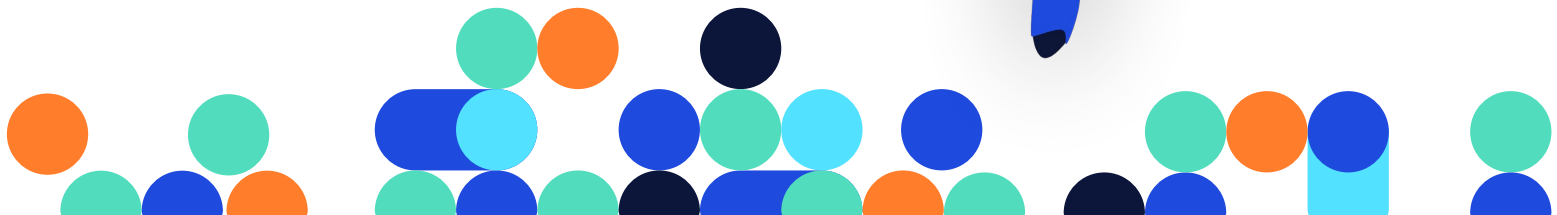
DataStax Developer Day

DATASTAX

Core Cassandra



Why do I need DSE & Apache Cassandra™?

- CARDS – an easy way to remember
  - Contextual – relevant data in context
  - Available – always on, no downtime
  - Realtime – response time in MS
  - Distributed – many servers in datacenters around the world
  - Scalable – near linear increase for each additional server

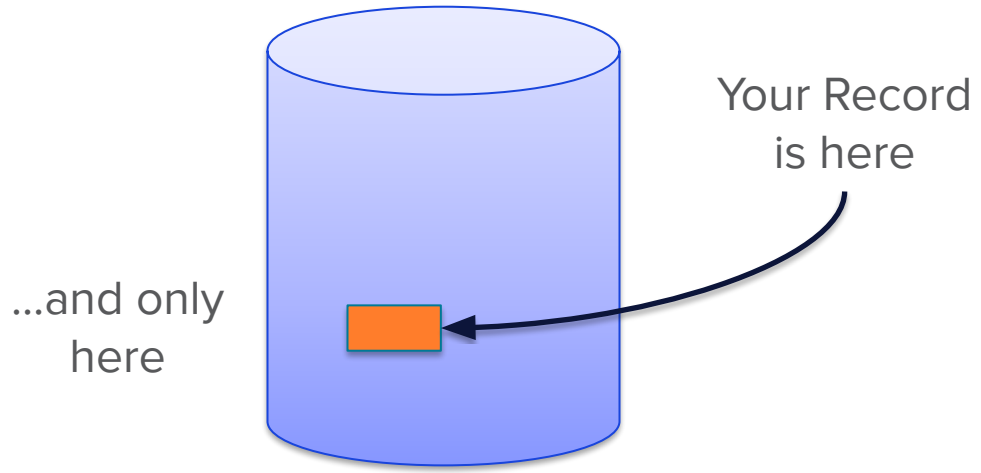
DataStax Developer Day

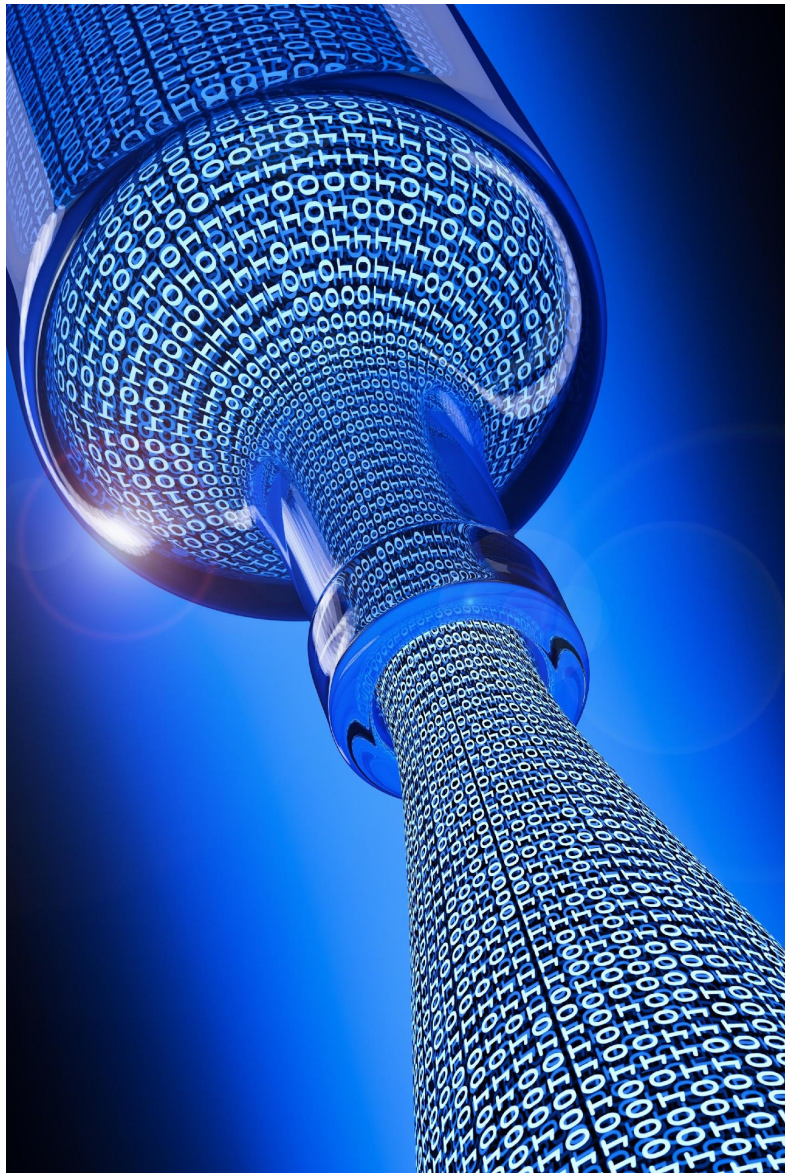
DATASTAX

Cassandra Architecture

Problems with Traditional RDBMS

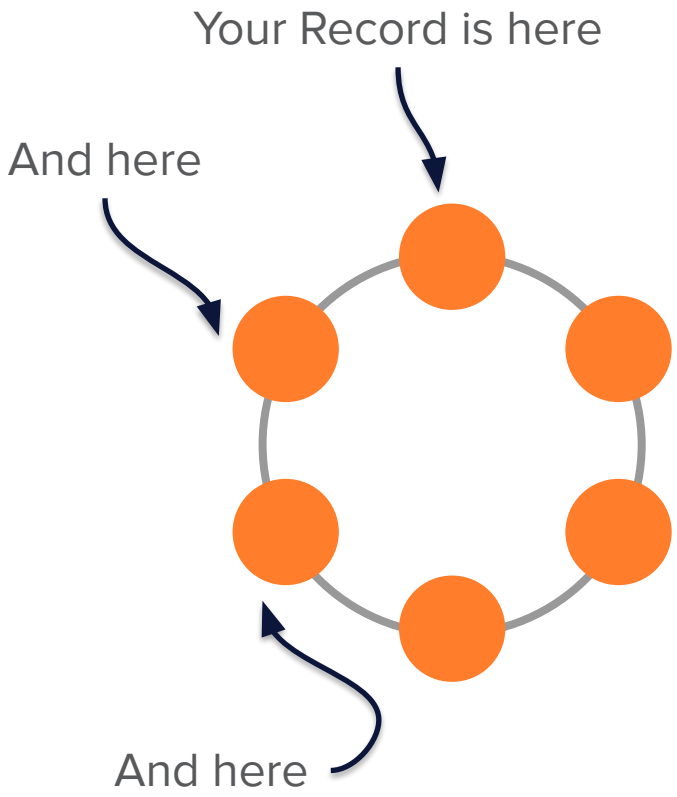
- Each record is in one location
- Update in place – causes bottlenecks
  - Reduced throughput and latency
- Single point of failure
  - Availability risk
- Dogmatic consistency

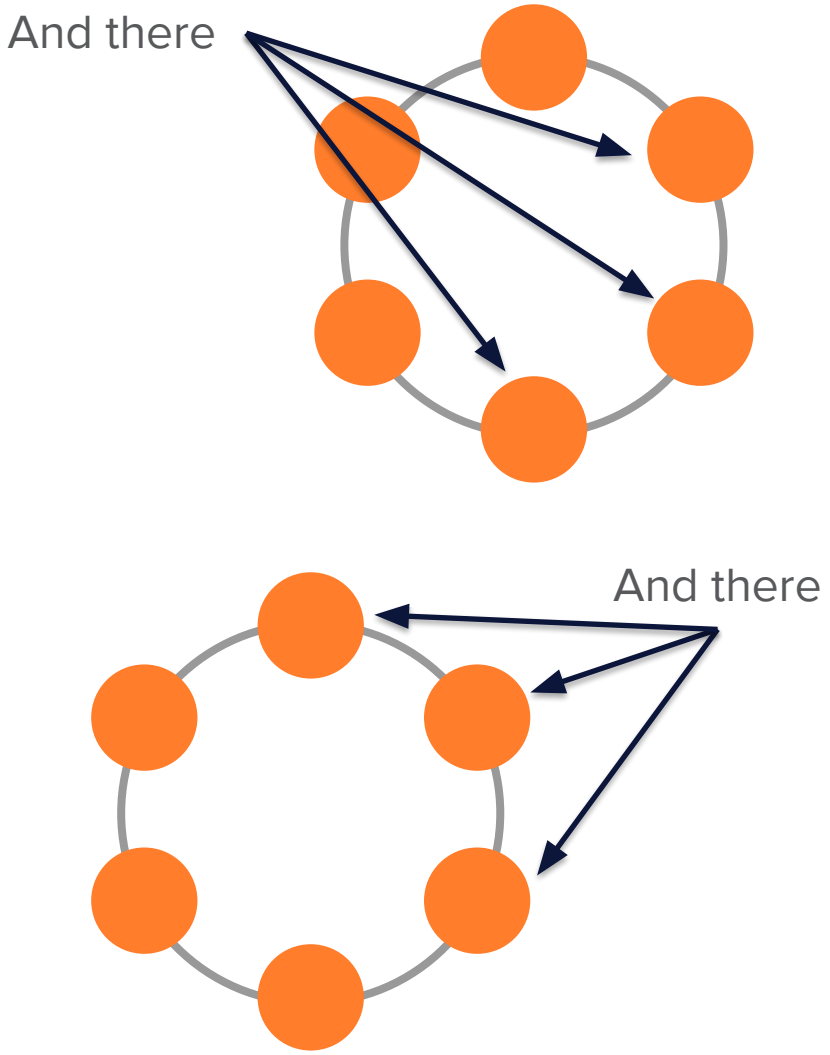




Cassandra Features

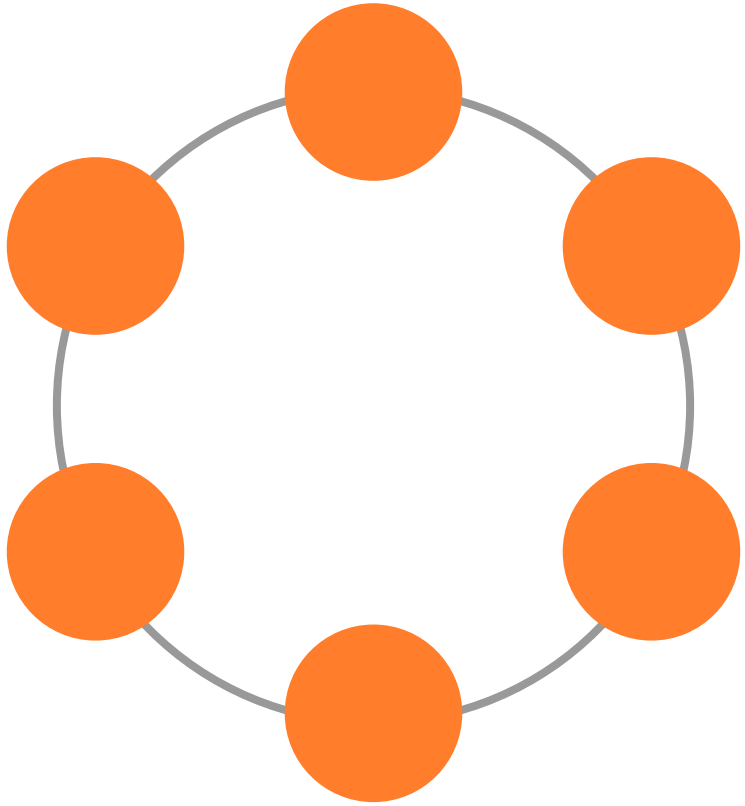
- Distributed
  - Available
  - Responsive
  - Scalable
- Log-structured
  - No bottlenecks
- Tunable consistency





Cassandra is Distributed

- Cassandra clusters have many nodes



Cassandra is Distributed

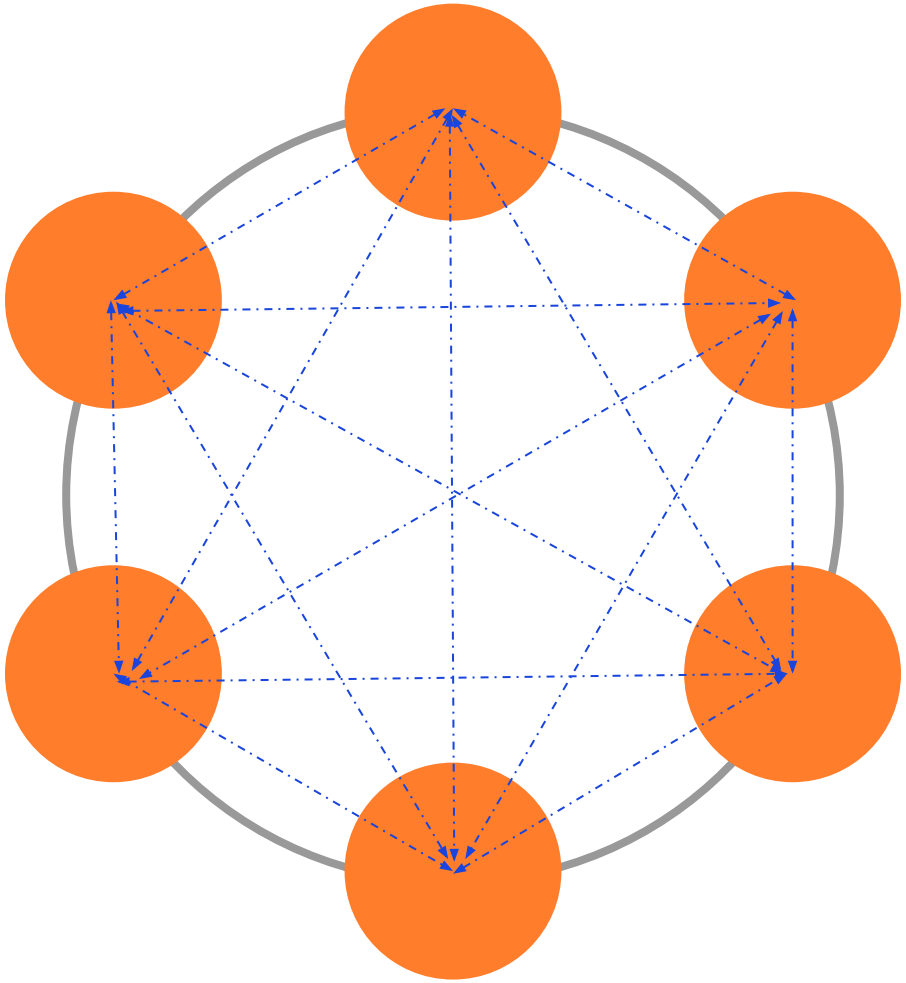
- Cassandra clusters have many nodes
- How does Cassandra manage all the nodes?
- There is no ”boss” node

Cassandra is Distributed

- Cassandra clusters have many nodes
- How does Cassandra manage all the nodes?
- There is no “boss” node
- The nodes collaborate

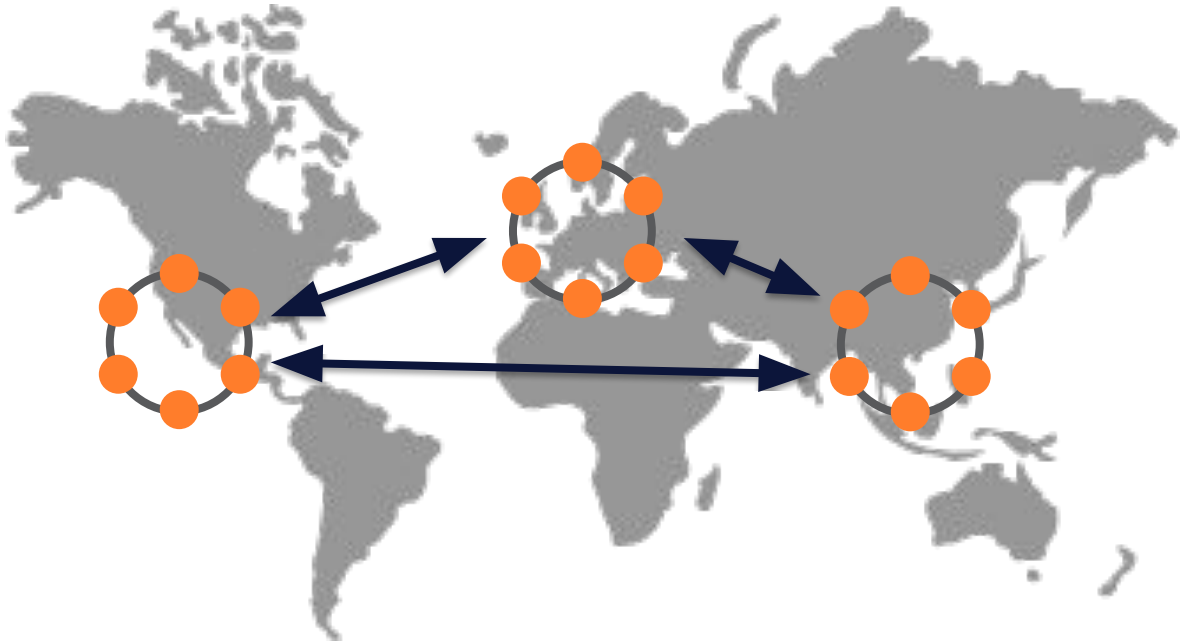
Cassandra is Distributed and Highly Available

- Nodes organized in a ring
- Nodes monitor each other's health status using a gossip protocol
- Cassandra can route requests away from nodes that are down or slow

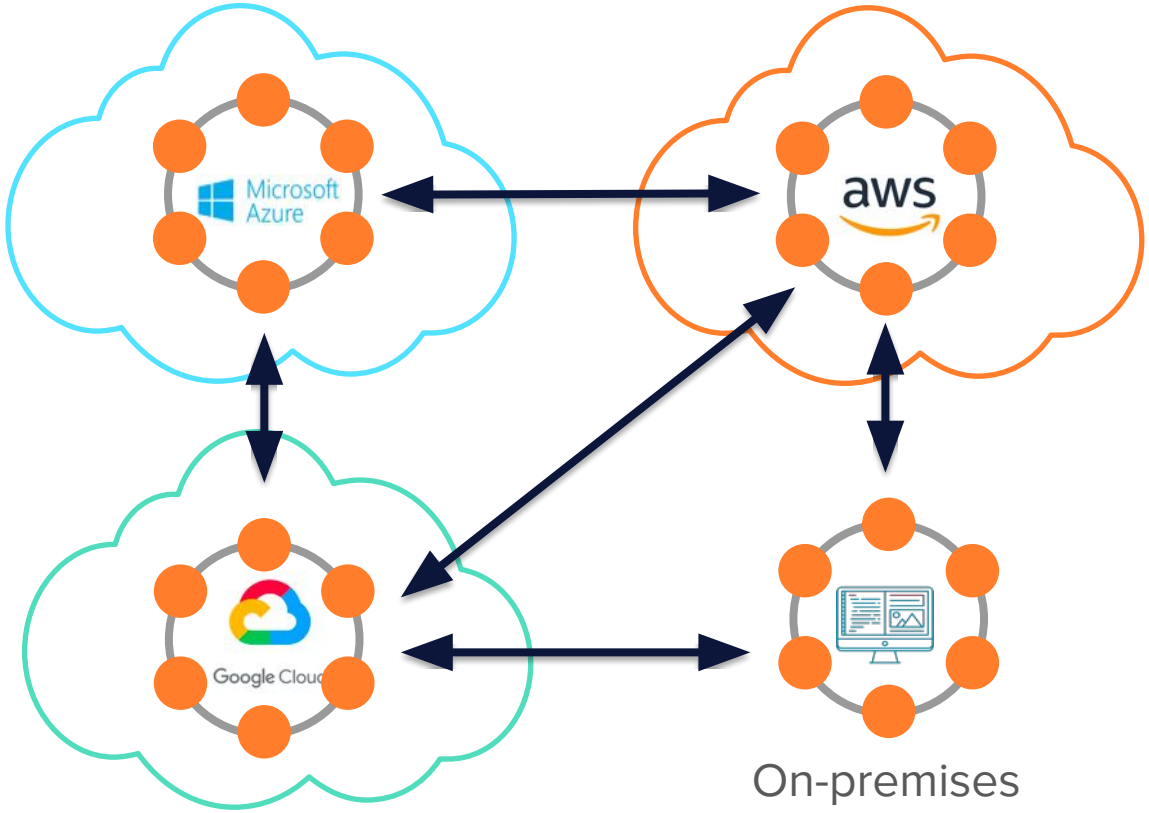


Data Distributed Everywhere

- Geographic Distribution



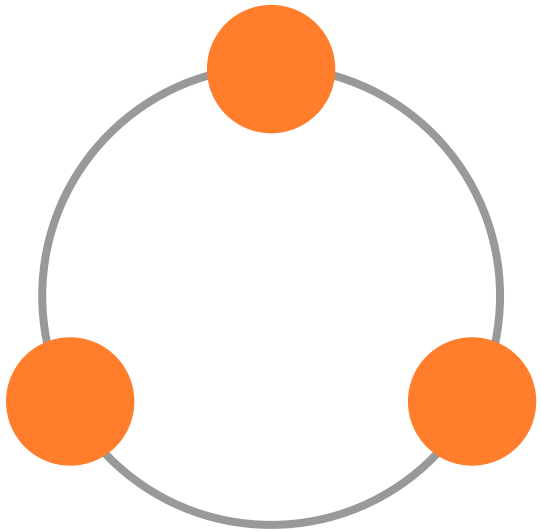
- Hybrid-Cloud and Multi-Cloud



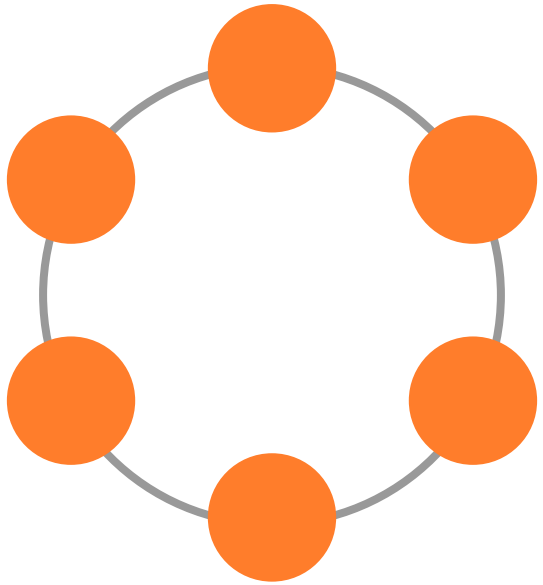
Horizontal vs. Vertical Scaling

1 Installation = 1 NODE
✔ Capacity: ± 1TB
✔ Throughput: 3000 Tx/sec/core

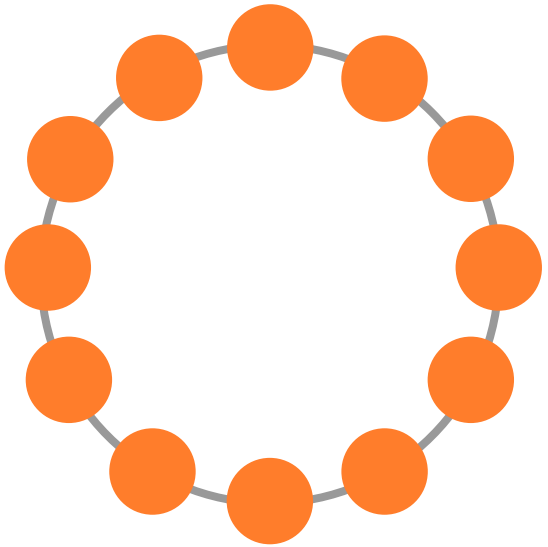
- Vertical scaling requires one large expensive machine
- Horizontal scaling requires multiple less-expensive commodity hardware



100,000 transactions/second



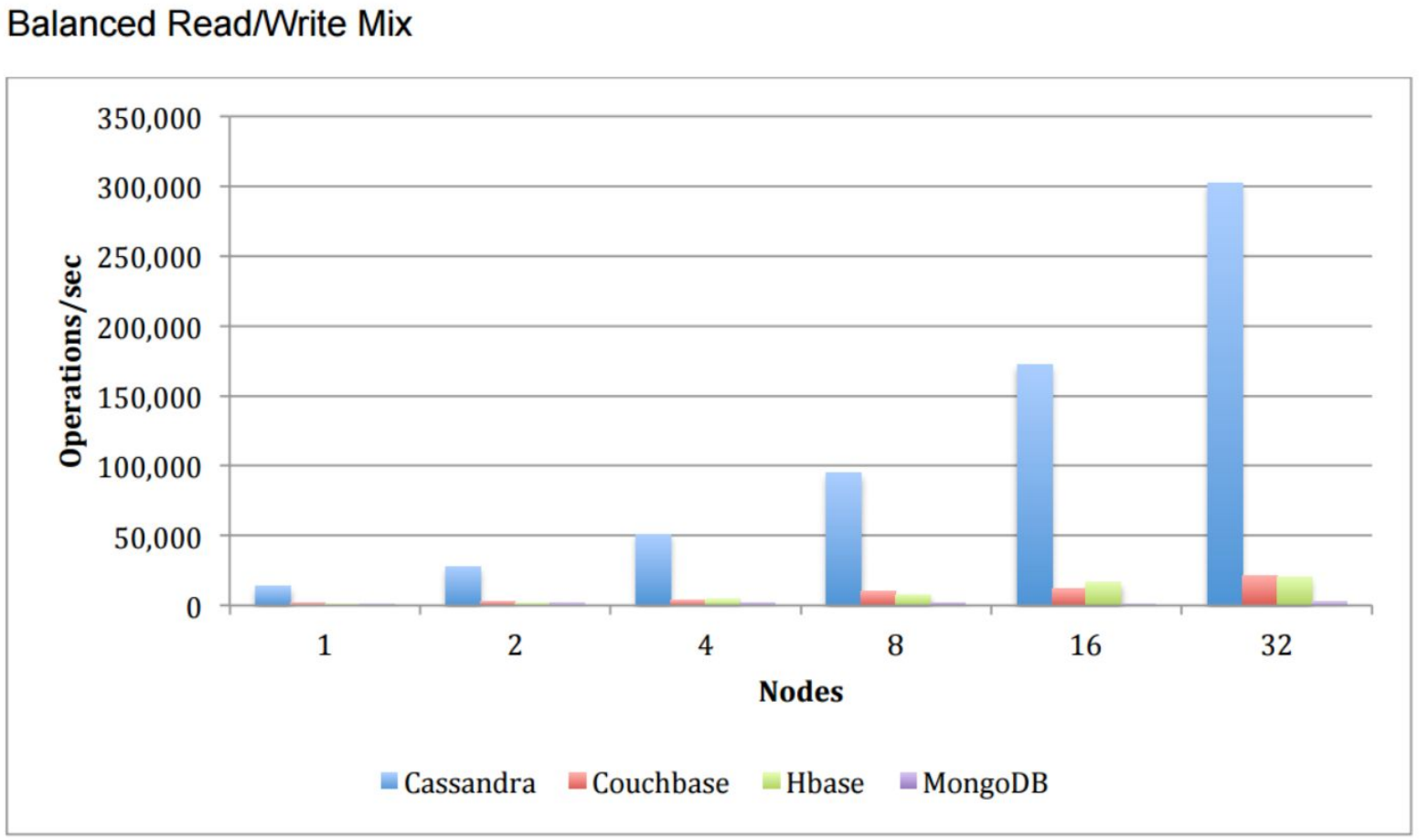
200,000 transactions/second



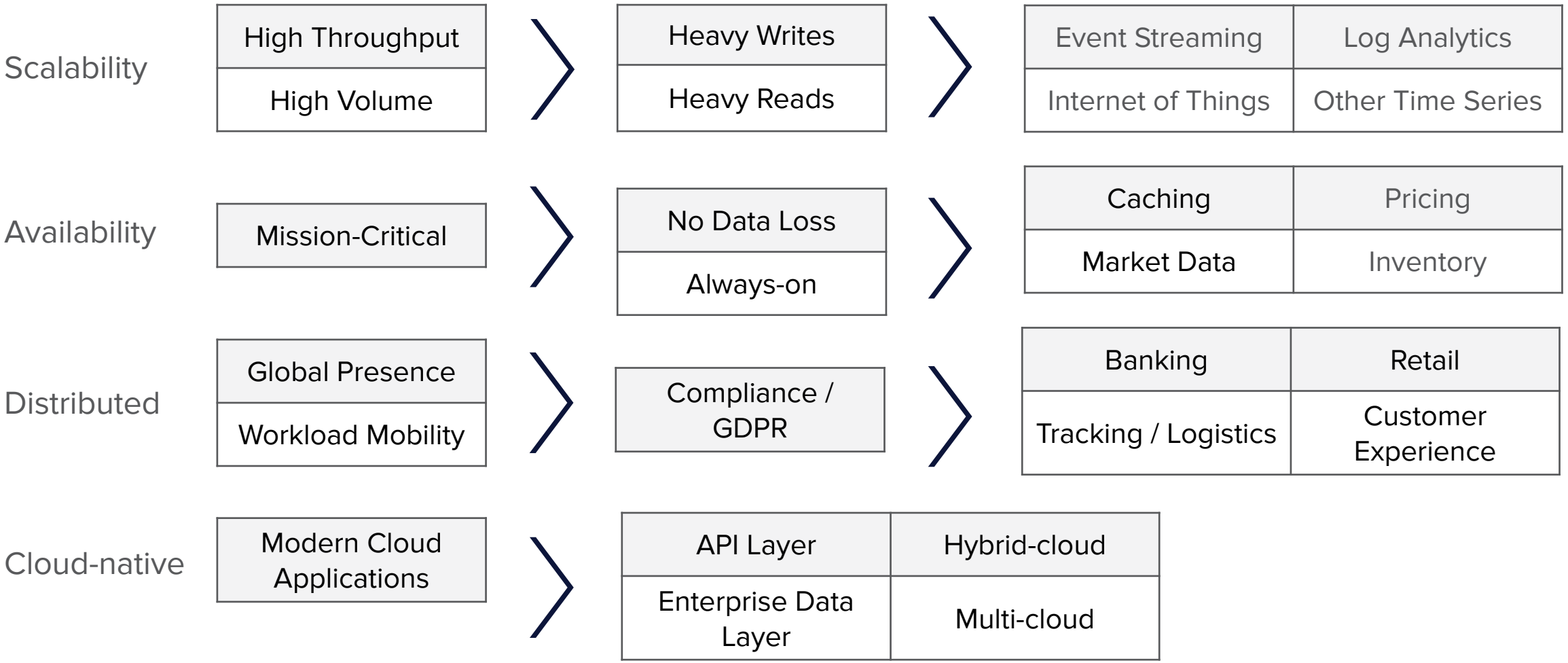
400,000 transactions/second

Scales Linearly

- Need more capacity?
- Need more throughput?
- Add nodes!



Understanding Use Cases

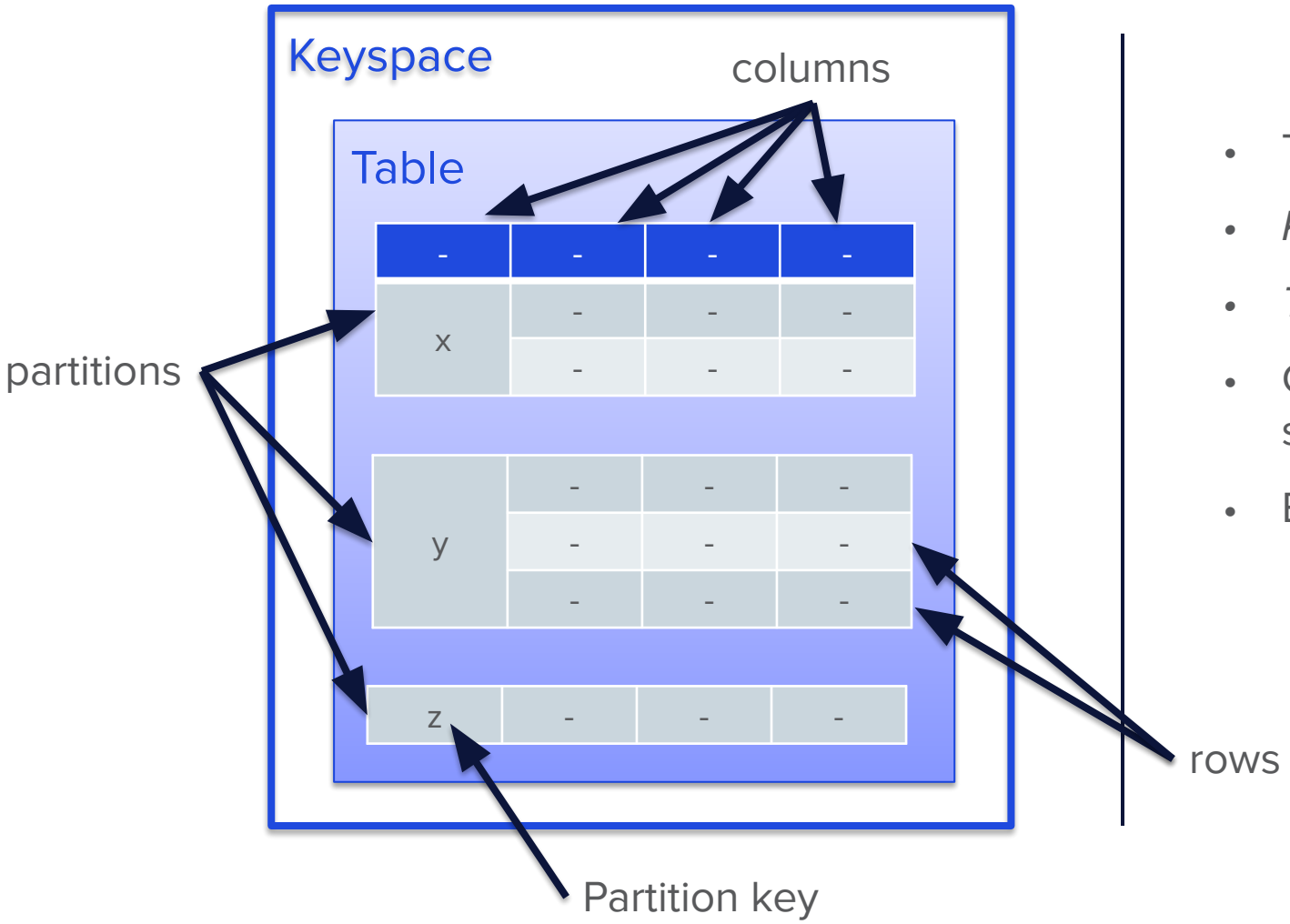


DataStax Developer Day

DATASTAX

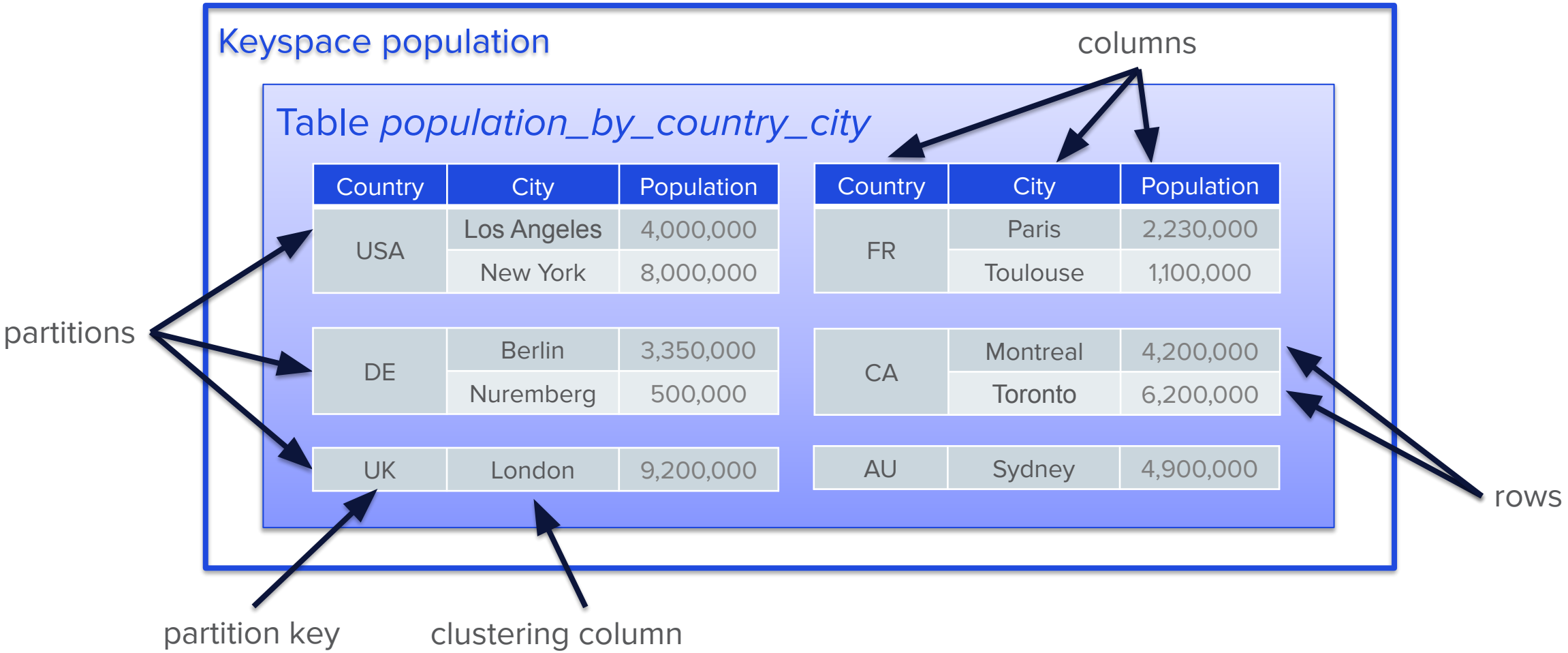
Cassandra's Data Model

How does Cassandra structure data?



- Tabular data model, with one twist
- *Keyspaces* contain *tables*
- *Tables* are organized in *rows* and *columns*
- Groups of related rows, called *partitions*, are stored together on the same node (or nodes)
- Each row contains a *partition key*
  - One or more columns that are hashed to determine which node(s) store that data

Example Data – City populations organized by country



Example Data – City populations organized by country

Keyspace population

Table *population_by_country_city*

Country	City	Population
USA	Los Angeles	4,000,000
	New York	8,000,000
DE	Berlin	3,350,000
	Nuremberg	500,000
UK	London	9,200,000

CQL Equivalent:

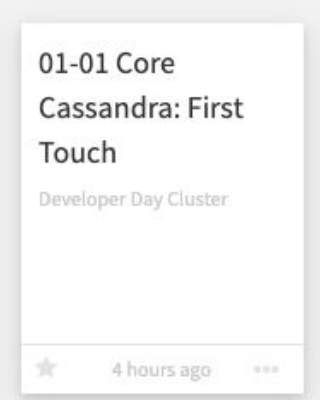
```
CREATE TABLE population_by_country_city (
  country text,
  city text,
  population int,
  PRIMARY KEY((country), city)
);
```

partition key

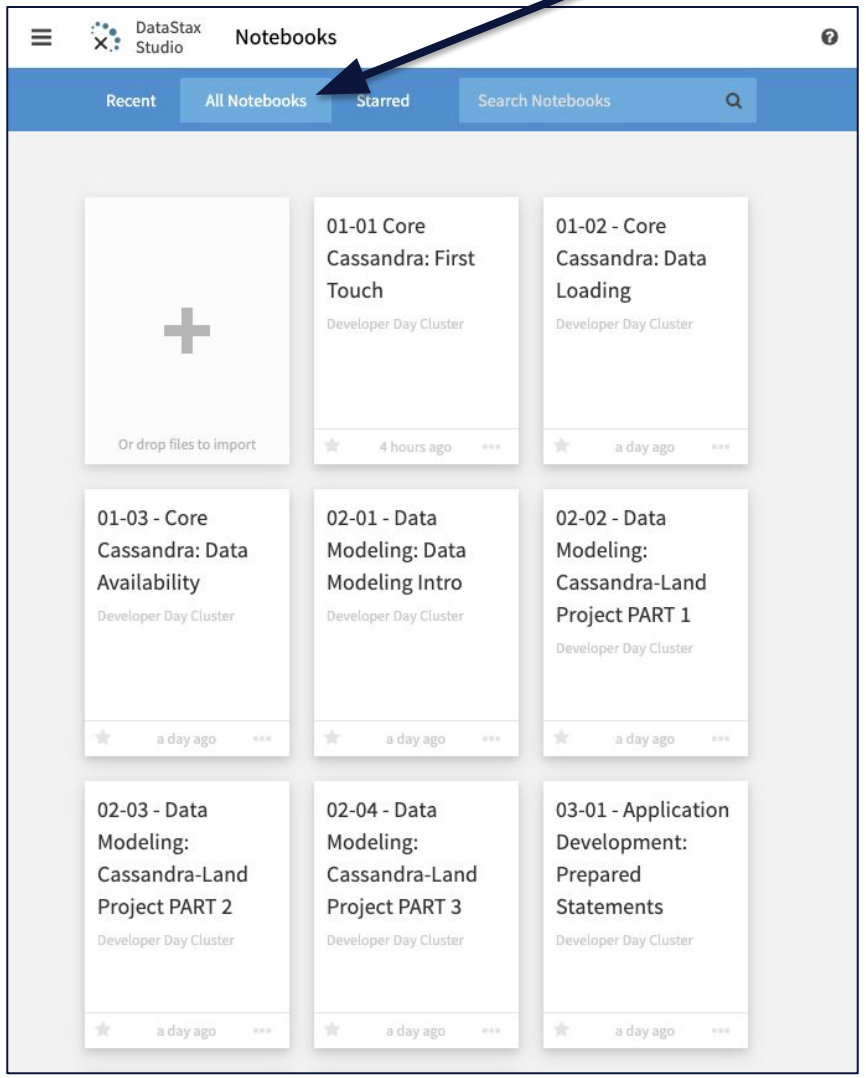
clustering column

Want to try it?

- Let's look at a cluster!
- Open a browser
  - Open the link for "Studio" from the instance you were provided
- Open notebook
  - "Core Cassandra: First Touch"



Sort the notebooks by title



Apache Cassandra™ First Touch – Quick Review

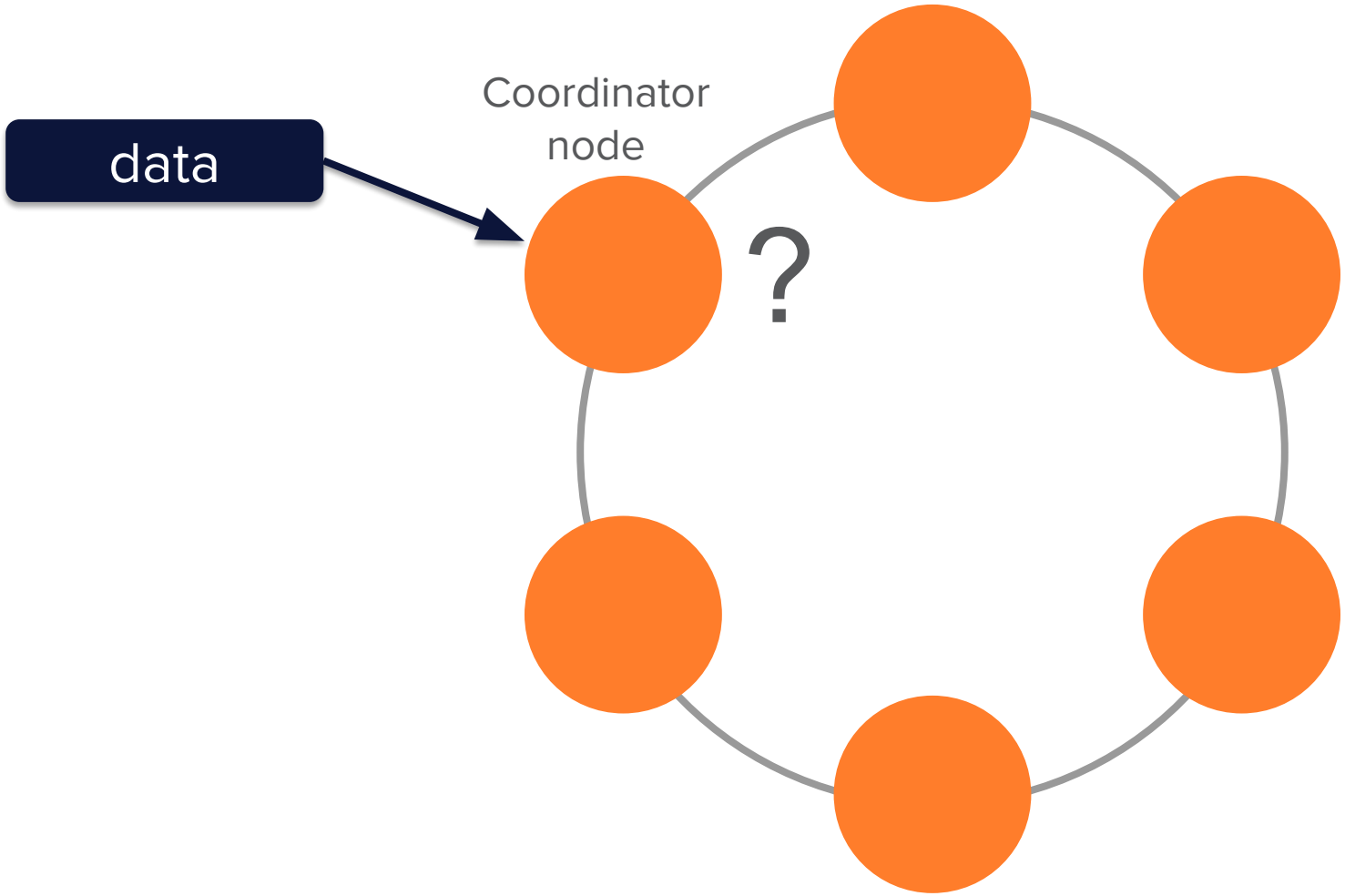
- Key Take-aways:
  - Clusters contain keyspaces
  - Keyspaces contain tables
  - Tables contain partitions
  - Partitions contain rows and columns
  - CQL has syntax similar to SQL

DataStax Developer Day

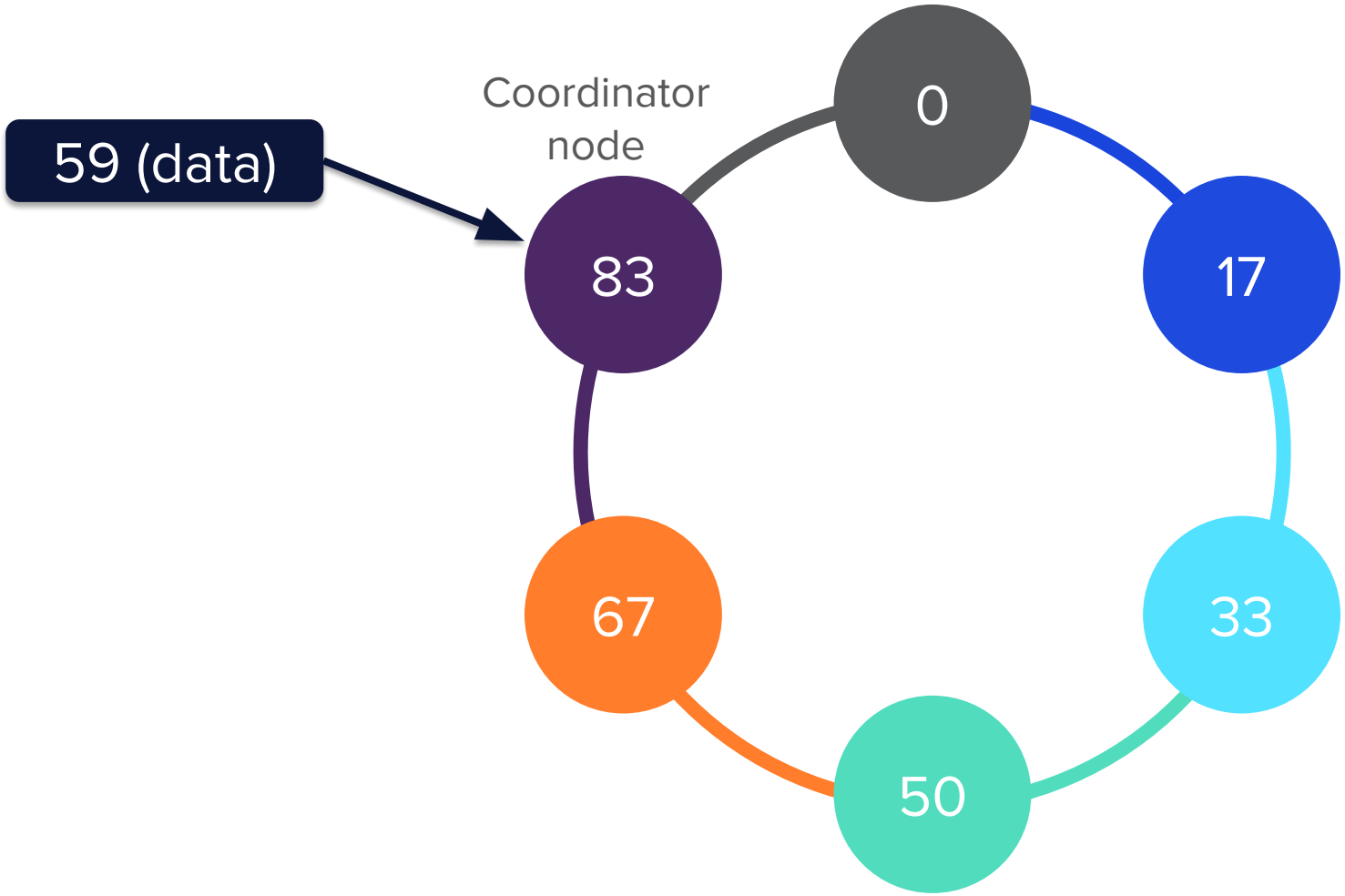
DATASTAX

Token Ring and Data Replication

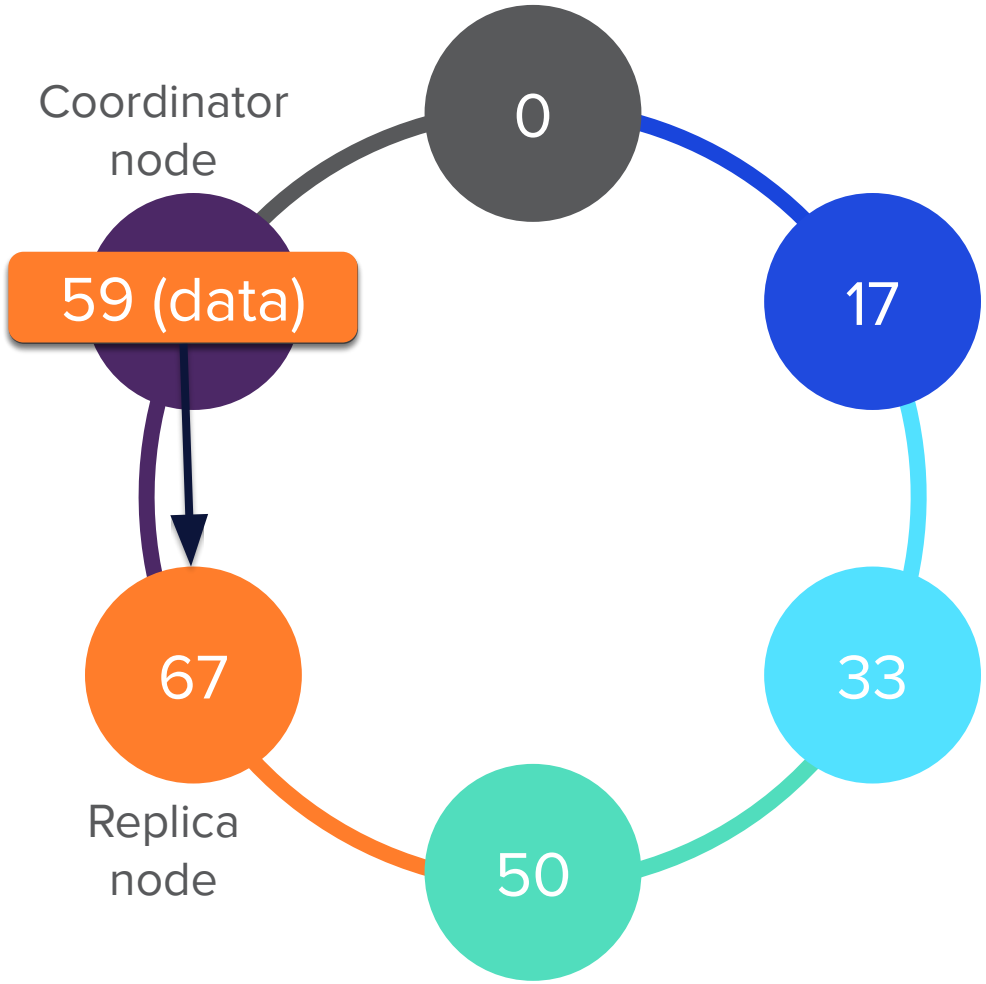
How the Ring Works



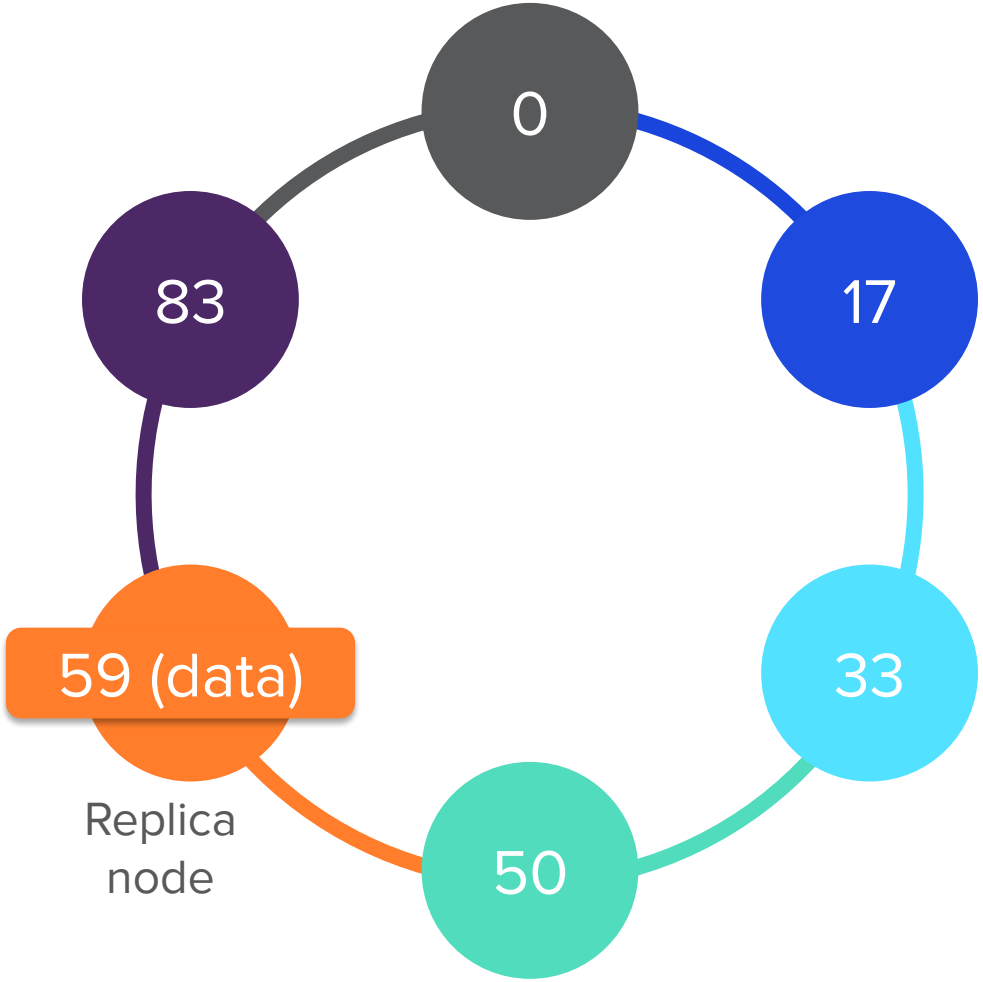
How the Ring Works



How the Ring Works



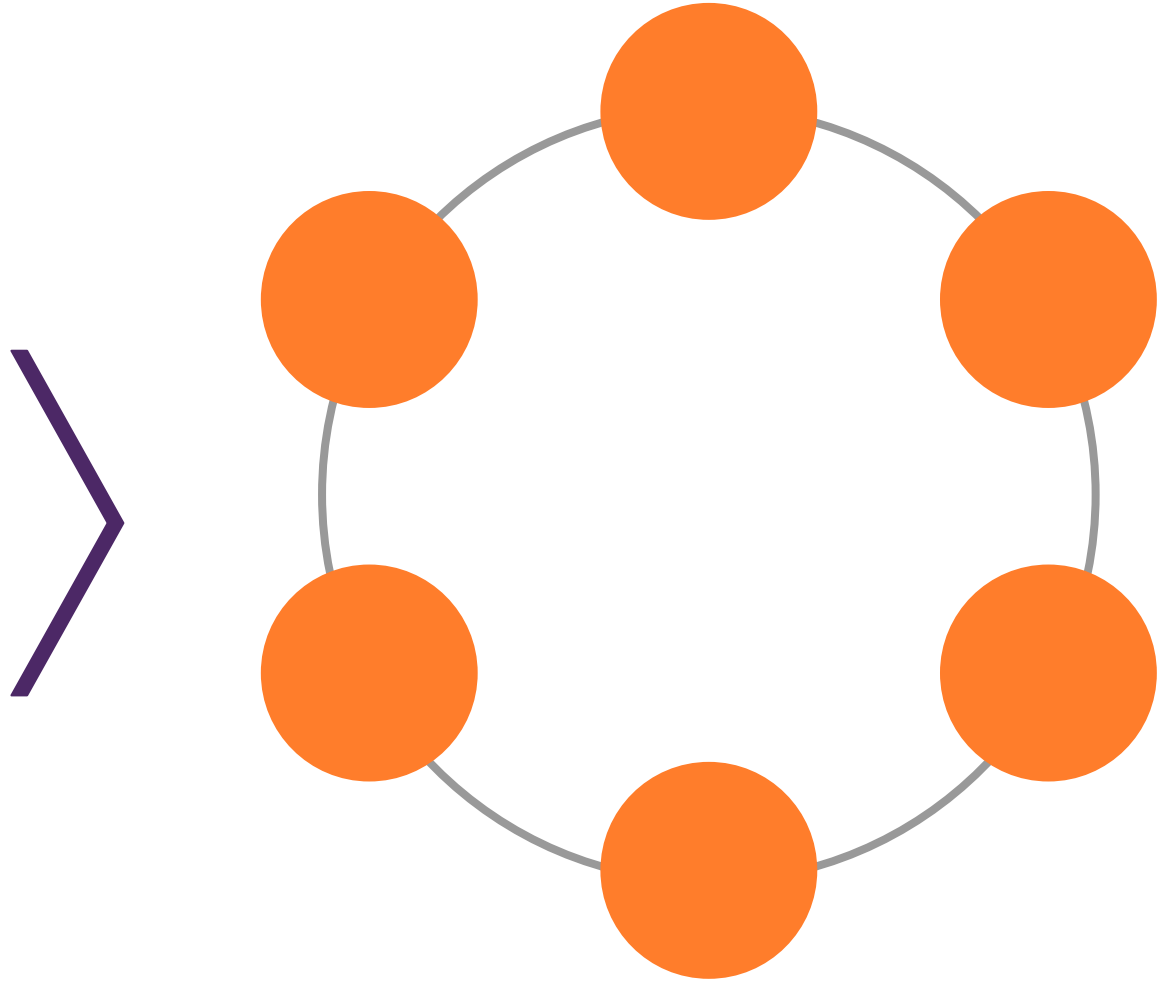
How the Ring Works



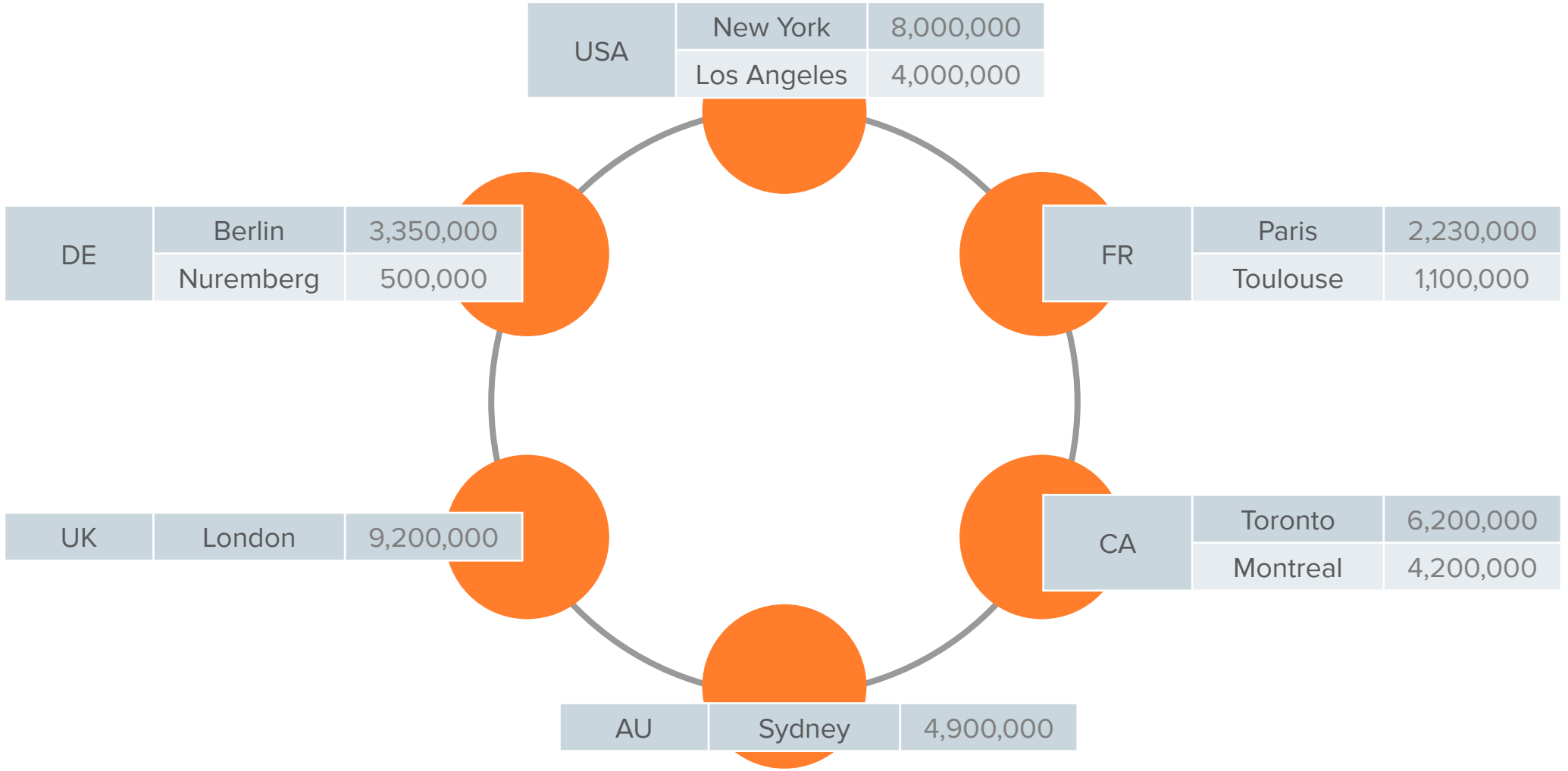
How the Ring Works

- How is the data distributed?

Country	City	Population
USA	New York	8,000,000
	Los Angeles	4,000,000
DE	Berlin	3,350,000
	Nuremberg	500,000
FR	Paris	2,230,000
	Toulouse	1,100,000
CA	Toronto	6,200,000
	Montreal	4,200,000
UK	London	9,200,000
AU	Sydney	4,900,000

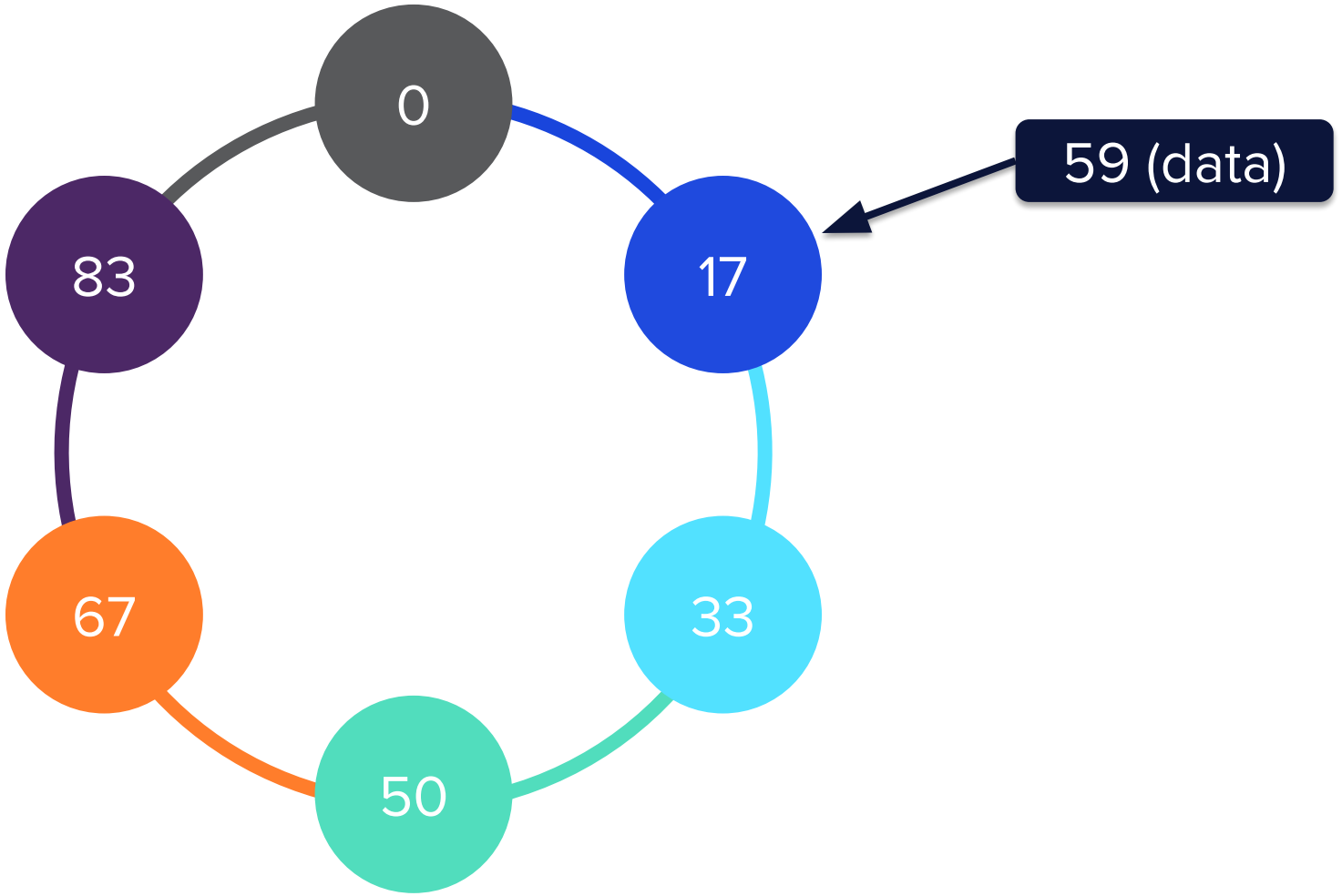


How the Ring Works



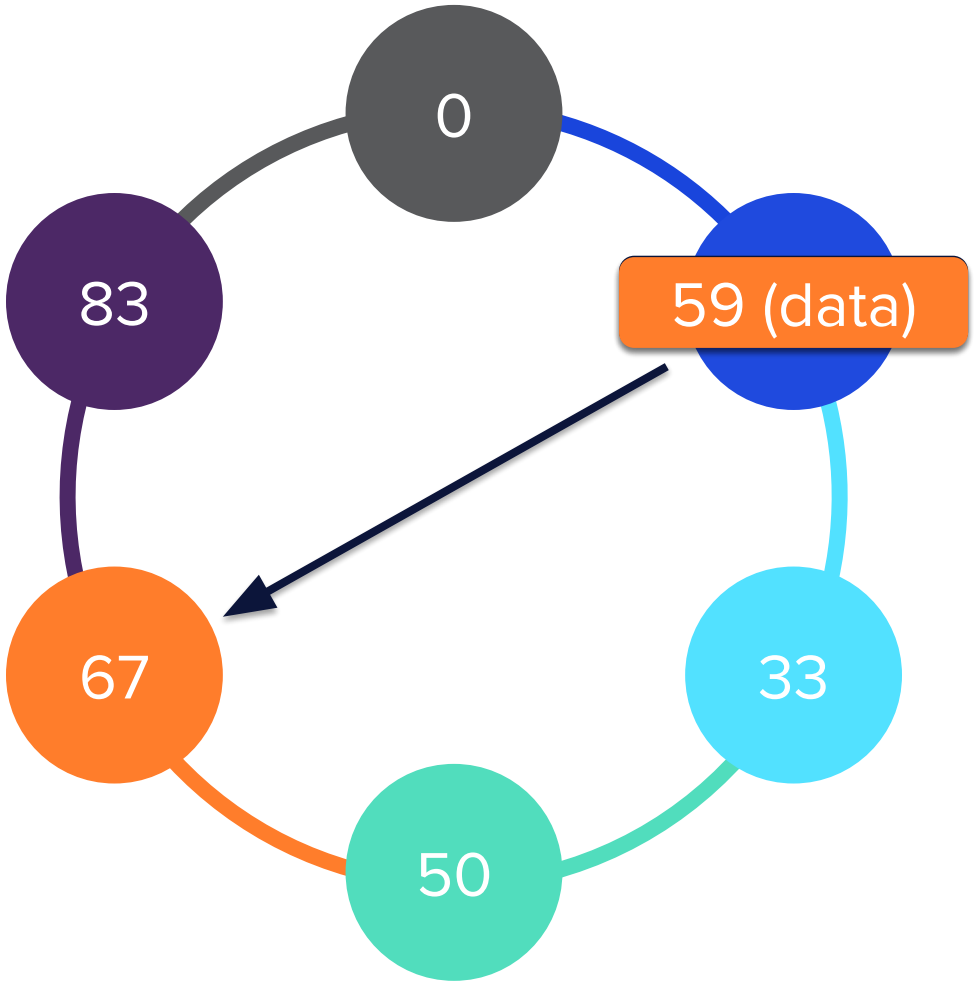
Replication within the Ring

RF = 1



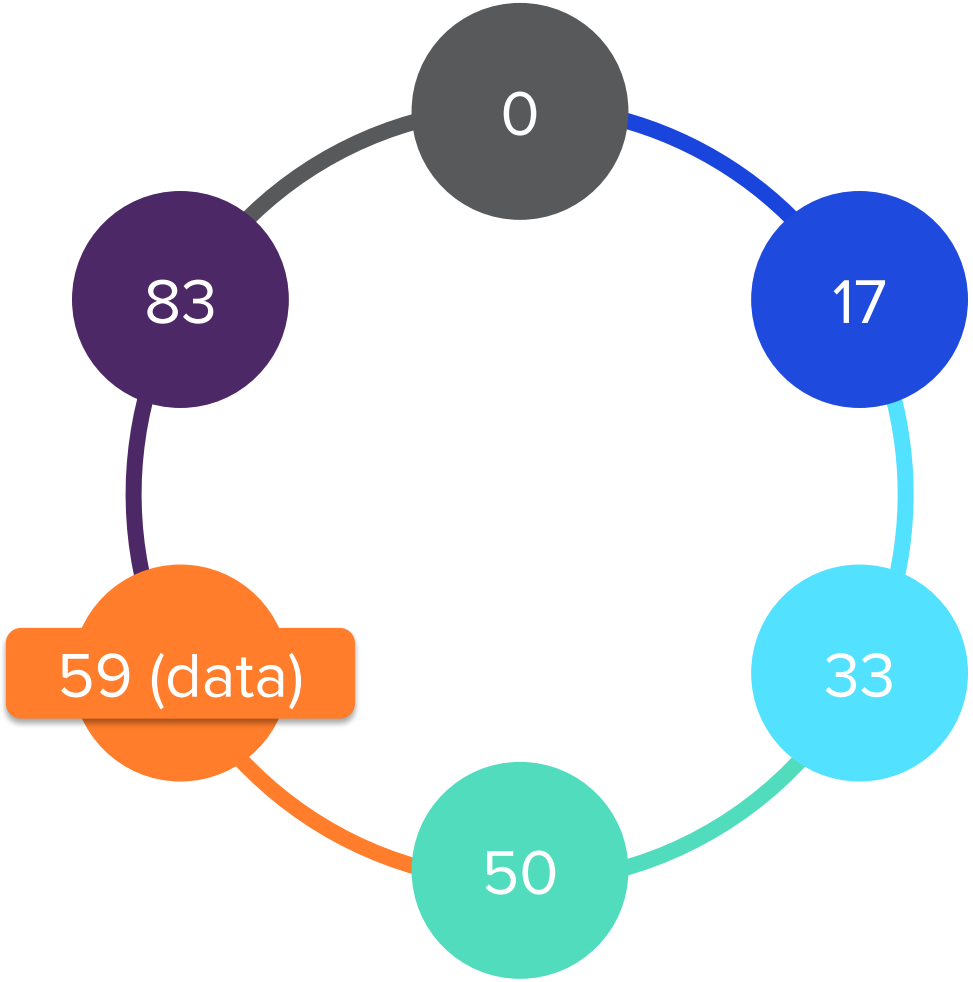
Replication within the Ring

RF = 1



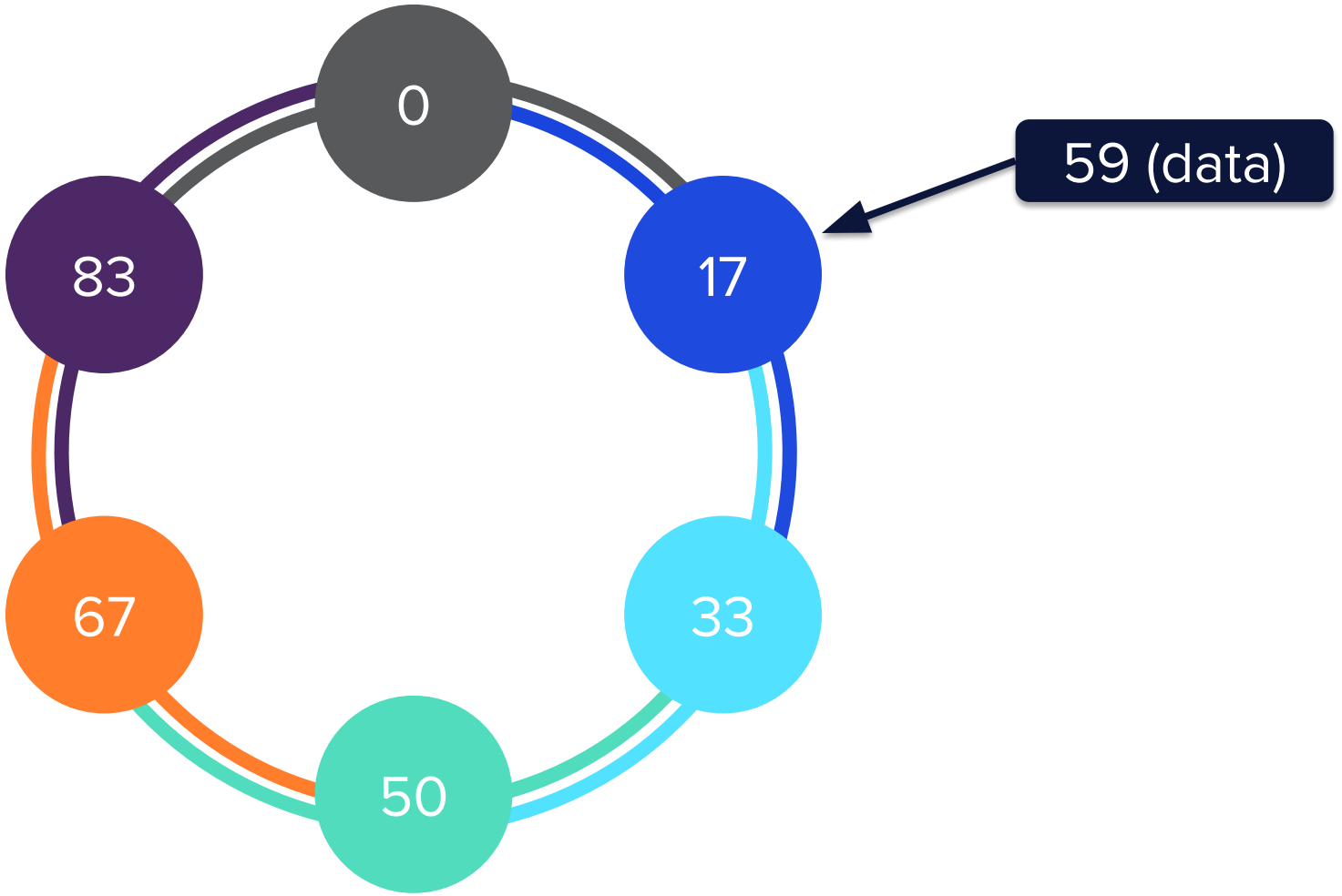
Replication within the Ring

RF = 1



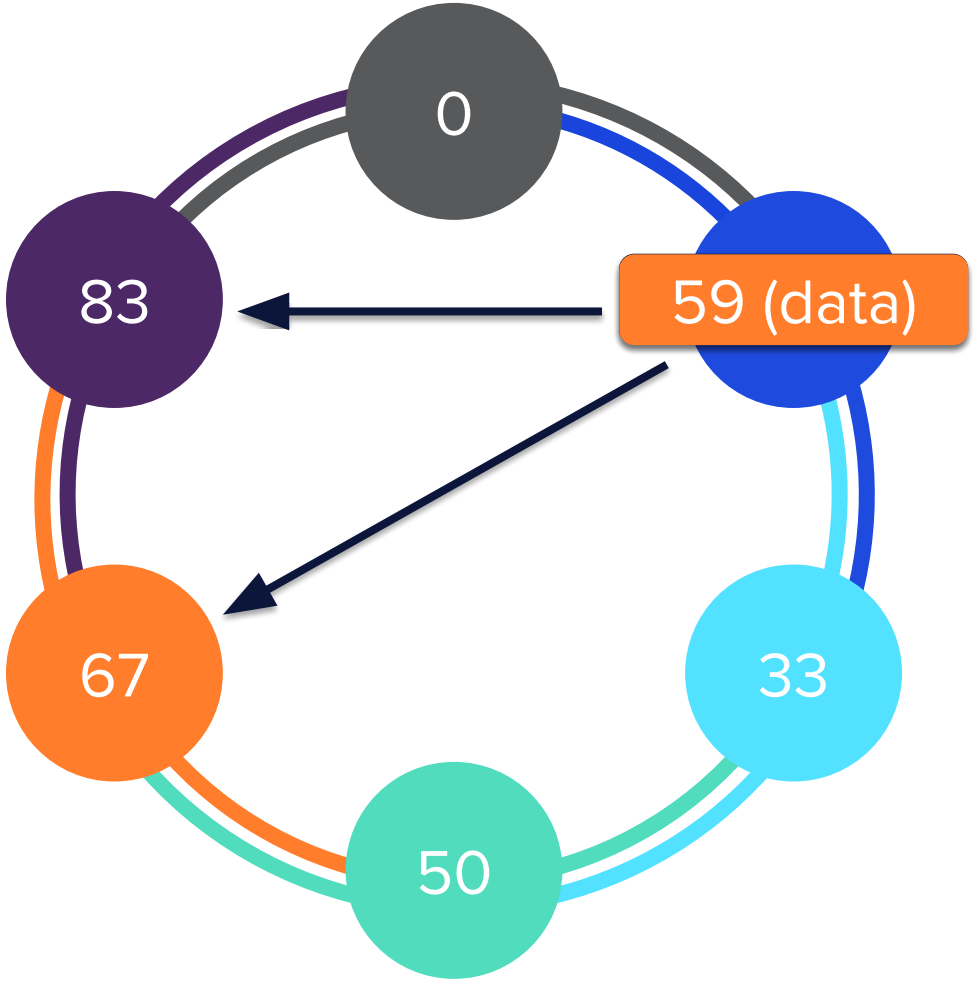
Replication within the Ring

RF = 2



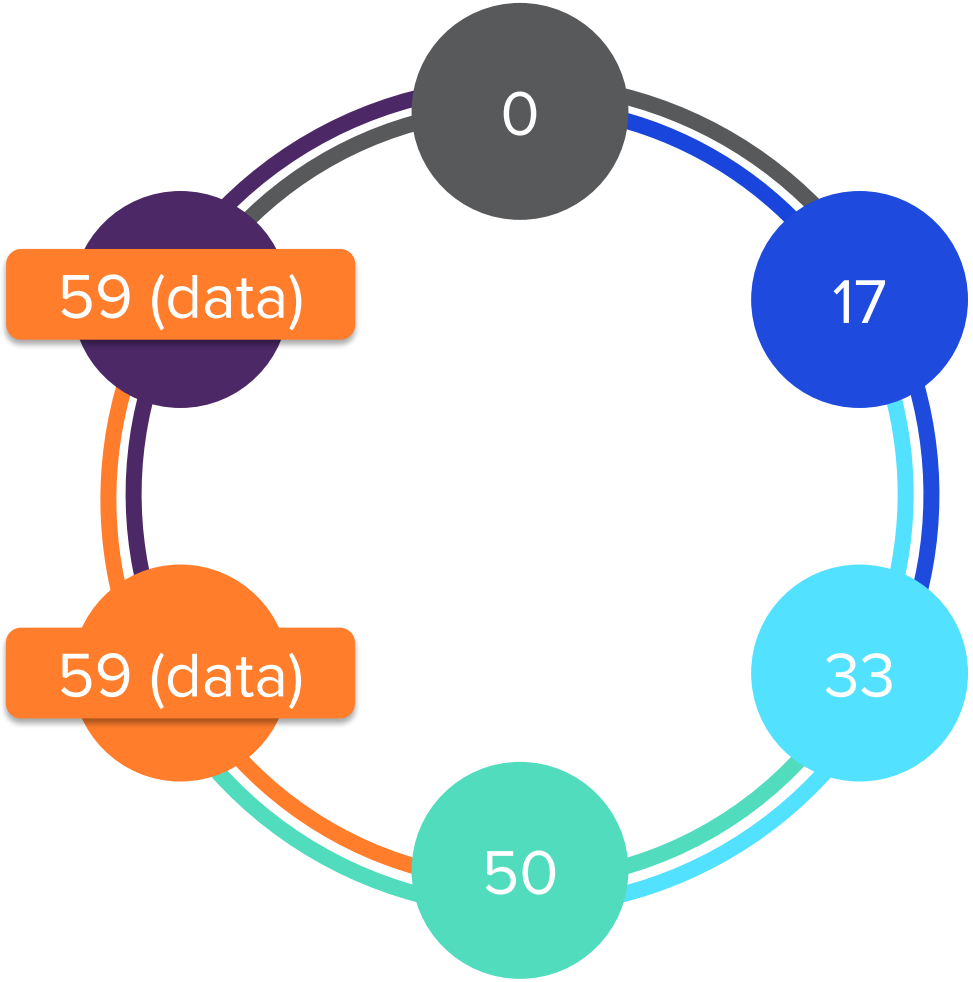
Replication within the Ring

RF = 2



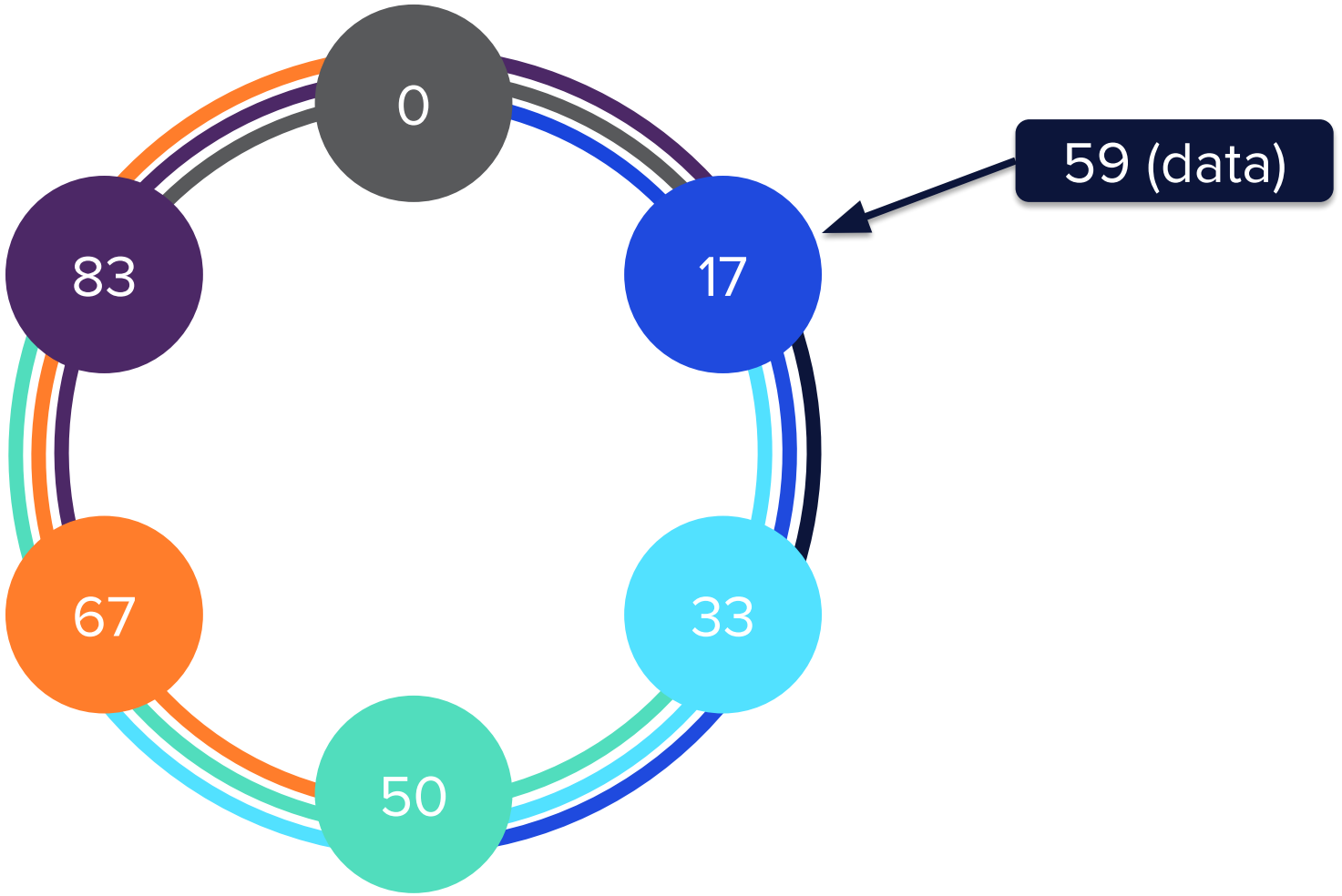
Replication within the Ring

RF = 2



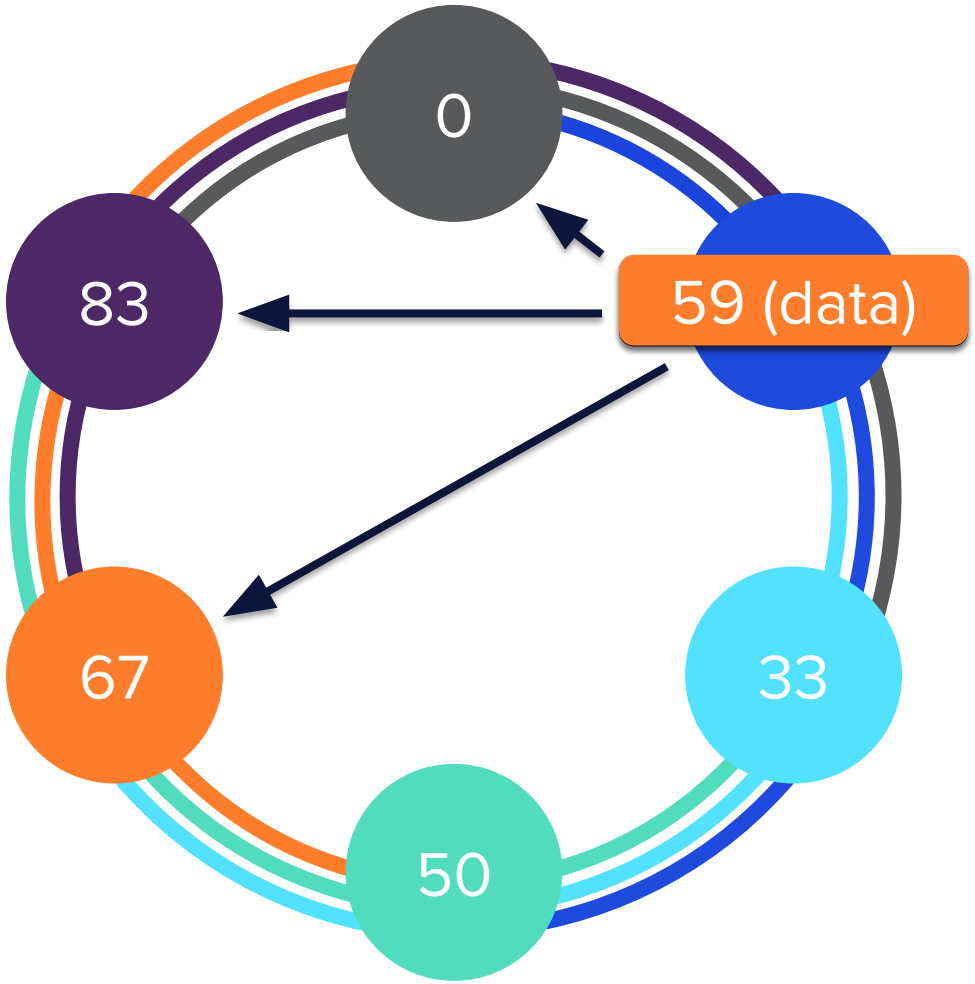
Replication within the Ring

RF = 3



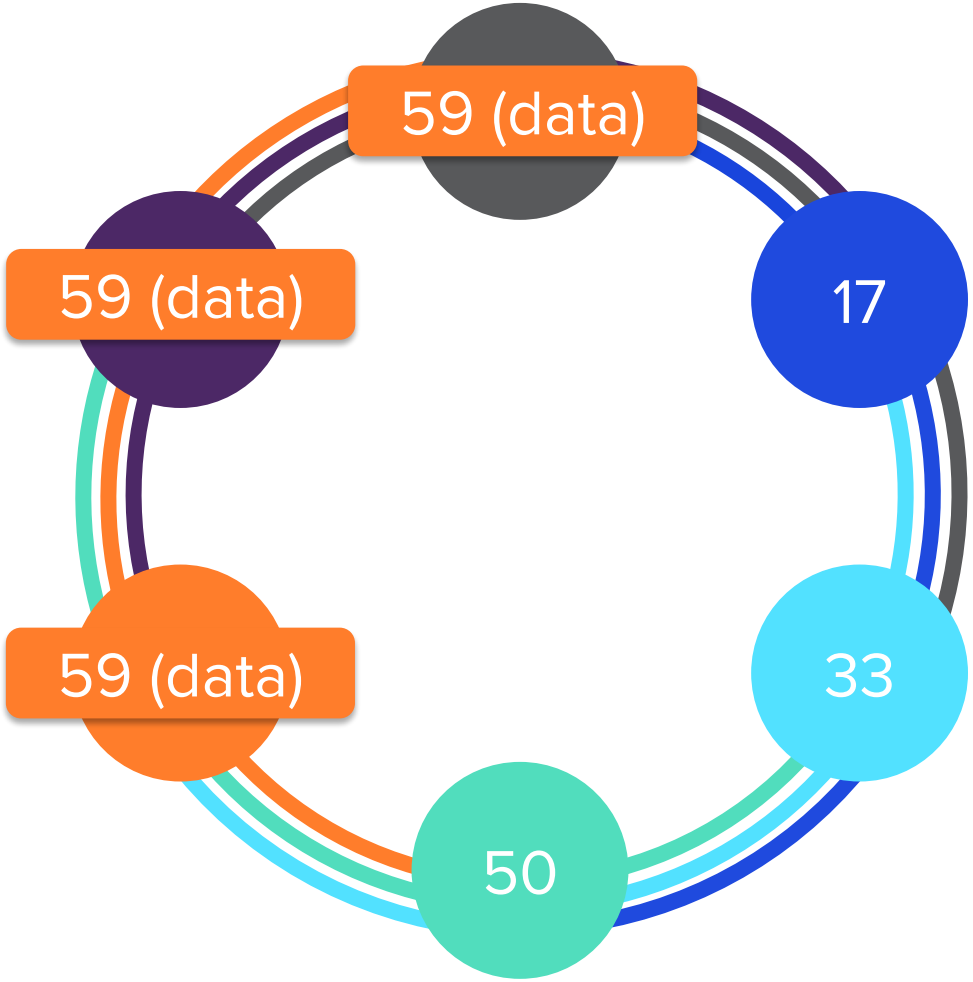
Replication within the Ring

RF = 3



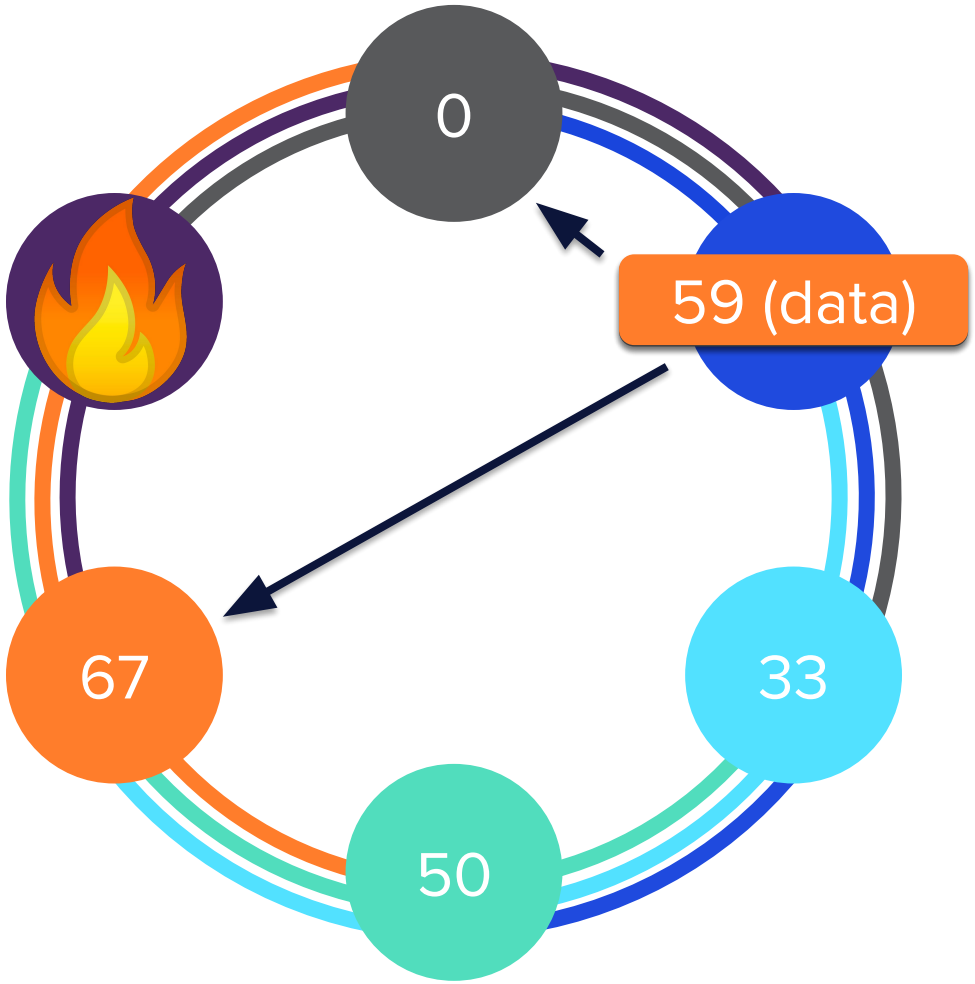
Replication within the Ring

RF = 3



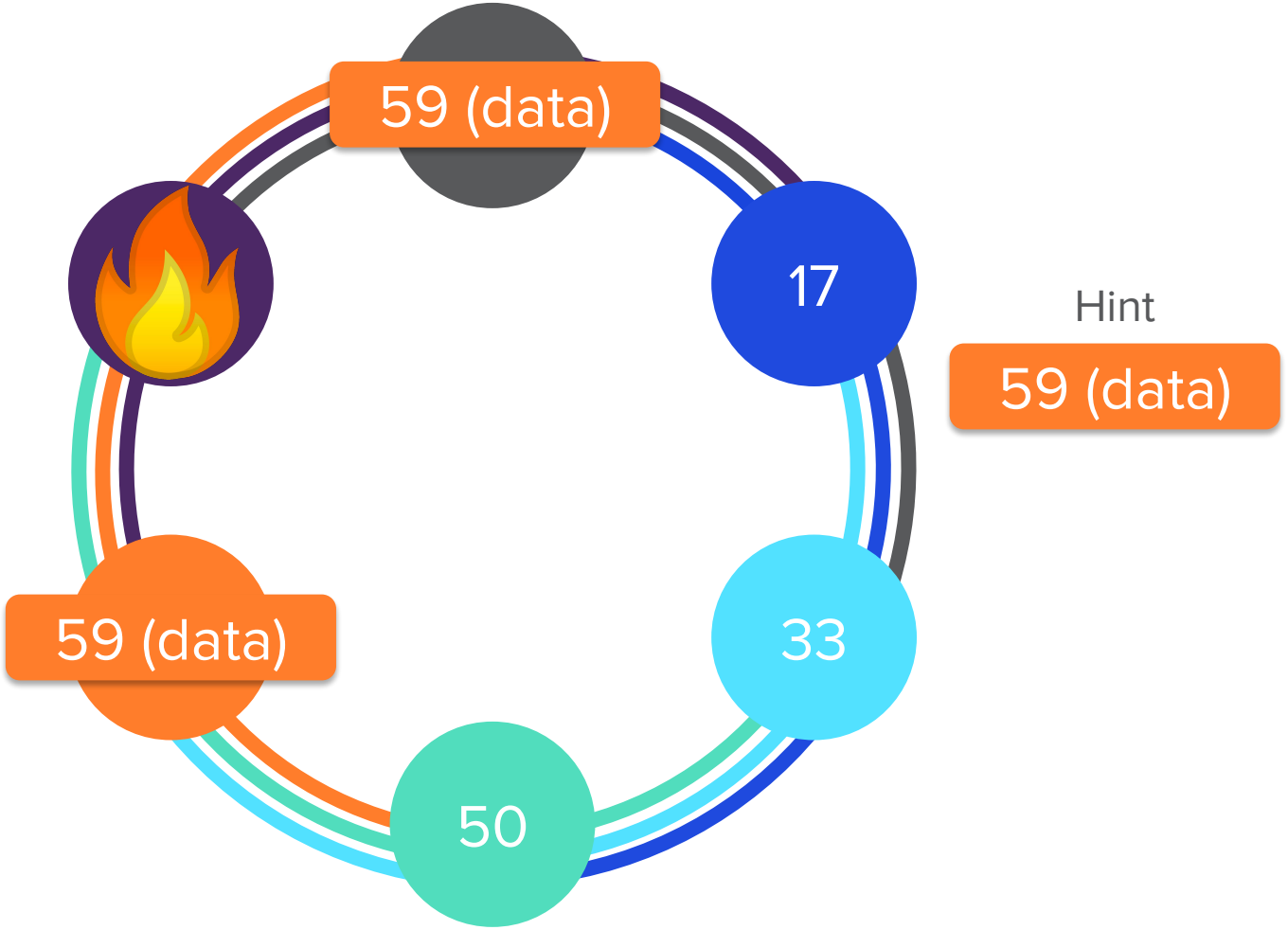
Node Failure

RF = 3



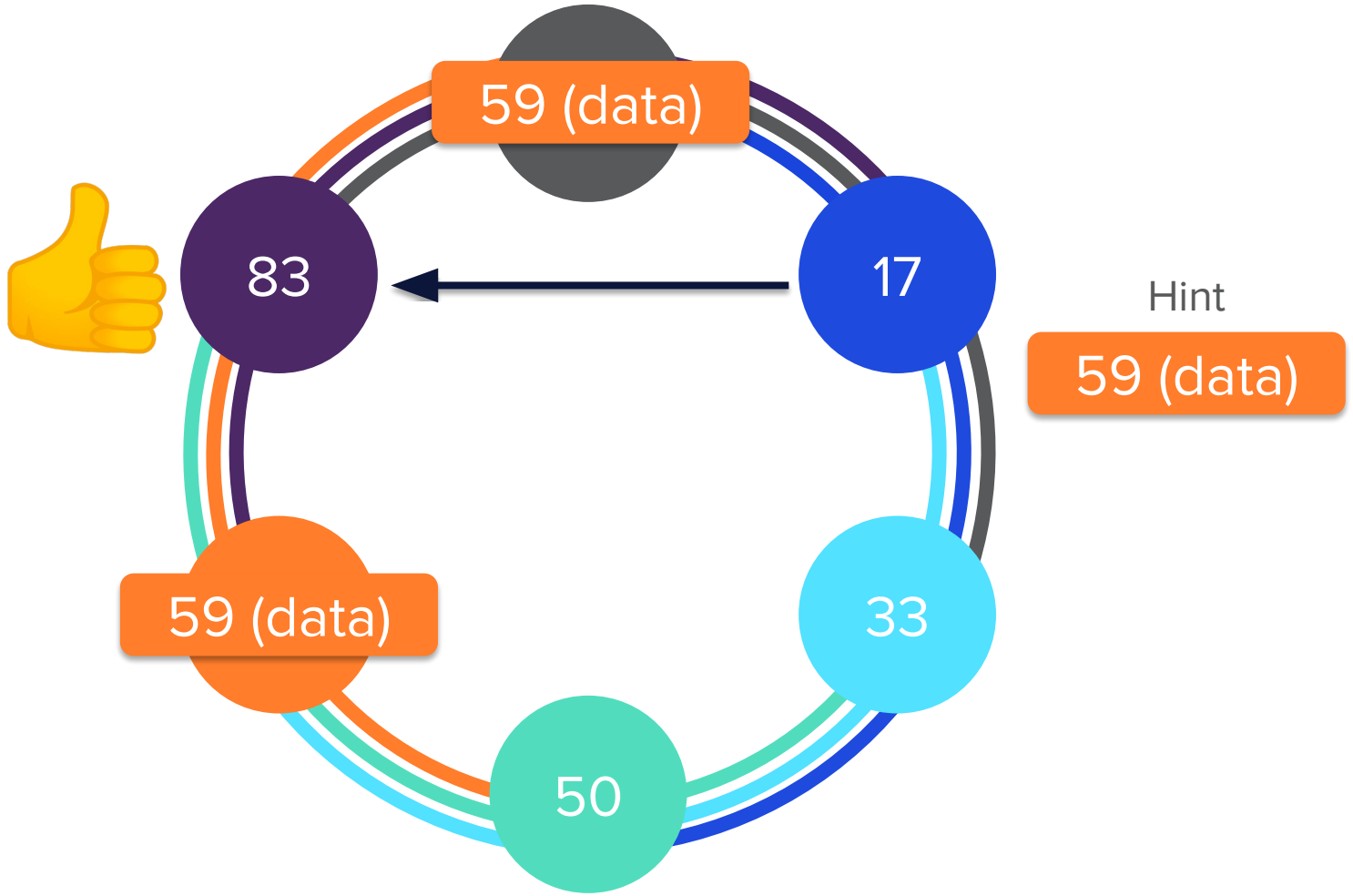
Node Failure

RF = 3



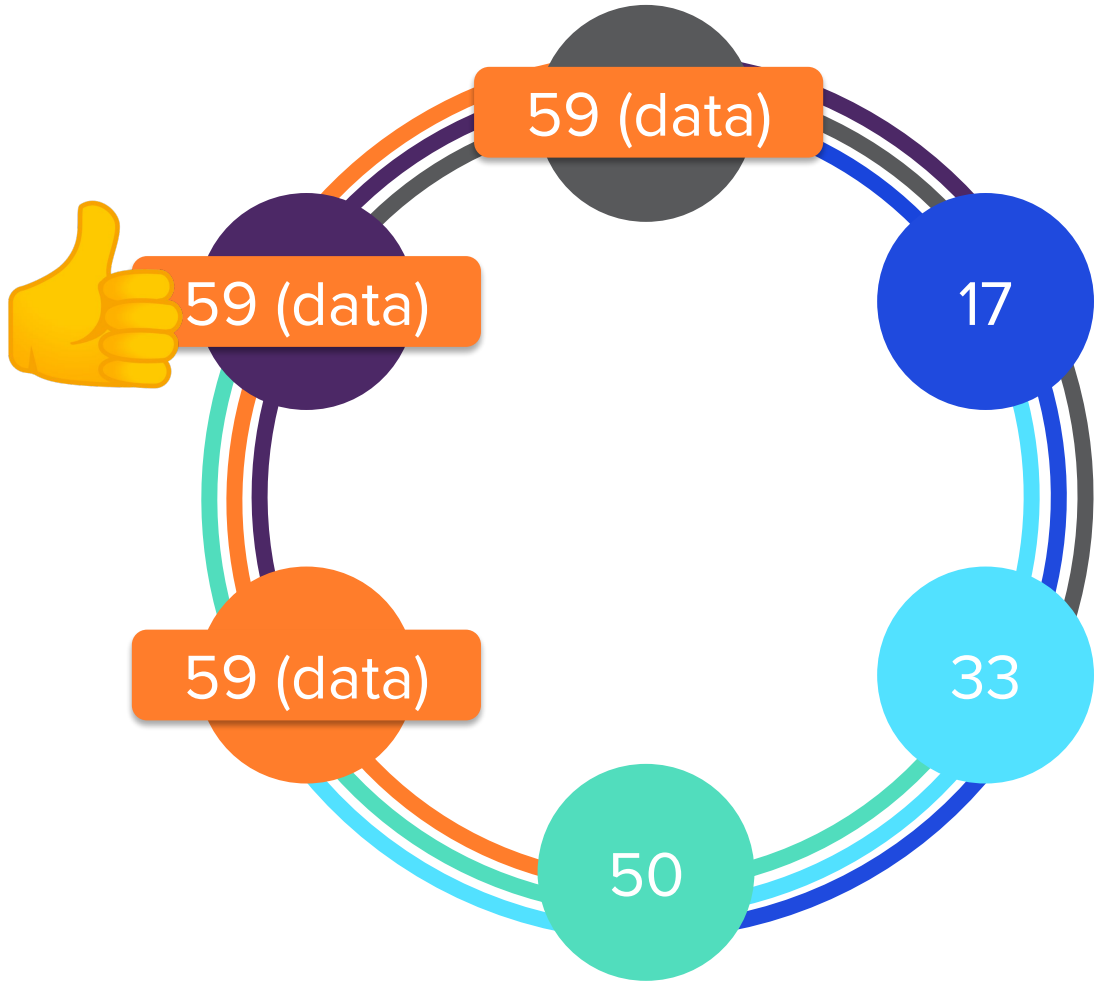
Node Failure

RF = 3



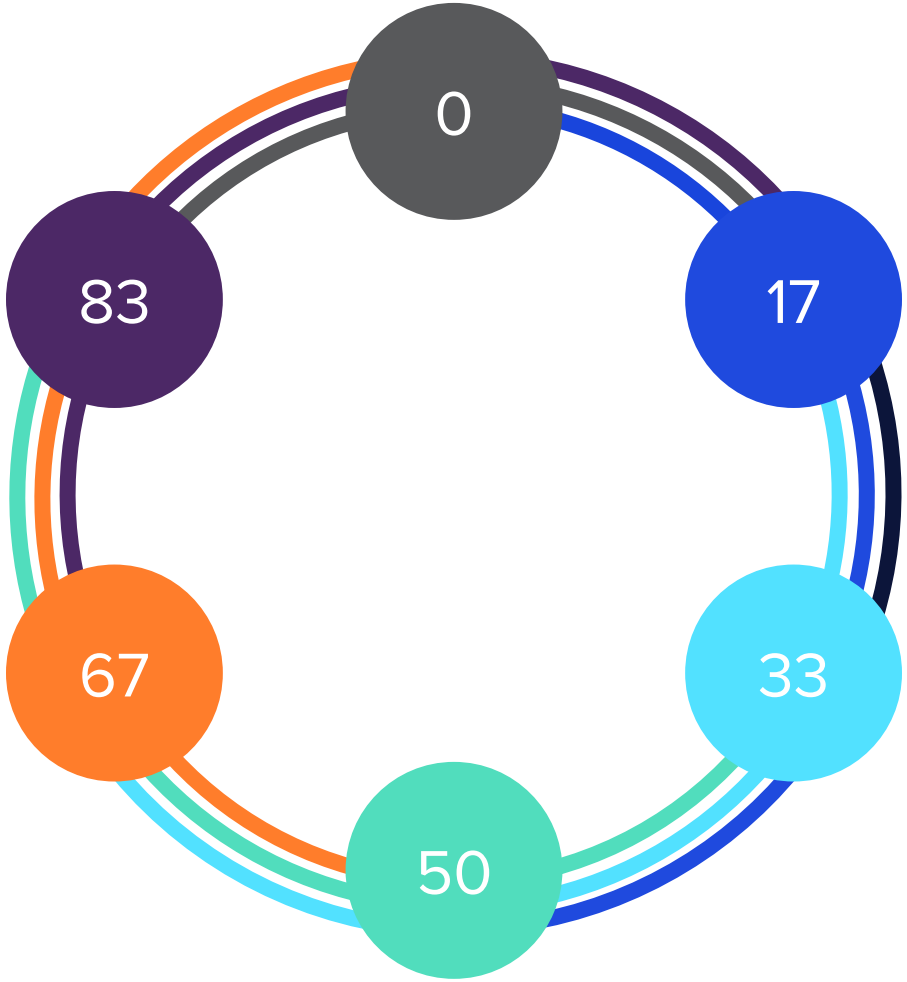
Node Failure – Recovered!

RF = 3



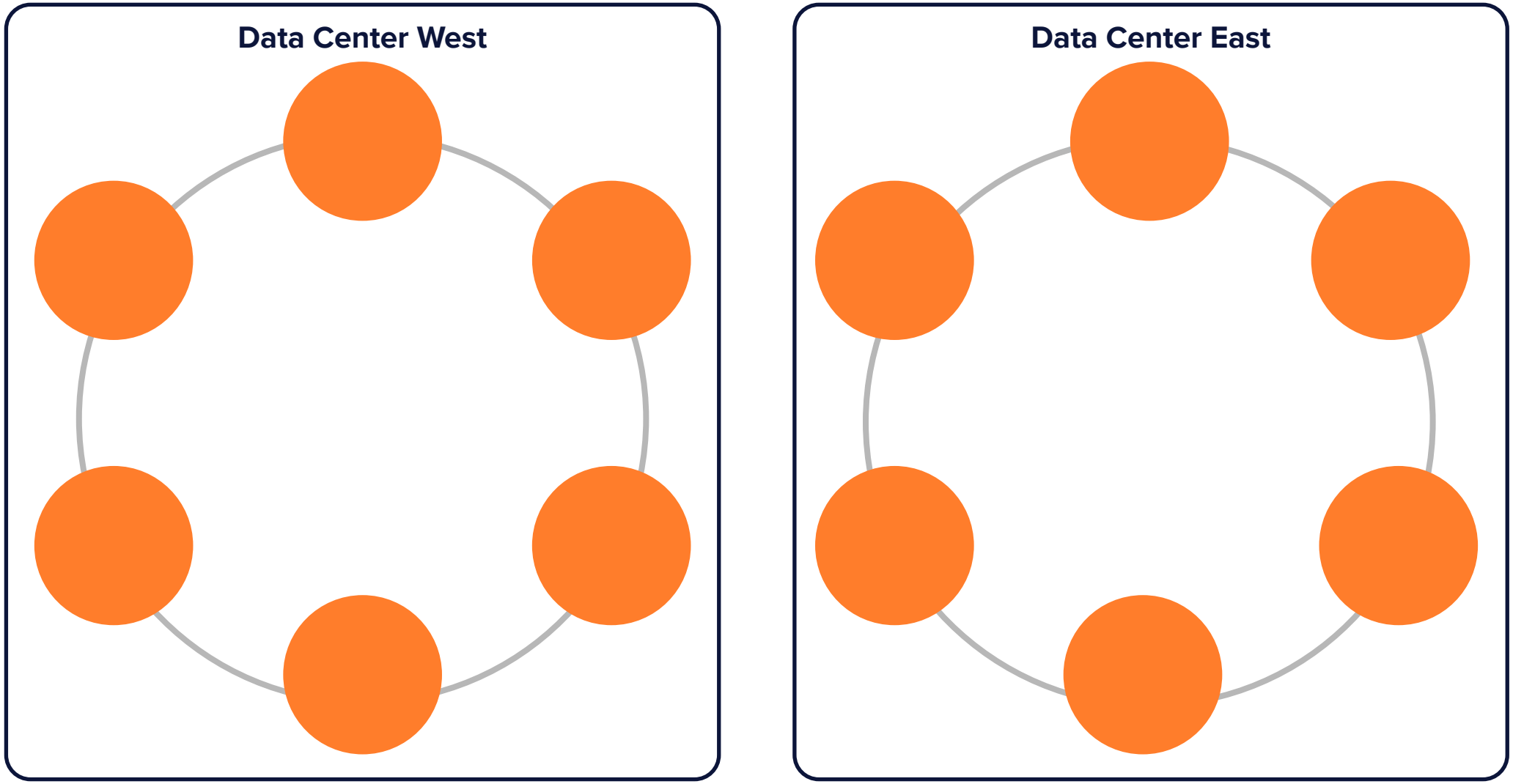
Multi-Data Center Replication

Data Center West

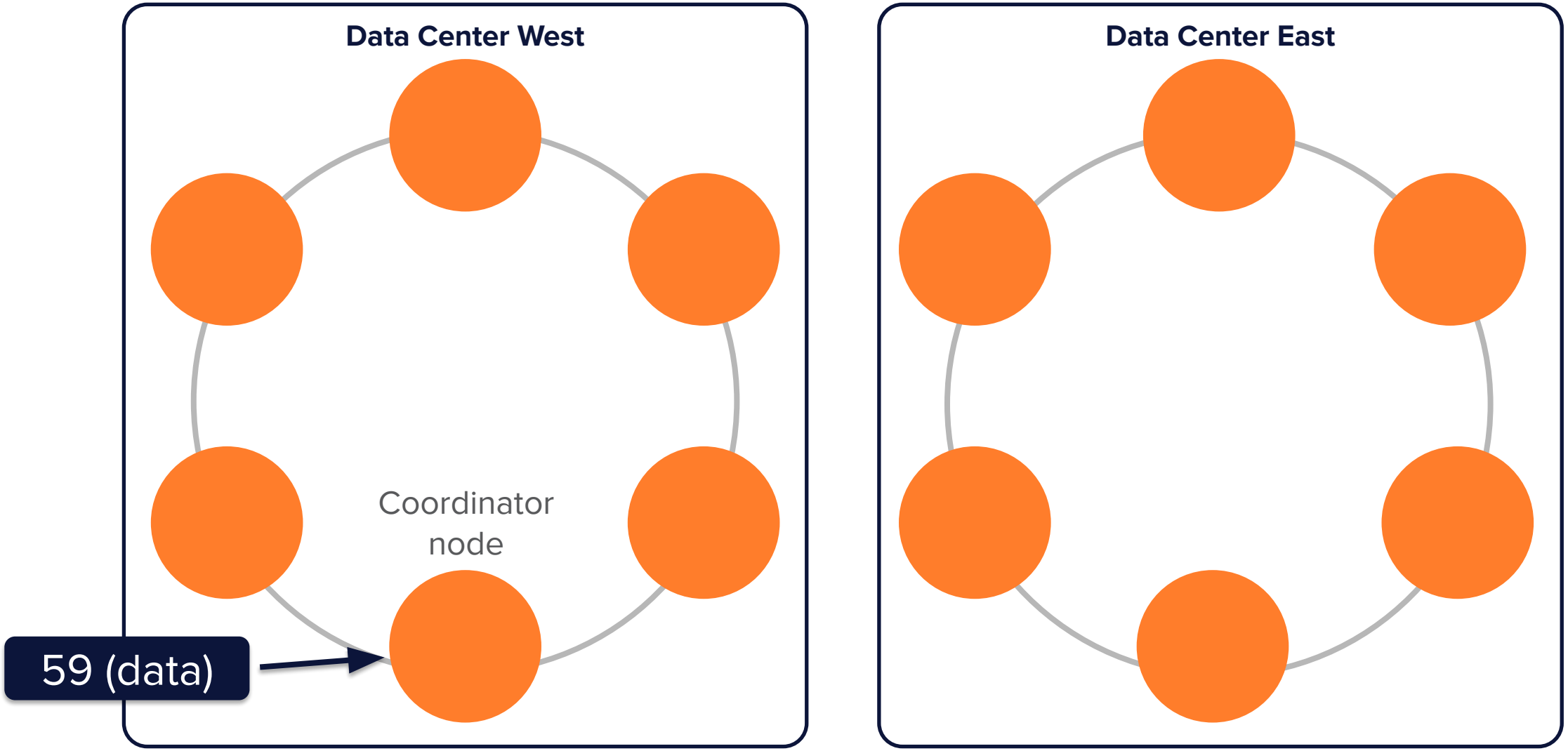


Data Center East

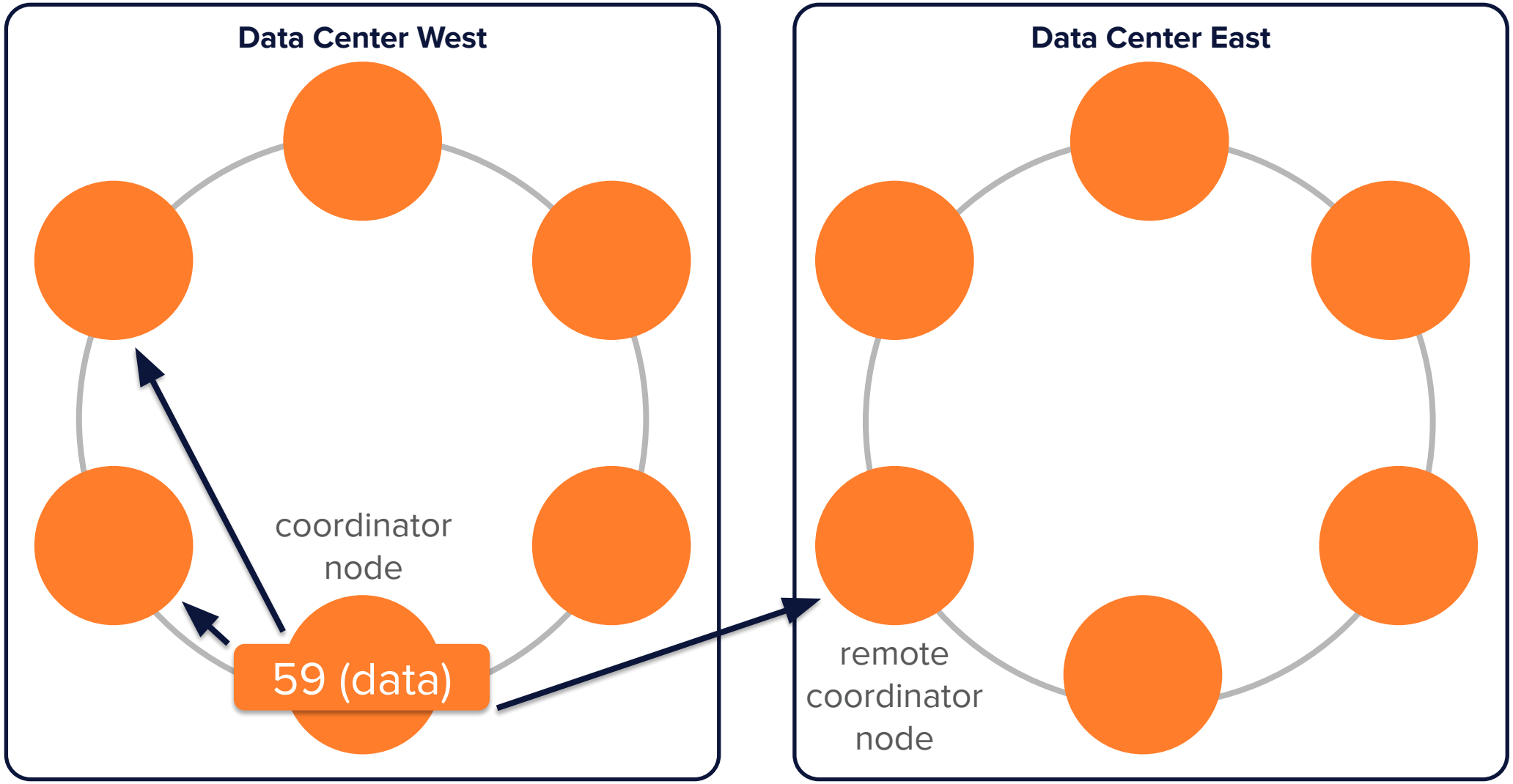
Multi-Data Center Replication (RF=2 in each DC)



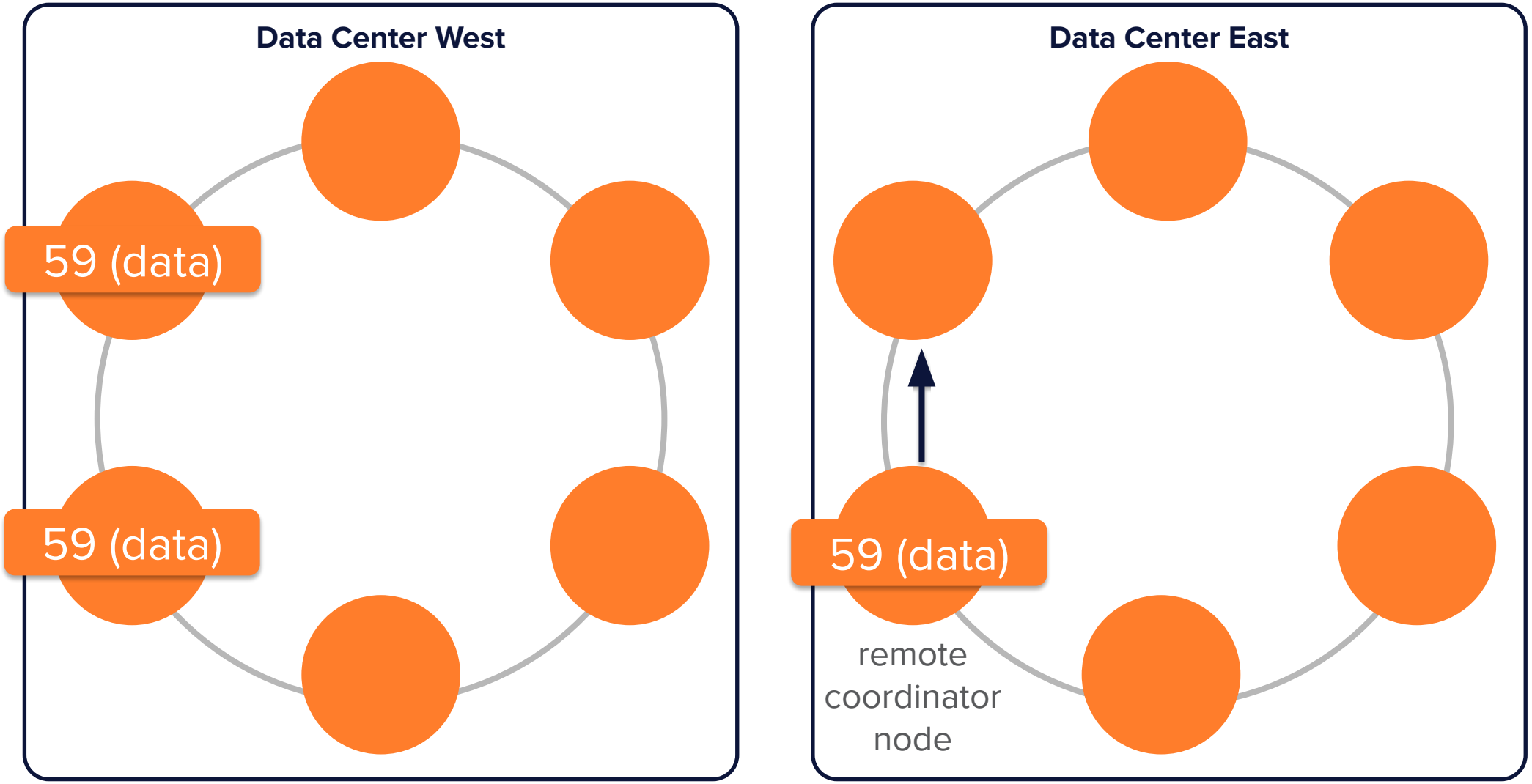
Multi-Data Center Replication (RF=2 in each DC)



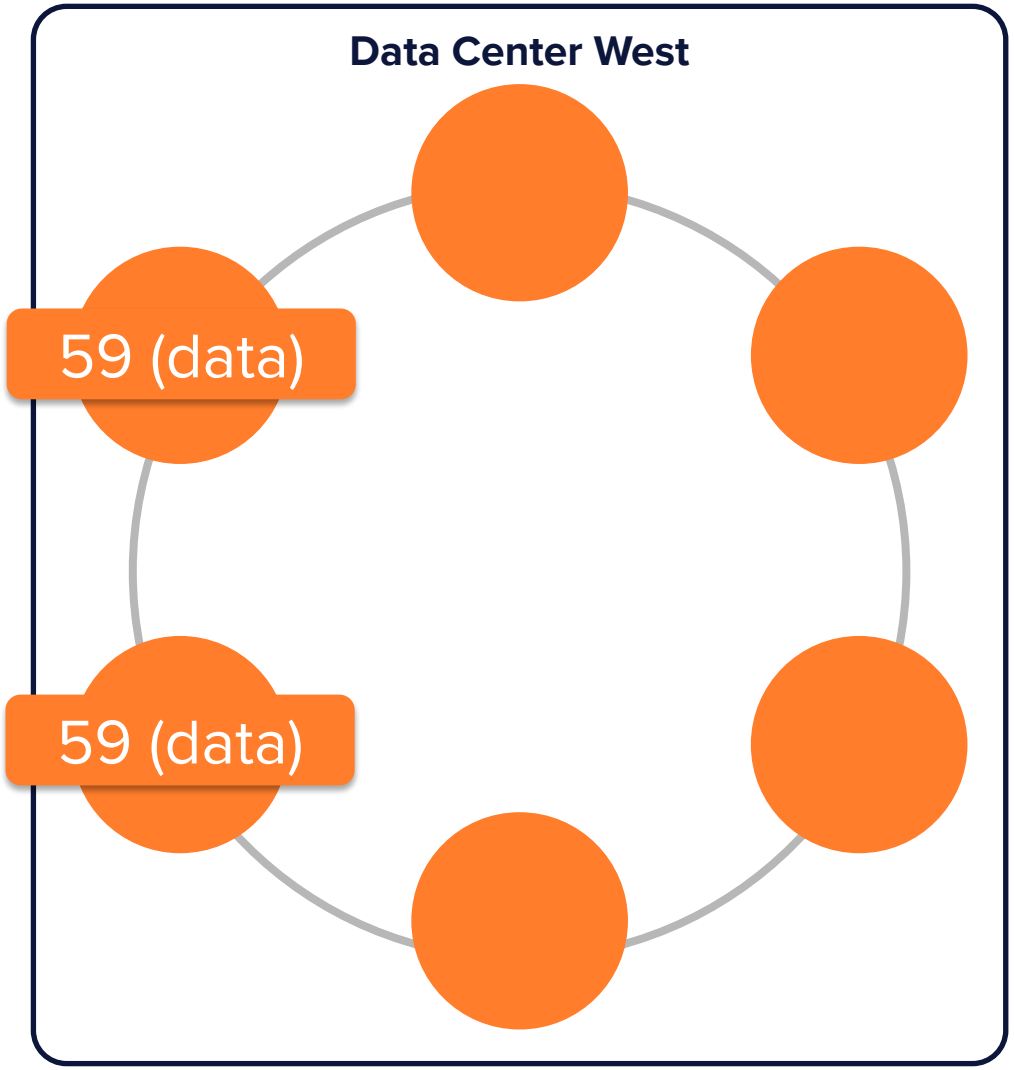
Multi-Data Center Replication (RF=2 in each DC)

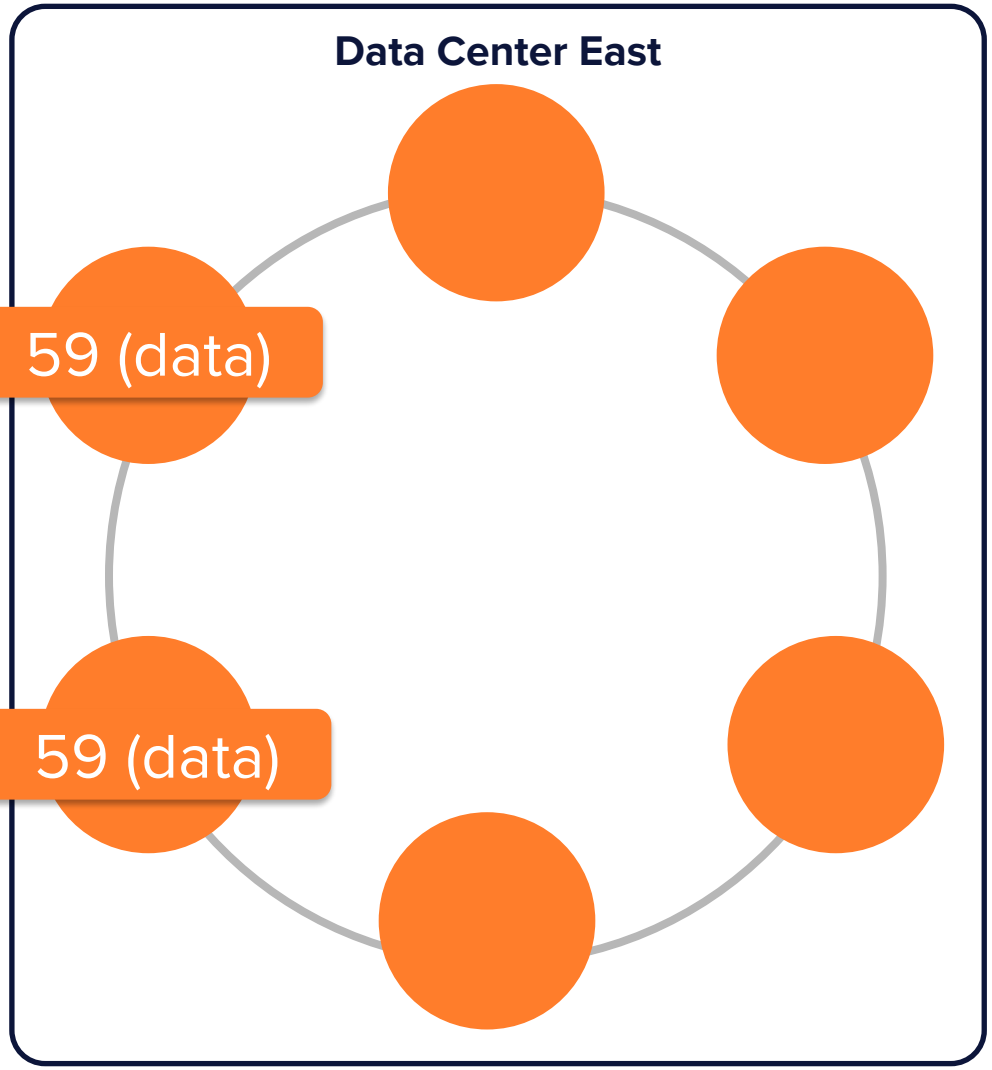


Multi-Data Center Replication (RF=2 in each DC)

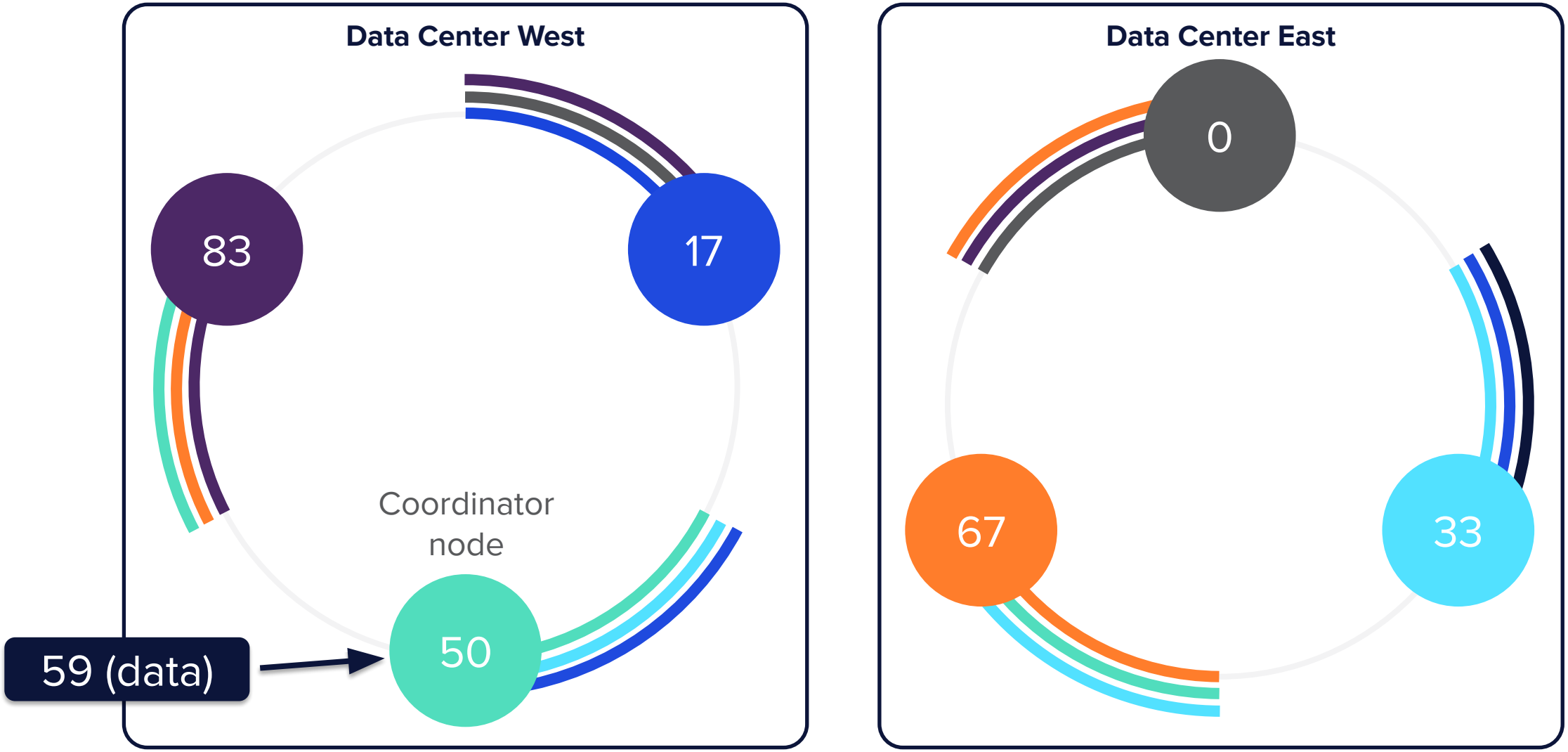


Multi-Data Center Replication (RF=2 in each DC)

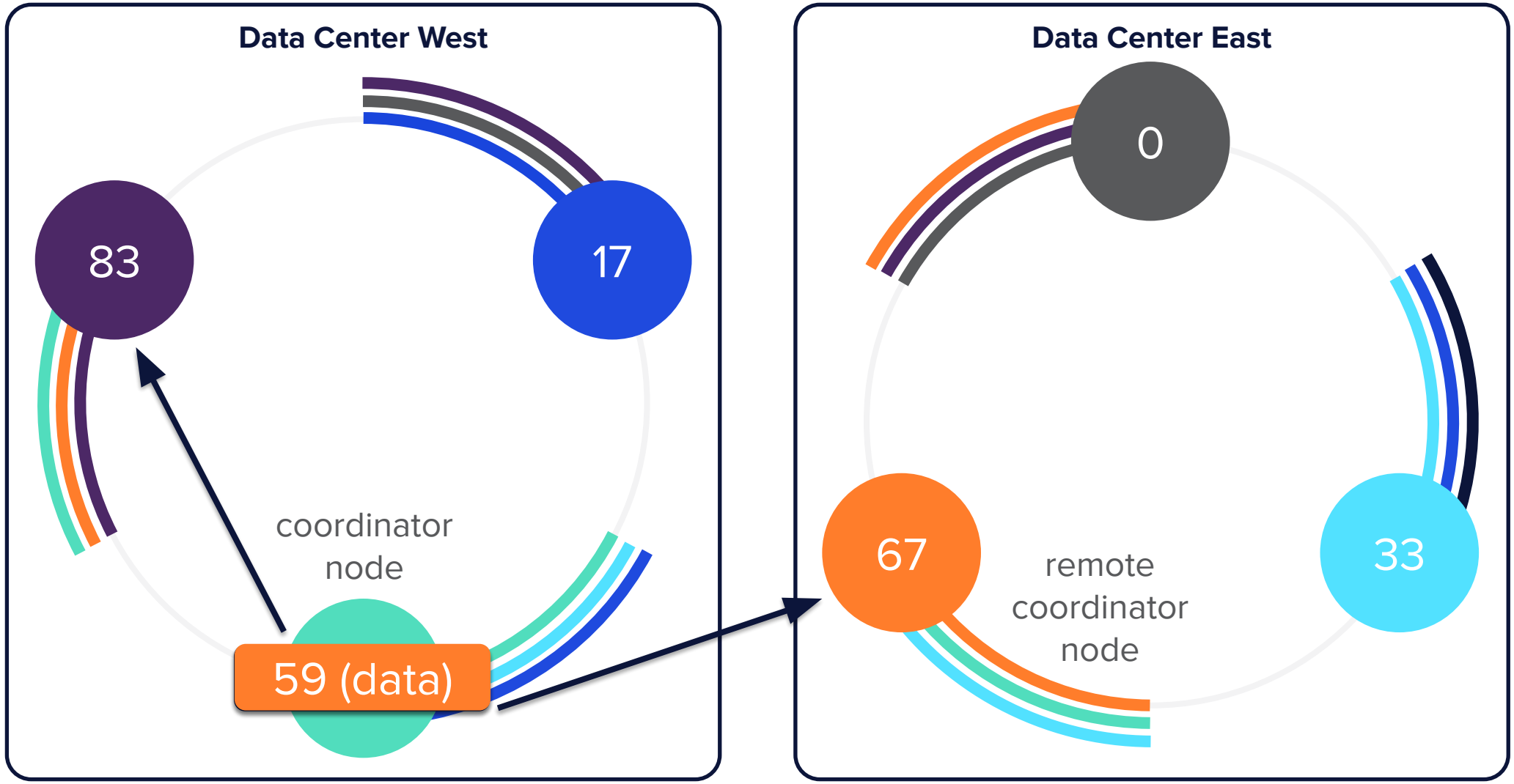




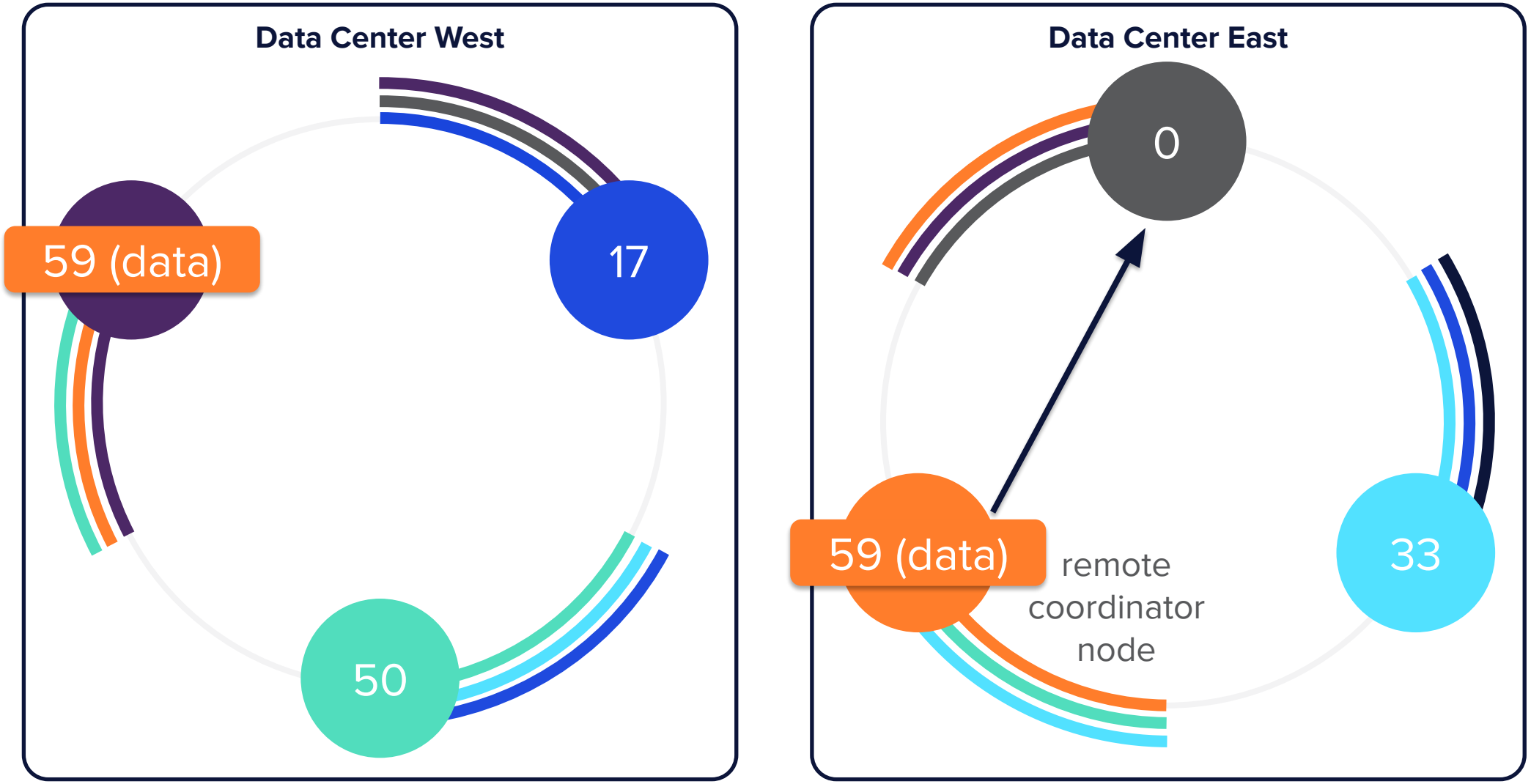
Multi-Data Center Replication



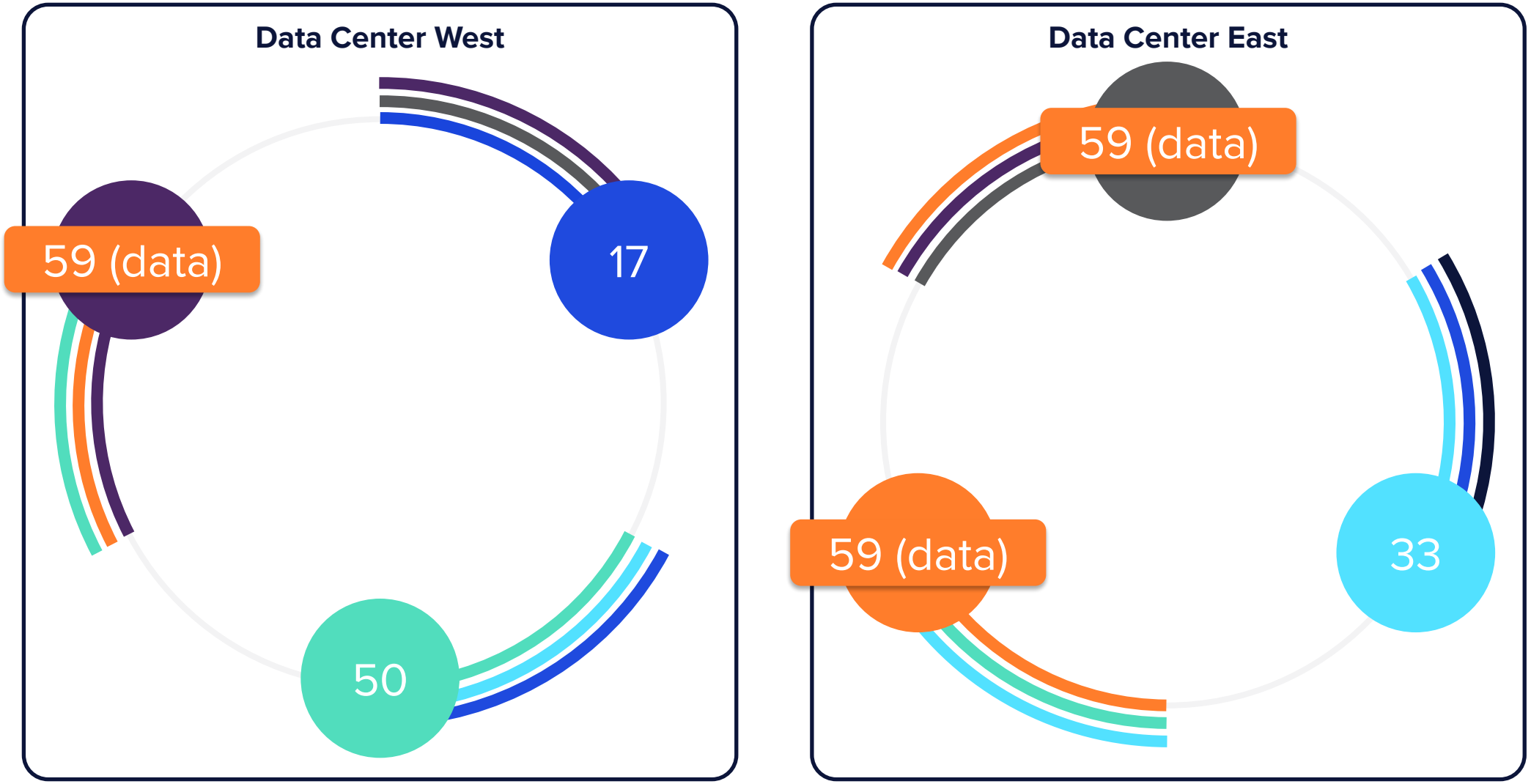
Multi-Data Center Replication



Multi-Data Center Replication



Multi-Data Center Replication

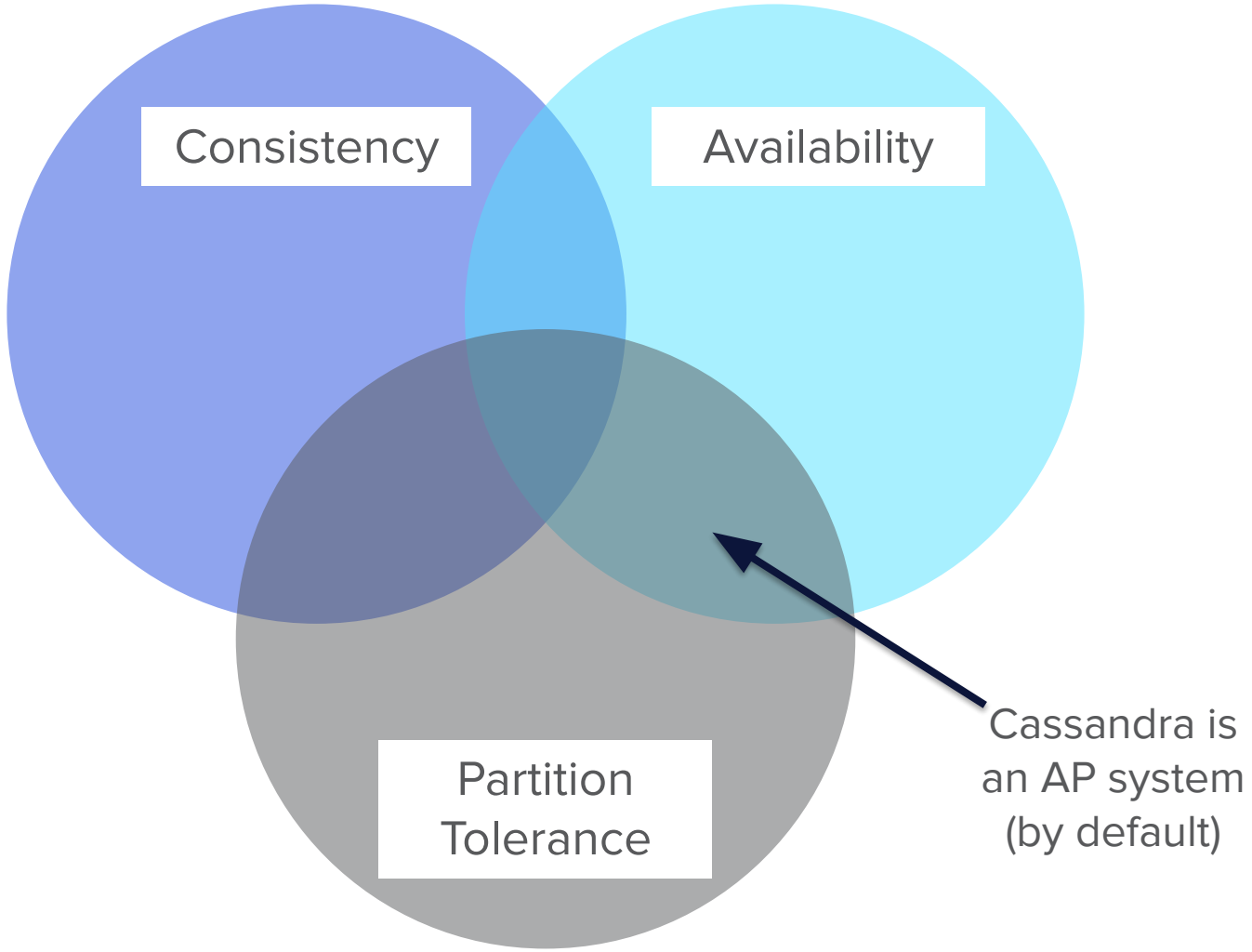


DataStax Developer Day

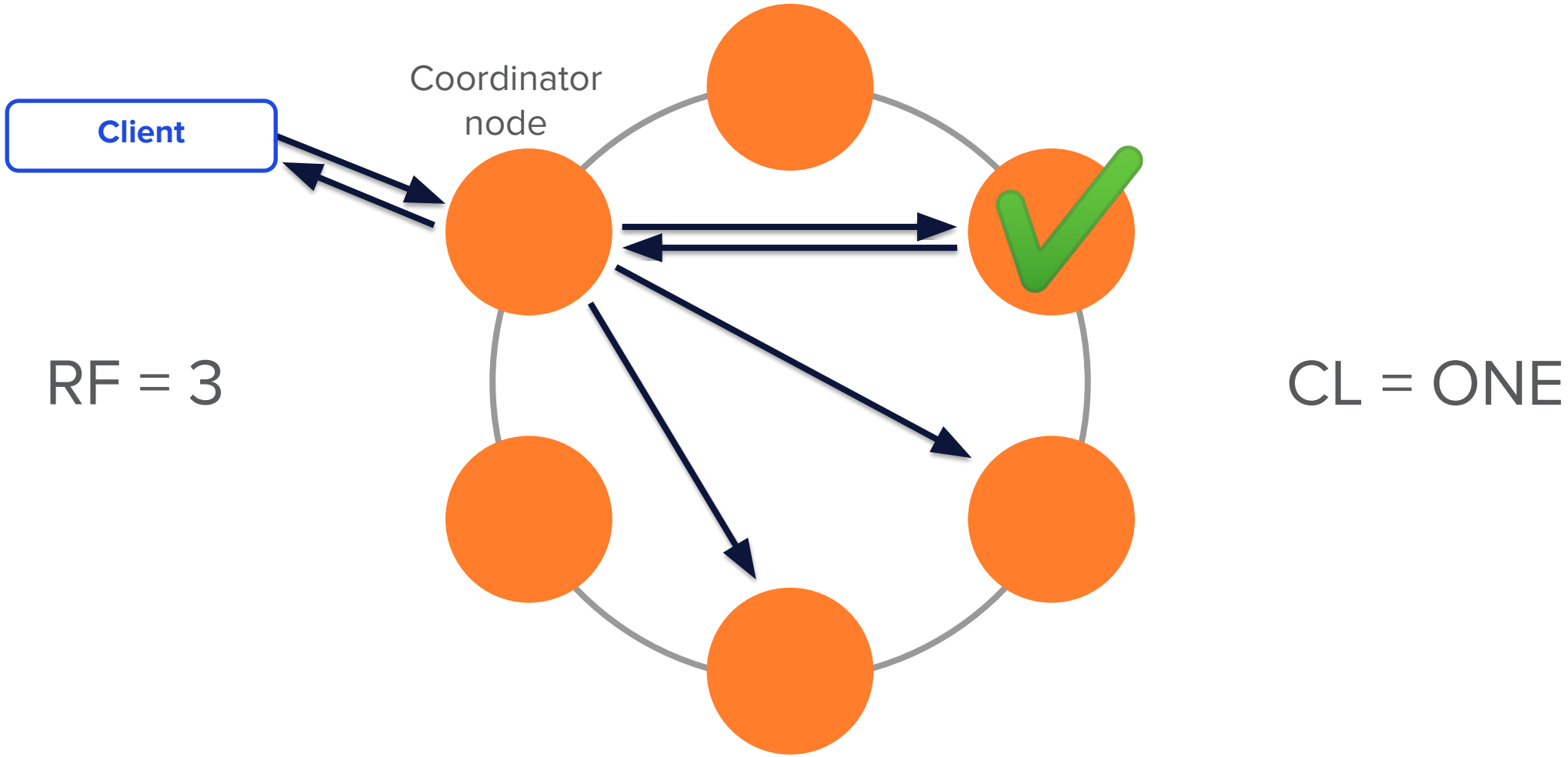
DATASTAX

Cassandra's Consistency

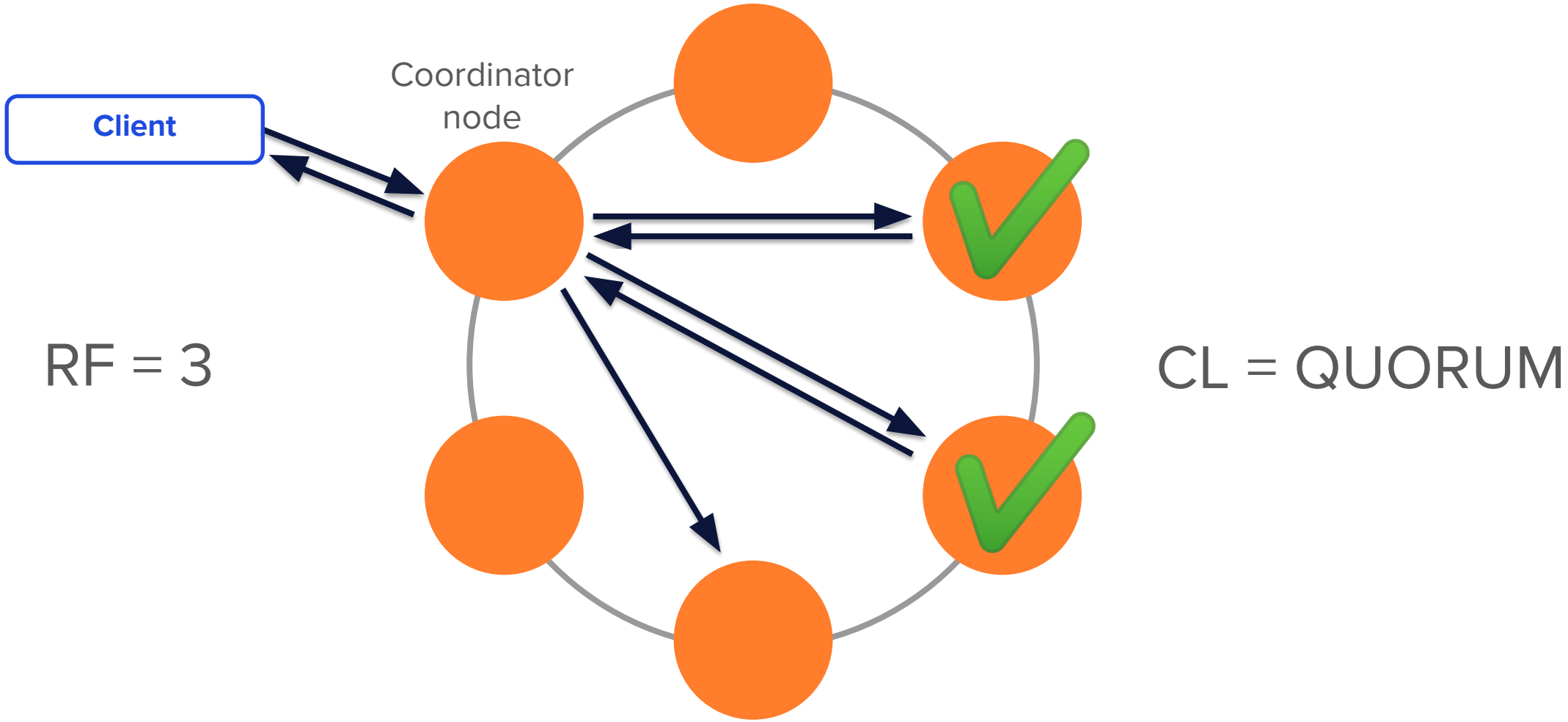
CAP Theorem


Consistency
Availability
Partition Tolerance
Cassandra is an AP system (by default)

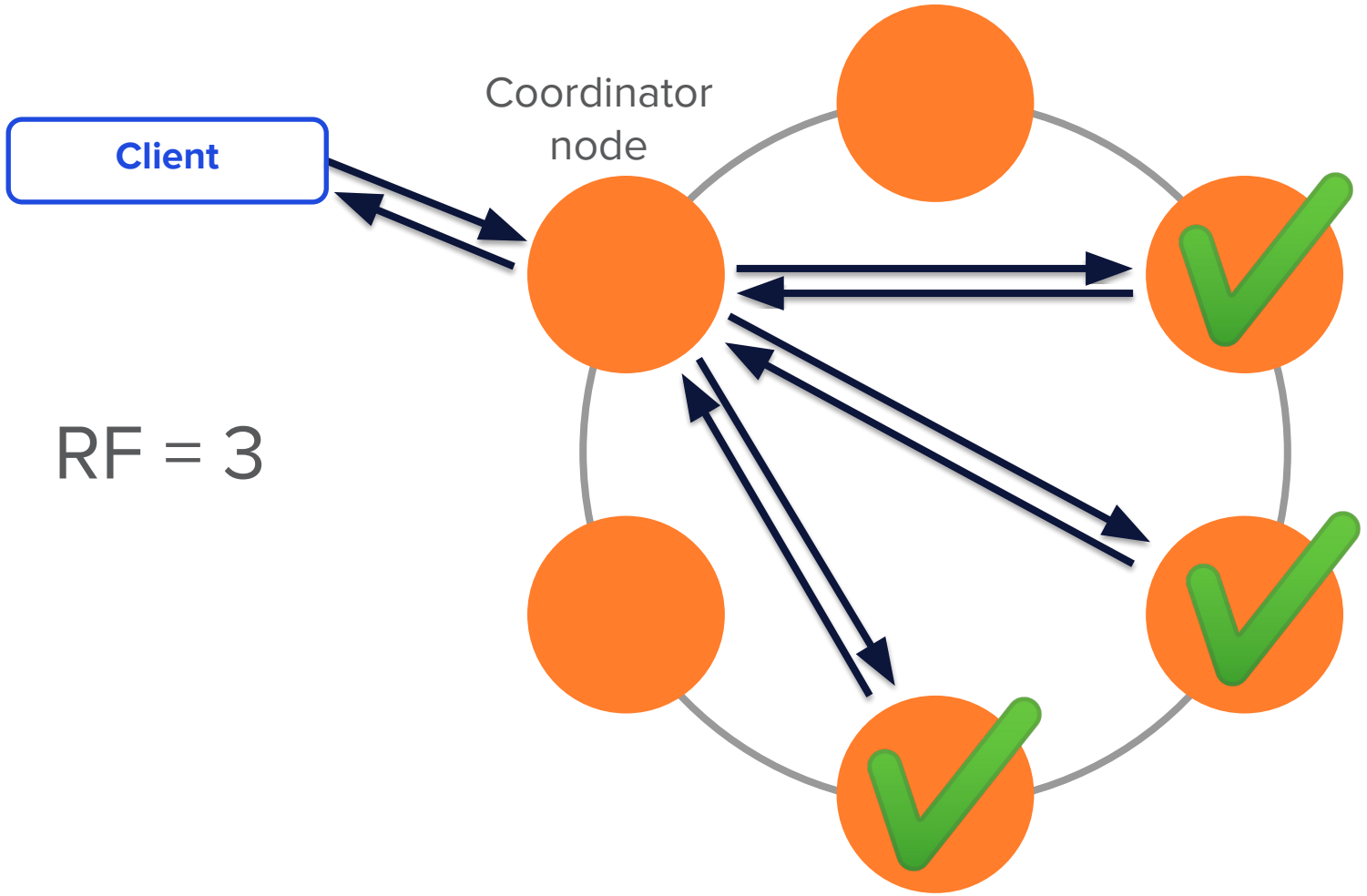
Consistency Levels



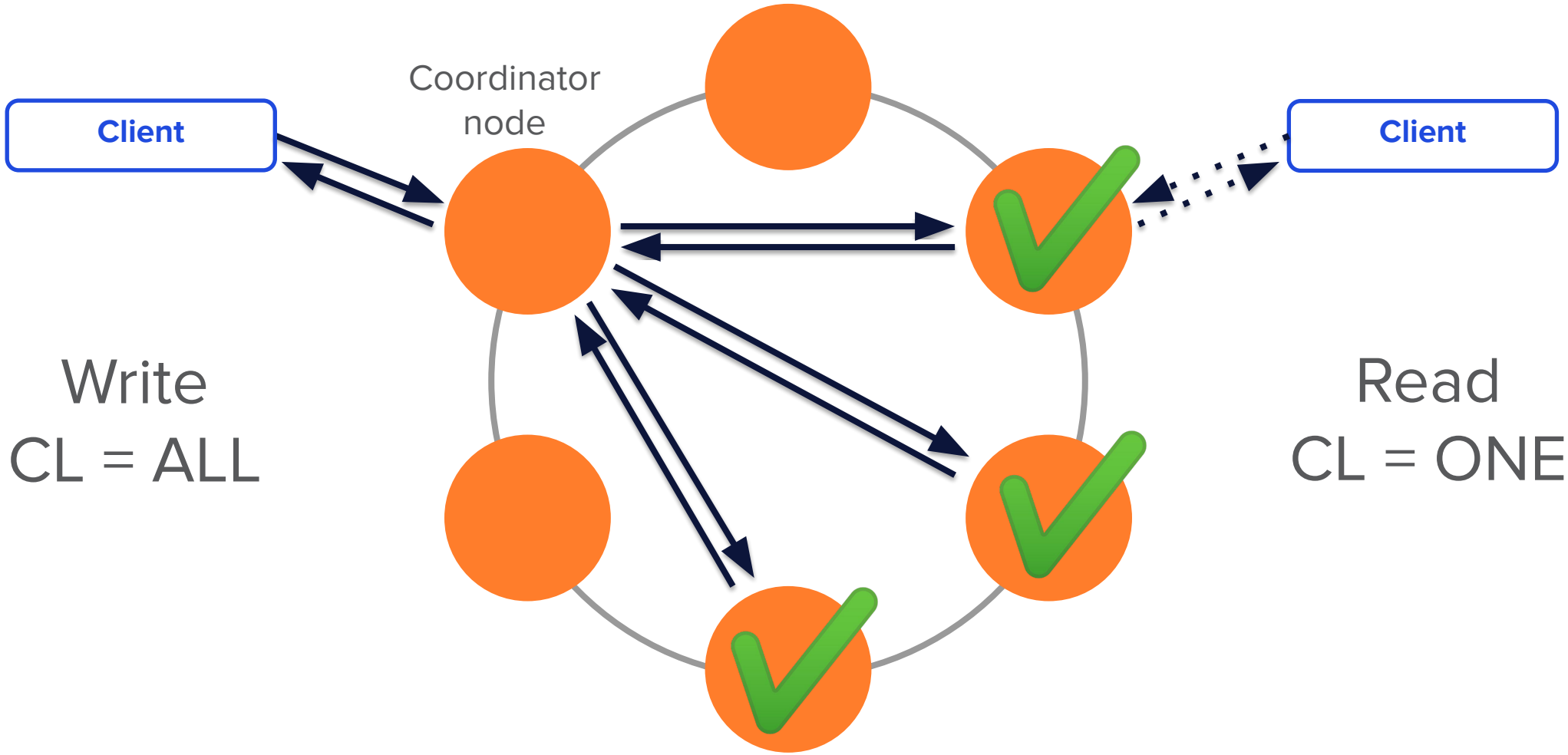
Consistency Levels



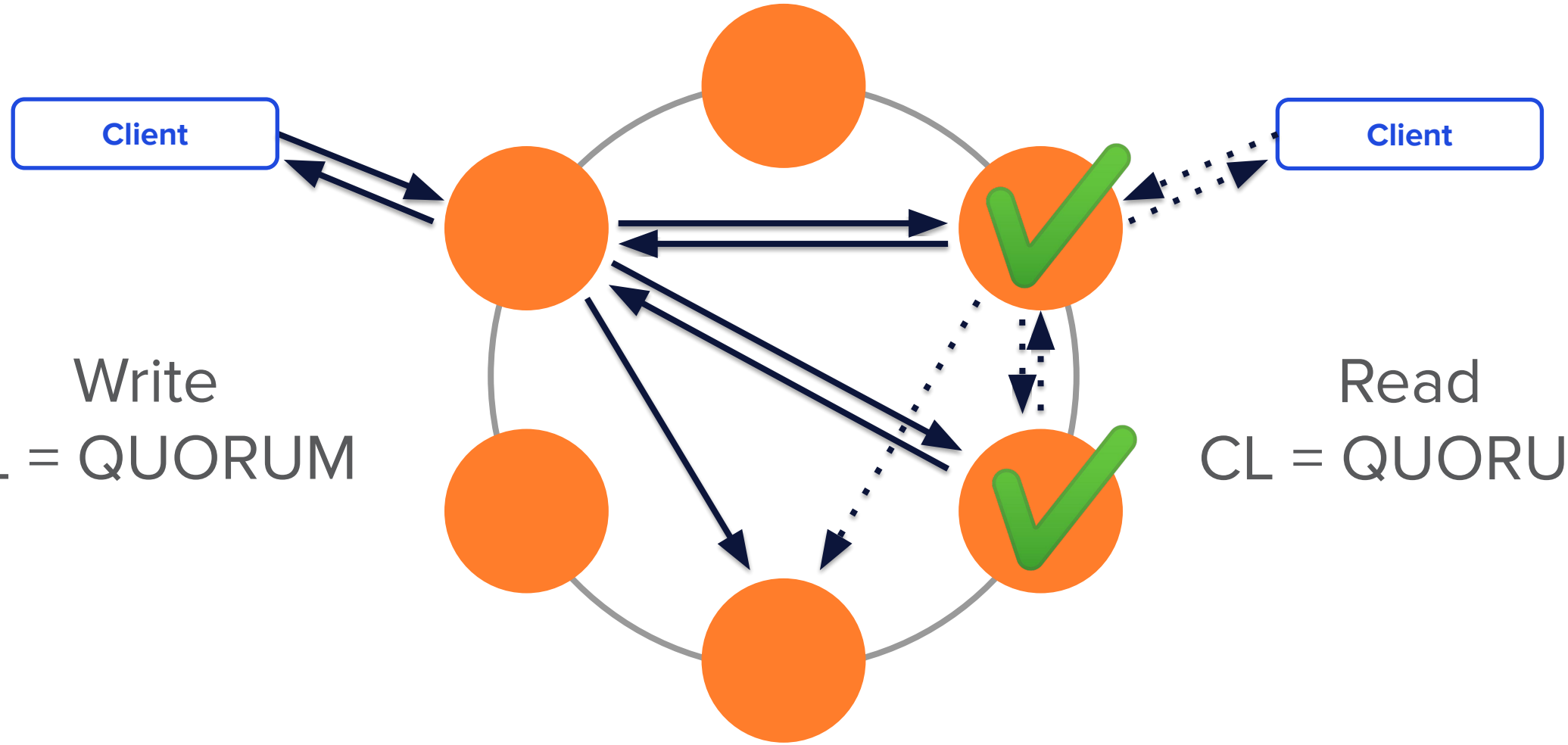
Consistency Levels



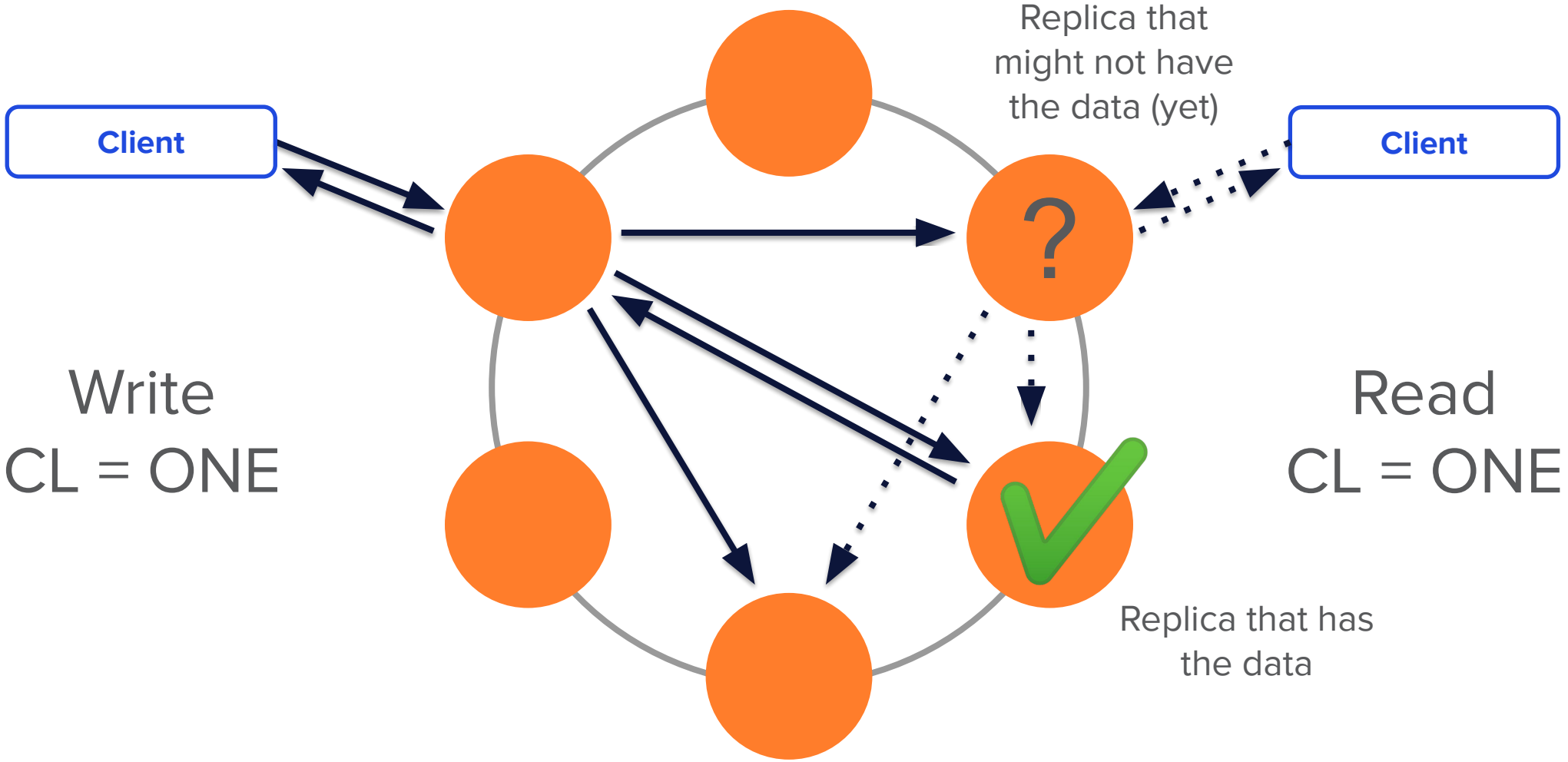
Strong Consistency – One Way



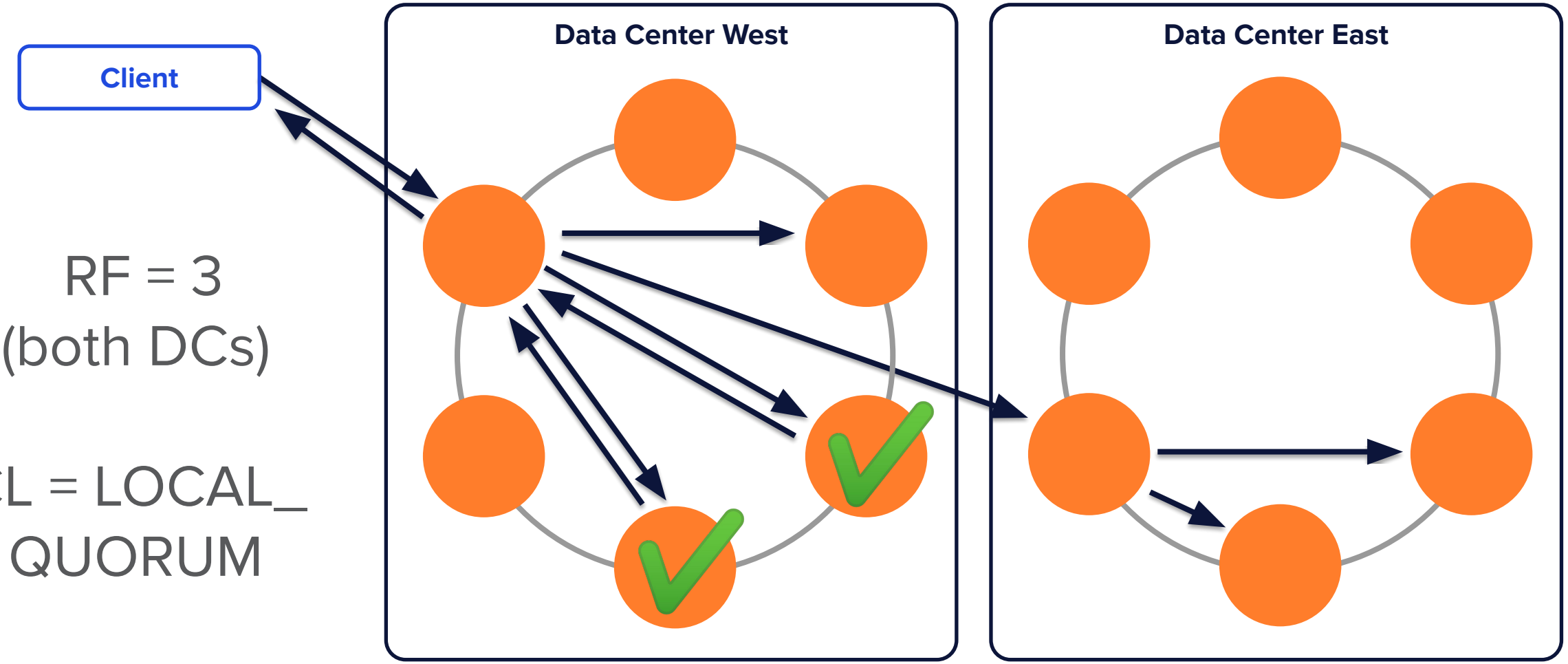
Strong Consistency – A Better Way



Weak(er) Consistency



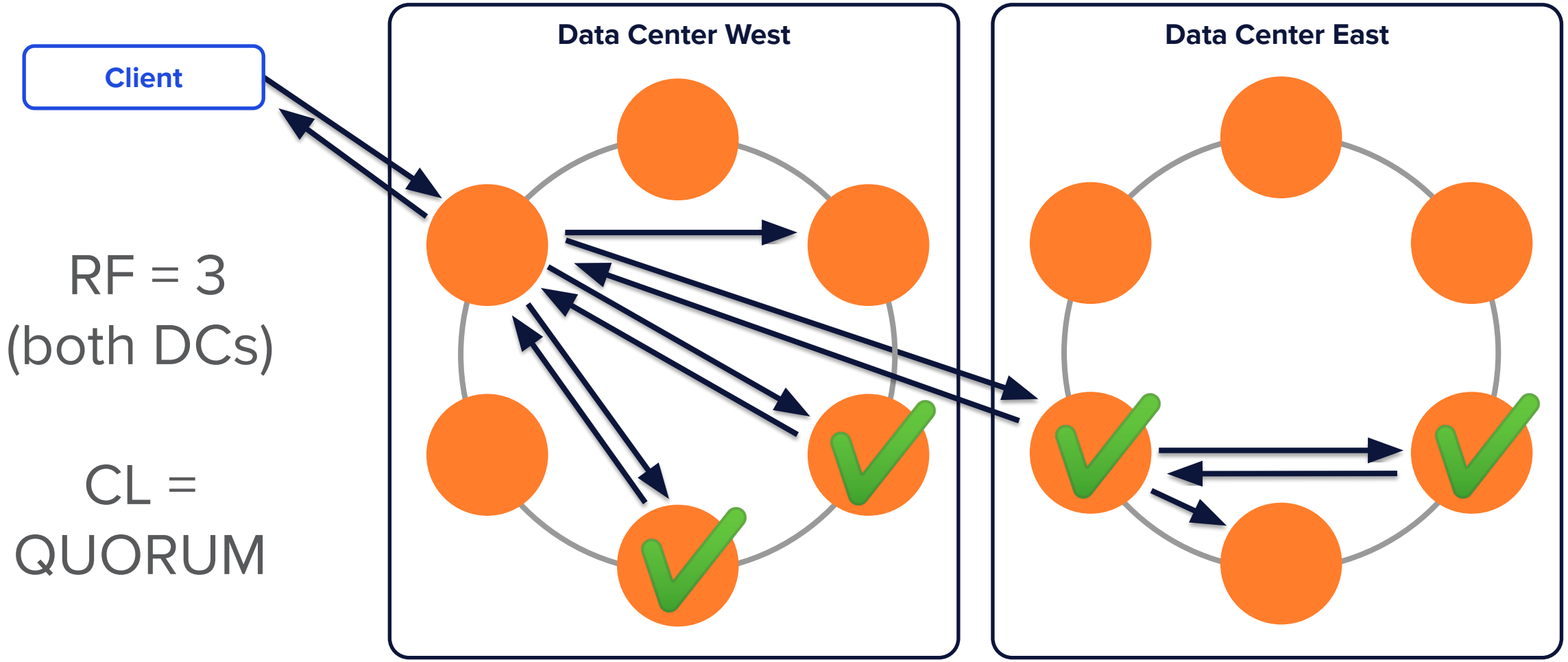
Consistency Across Data Centers



RF = 3
(both DCs)

CL = LOCAL_
QUORUM

Consistency Across Data Centers



Consistency Settings

- Weakest to strongest

Setting	Description
ANY	Storing a hint at minimum is satisfactory
ONE, TWO, THREE	Checks closest node(s) to coordinator
QUORUM	Majority vote, (sum_of_replication_factors / 2) + 1
LOCAL_ONE	Closest node to coordinator in same data center
LOCAL_QUORUM	Closest quorum of nodes in same data center
EACH_QUORUM	Quorum of nodes in each data center, applies to writes only
ALL	Every node must participate

Consistency Tradeoffs

- The higher the consistency, the less chance you may get stale data
  - Pay for this with latency
  - Depends on your situational needs
- IoT and Sensor systems with write-heavy workloads may benefit from CL=ONE for writes if data isn't required immediately

DataStax Developer Day

DATASTAX

Loading Data into Cassandra

Loading Data Into Cassandra with DSBulk

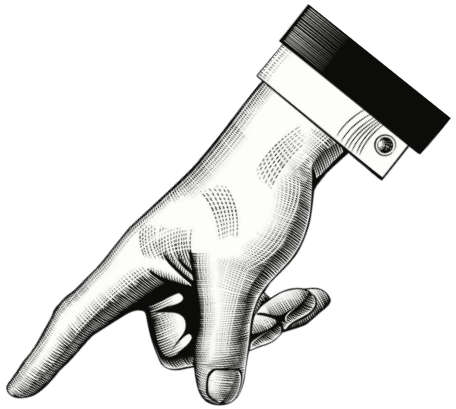
- Moves Cassandra data to/from files in the file system
- Uses both CSV or JSON formats
- Command-line interface
- Available with DataStax supported versions

DSBulk Use Cases

- Loading data from a pile of files
- One-time load or part of production flow
- Initial developer experience (load an existing familiar DB)
- Unload data for backup
- Migration from DSE to DSE (due to data model changes)

DSBulk Example

- Parameters:
  - file1.csv – this is the input file
  - ks1 – this is the keyspace name
  - table1 – this is the table name
- Steps:
  - Create the keyspace and table
  - Map the column using the header or via a config file
  - Run the command

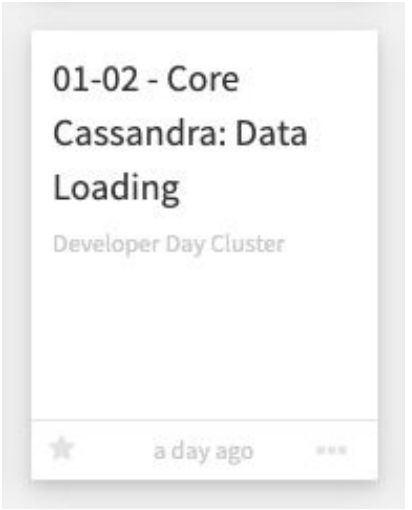


```
dsbulk load -url file1.csv -k ks1 -t table1
```

Time for an exercise!

DATASTAX

“Core Cassandra: Data Loading” Notebook



Loading Data into Cassandra - Review

- dsbulk is a command line tool for loading/unloading data
  - Use header or config file to map columns
  - Works for CSV and JSON
  - Create your table first
  - Handles various data types (e.g., text, date, float)

DataStax Developer Day

DATASTAX

Data Availability

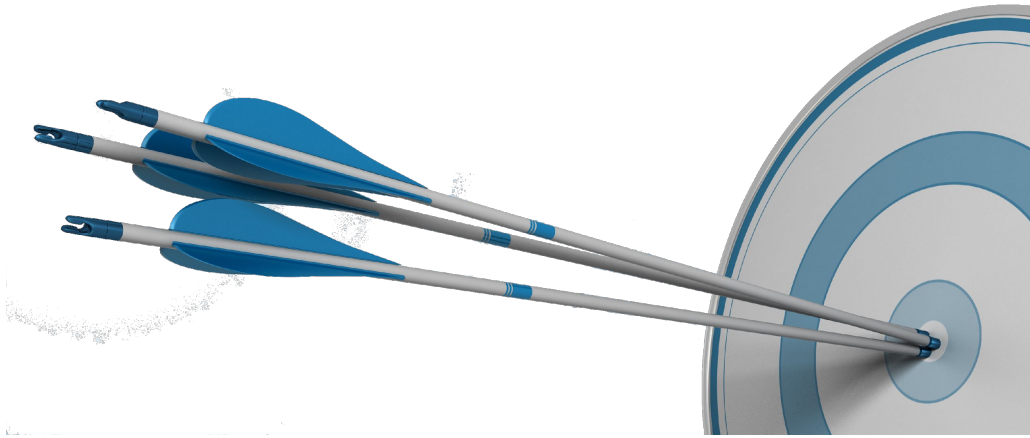
Cassandra Data Availability

- Availability = Replication
- Replication implies consistency concerns

~~Consistency must be absolute!~~ Consistency can be a continuum

Cassandra Data Availability Concepts

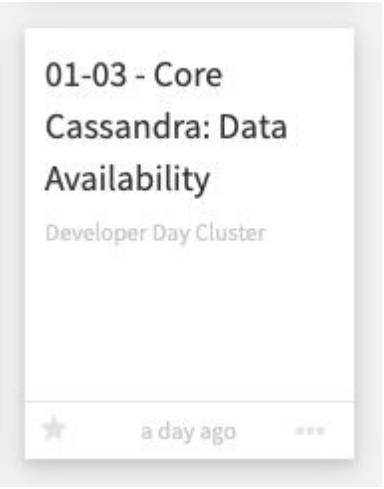
- Replication Factor – the number of copies of your data
  - Increasing replication increases chances of availability
  - Increasing replication increases chances of inconsistency
- Consistency Level – the number of acknowledged copies read/written
- Strong Consistency: number of writes + number of reads > replication factor
- EXAMPLE:
  - Replication Factor = 3
  - Read/Write Consistency Level = QUORUM
  - 2 + 2 = 4 > 3 👍



Time for an exercise!

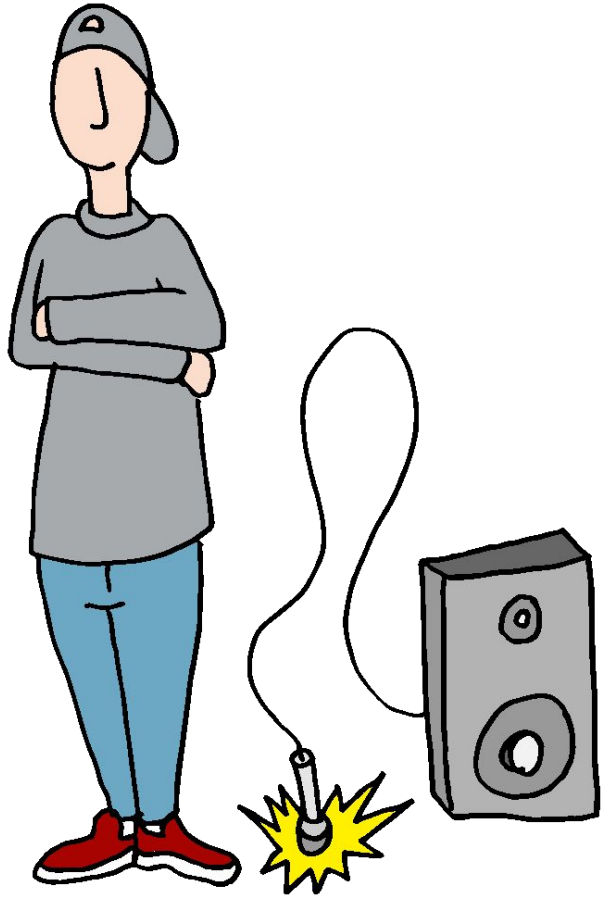
DATASTAX

“Core Cassandra: Data Availability” Notebook



Final Words

- Now, you should:
  - Understand what Apache Cassandra is
  - Know what Cassandra does
  - Yearn to have Cassandra in your shop
- Want More?
  - Visit academy.datastax.com
  - It's free!



DATASTAX®

Thank You