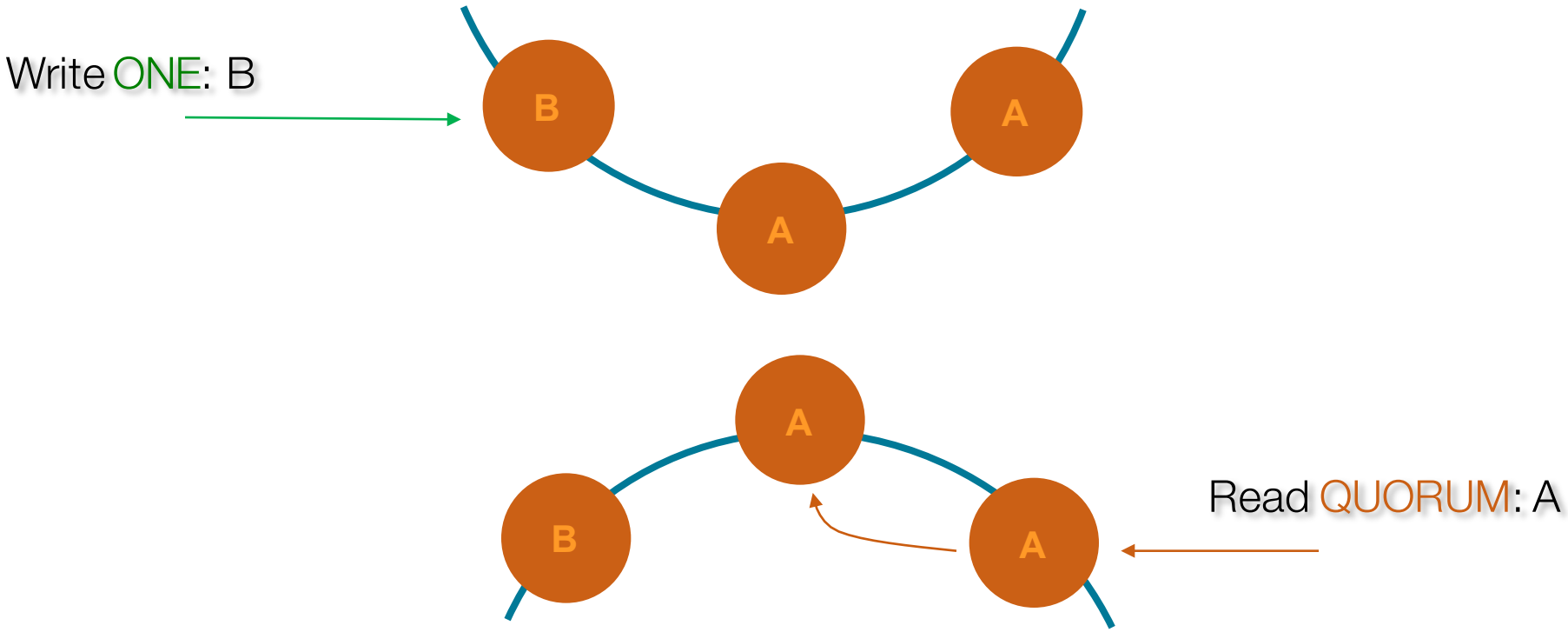
- RF = 3, Write ONE, Read ONE



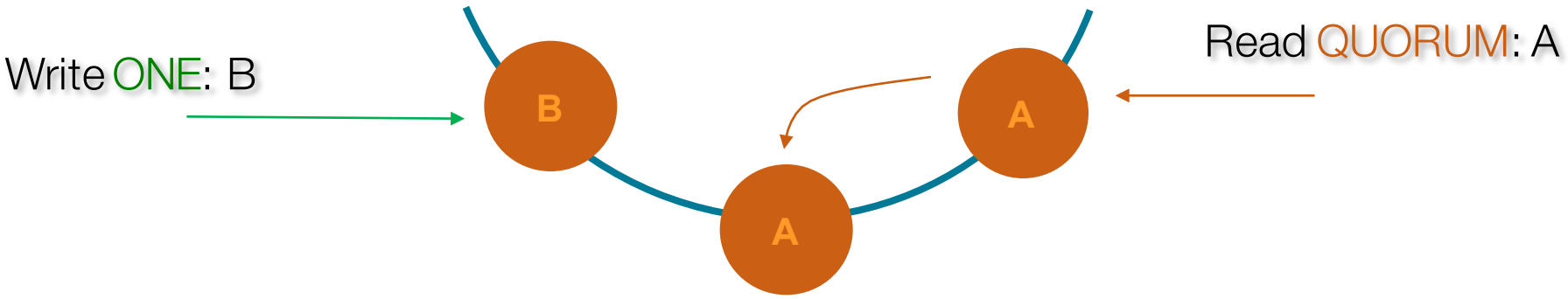
- RF = 3, Write ONE, Read ONE



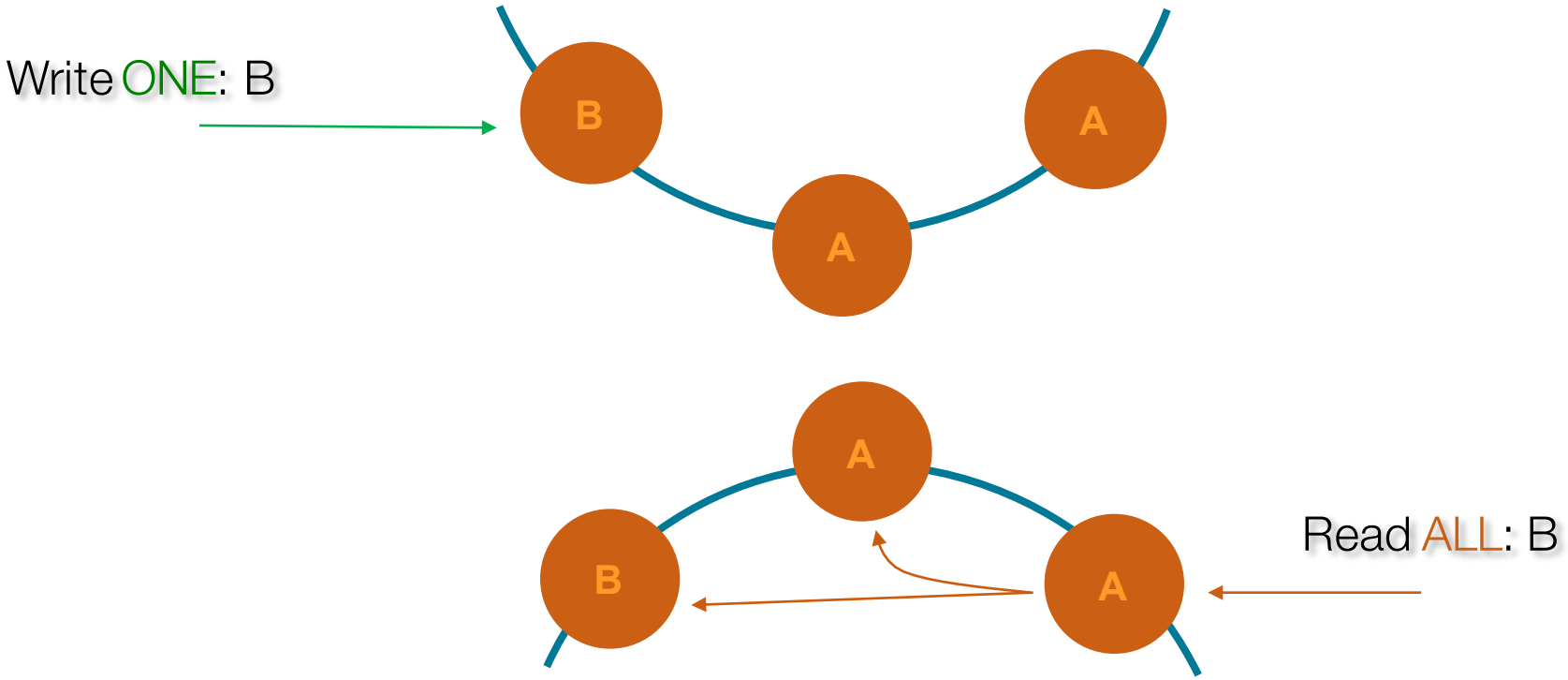
- RF = 3, Write ONE, Read QUORUM



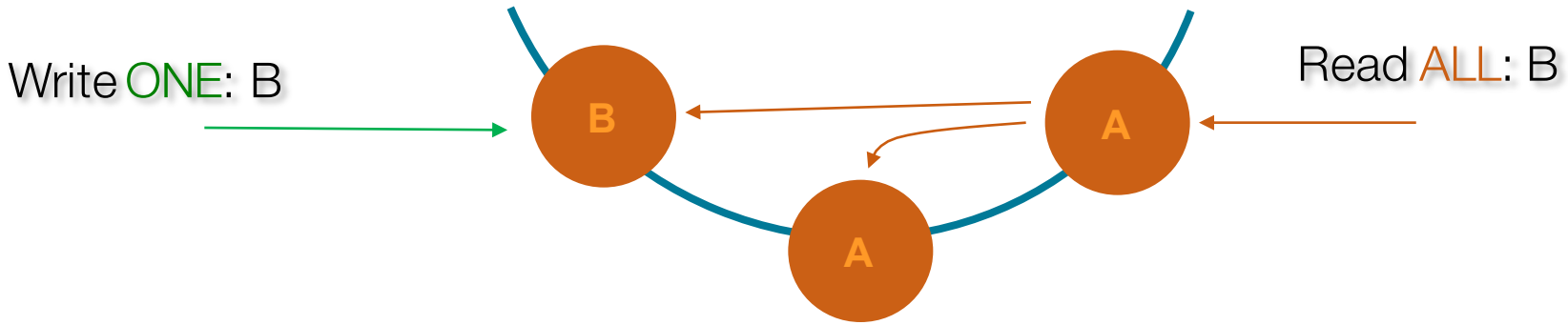
- RF = 3, Write ONE, Read QUORUM



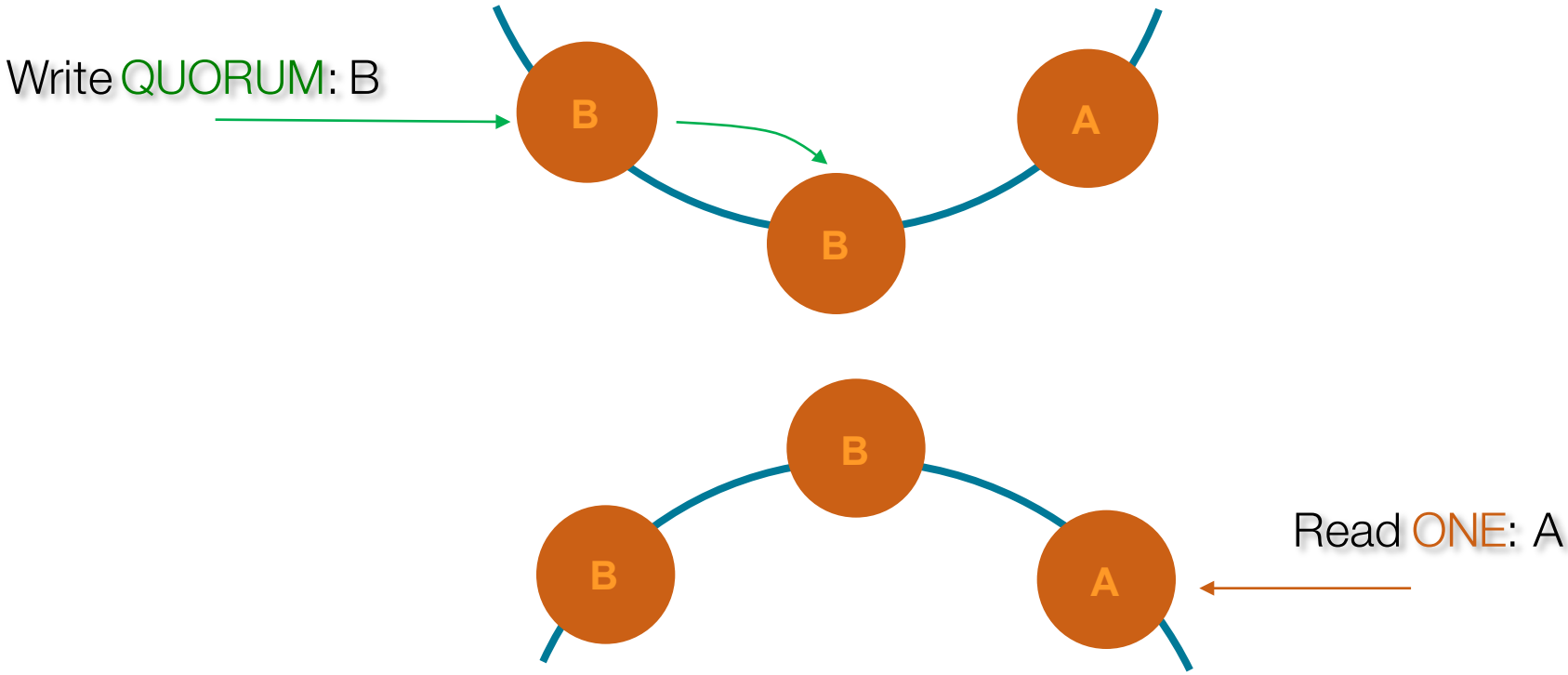
- RF = 3, Write ONE, Read ALL



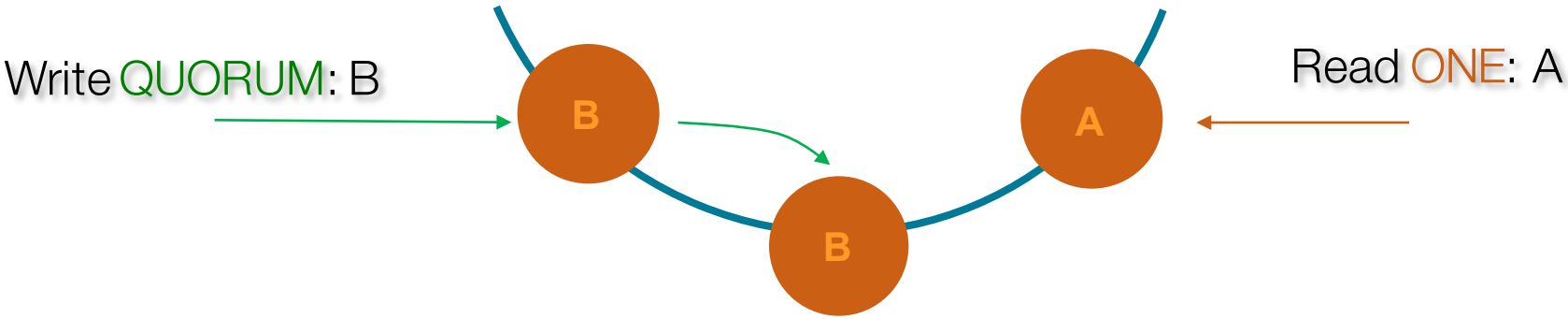
- RF = 3, Write ONE, Read ALL



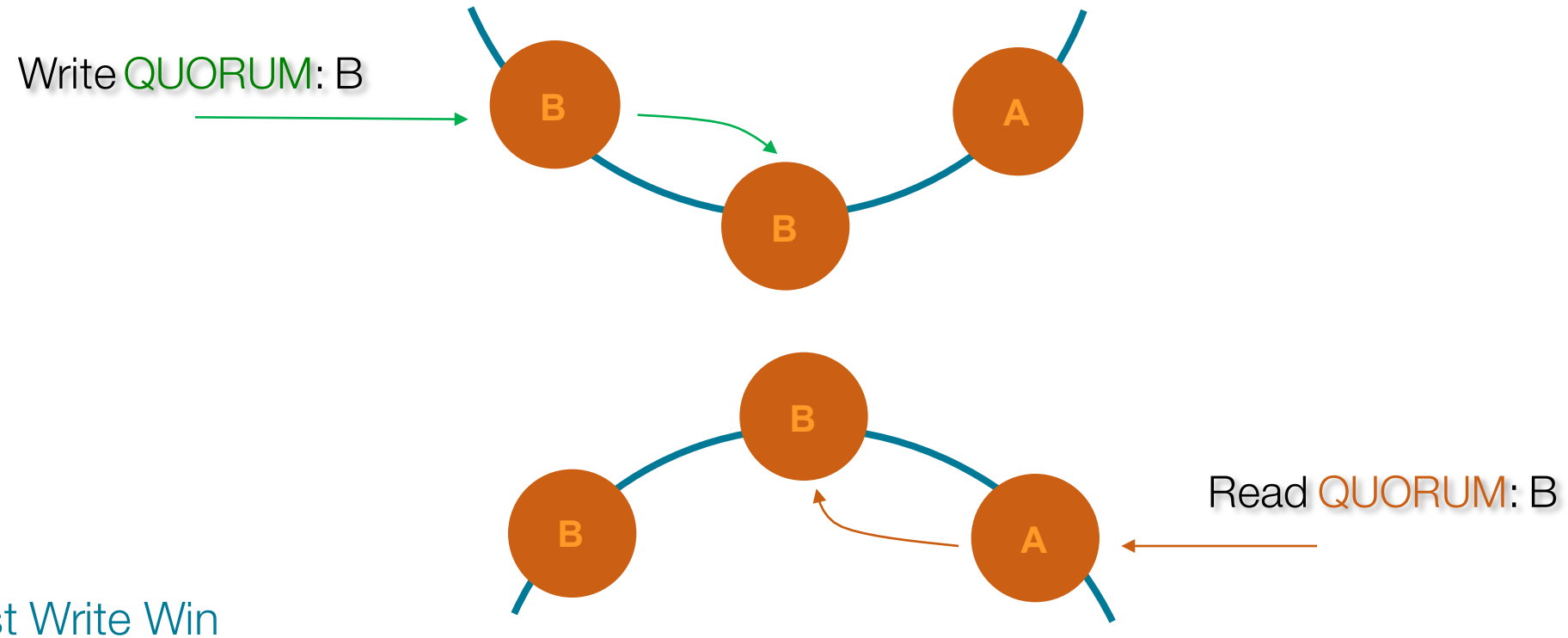
- RF = 3, Write QUORUM, Read ONE



- RF = 3, Write QUORUM, Read ONE

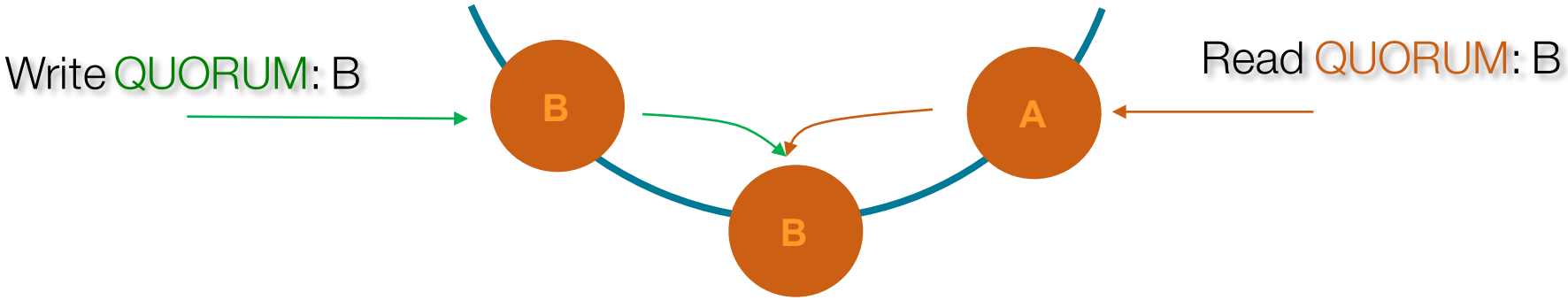


- RF = 3, Write QUORUM, Read QUORUM



- Last Write Win
- R+W > RF = immediate consistency
- Background vs. foreground Read Repair. See more...

- RF = 3, Write QUORUM, Read QUORUM



- Last Write Win
- R+W > RF = immediate consistency
- Background vs. foreground Read Repair. See more...

Consistency trade-off

DATASTAX

Consistency summary

$ONE_{Read} + ONE_{Write}$

available for read/write even (N-1) replicas down

$QUORUM_{Read} + QUORUM_{Write}$

available for read/write even 1+ replica down

DATASTAX

- Background vs. Forground 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

CRUD

```
INSERT INTO sales_by_customer (custid, salesdt, revenue) VALUES('1', '20160103', 799);

UPDATE sales_by_customer SET discount = 10 WHERE custid = 1;

DELETE sales_by_customer FROM sales_by_order WHERE custid = 1;

SELECT revenue FROM sales_by_customer WHERE custid = 1;
```


DATASTAX

Time To Live

```
INSERT INTO users(…) VALUES(...) USING TTL = 3600;

UPDATE users USING TTL = 3600 SET age = xxx WHERE ...;
```


DATASTAX

Time To Live

```
CREATE TABLE readings_by_sensor(
   sensor_id text,
   collect_time timestamp,
   electricity long,
   gas long,
   health double,
   PRIMARY KEY (sensor_id, collect_time)
)
WITH CLUSTERING ORDER BY (collect_time ASC)
AND default_time_to_live = 86400; // TTL 1 day
```

```
INSERT INTO readings_by_sensor(…) VALUES(...) USING TTL = 86400;
```

Collections

```
CREATE TABLE users (
        login text,
        name text,
        age int,
        friends set<text>,
        hobbies list<text>,
        languages map<int, text>,
        …
        PRIMARY KEY(login));
```


DATASTAX

User Defined Types (UDT)

```
CREATE TYPE address (
        street_number int,
        street_name text,
        postcode int,
        country text);

    CREATE TABLE users (
        login text,
        …
        location address,
        …
        PRIMARY KEY(login));
```


DATASTAX

UDT DML

```
INSERT INTO users(login,name, location) VALUES (
        ‘jdoe’,
        ’John DOE’,
        {
            ‘street_number’: 124,
            ‘street_name’: ‘Congress Avenue’,
            ‘postcode’: 95054,
            ‘country’: ‘USA’
        });
```

```
UPDATE users set location =
        {
            ‘street_number’: 125,
            ‘street_name’: ‘Congress Avenue’,
            ‘postcode’: 95054,
            ‘country’: ‘USA’
        }
    WHERE login = jdoe;
```


DATASTAX

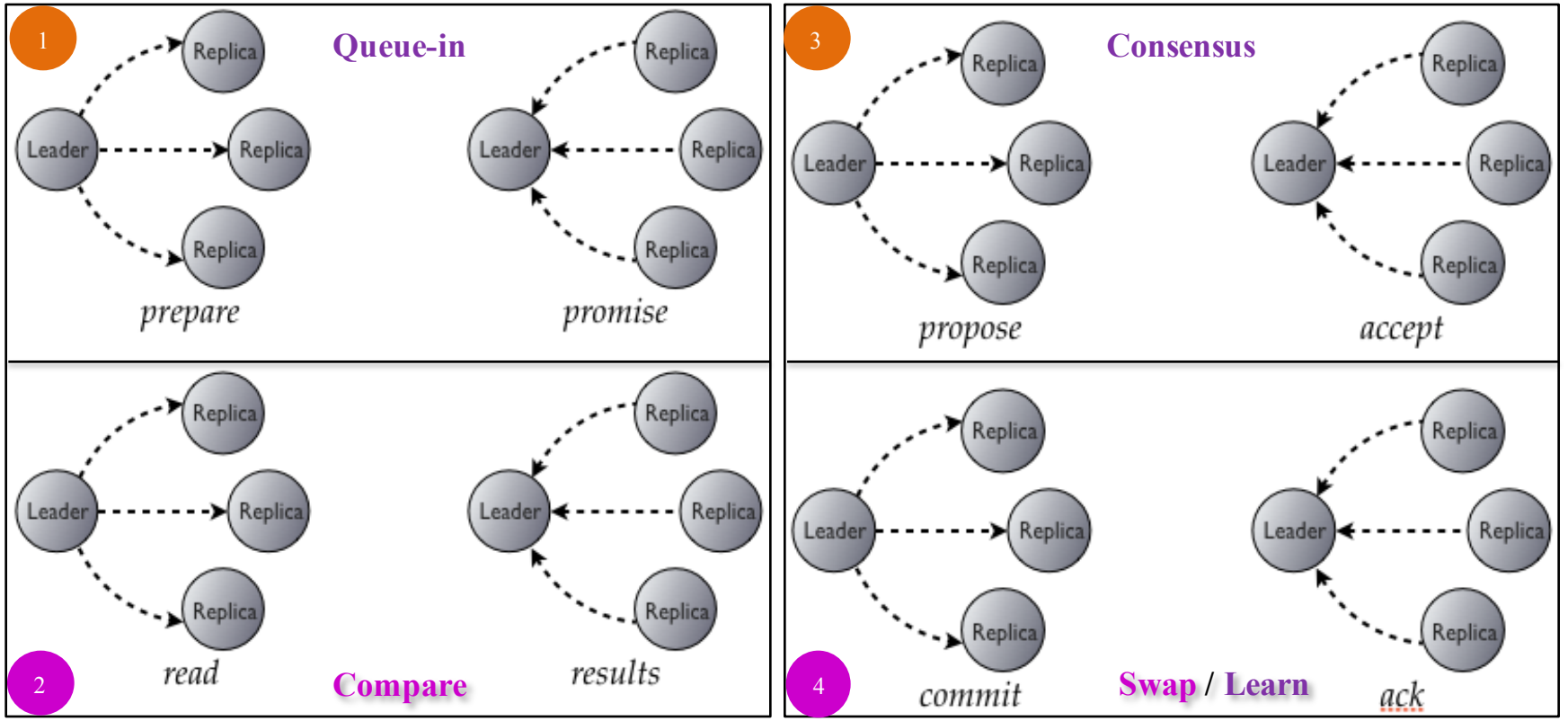
Light Weight Transactions

```
INSERT INTO users(…) VALUES(...) IF NOT EXISTS;

DELETE users WHERE ... IF EXISTS; UPDATE users

SET age = xxx WHERE ... IF age = 30;
```

Linearizable writes on a single partition

DATASTAX

Lightweight Transaction (LWT)
PAXOS implementation



DATASTAX

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_by_customer(custid, salesdt) values (?, ?)”);

BoundStatement bound = prepared.bind(”1", ”20170102");

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


DATASTAX

Lab 2 : Hands-on DSE CQL

Vielen Dank!

Eventuell Bootstrap, ReBalance, Num Tokens VNodes

- All nodes are peers
  - Including seed nodes
  - No master
  - Discovery through gossip

- Built-in replication
  - Simplify your architecture!

DATASTAX