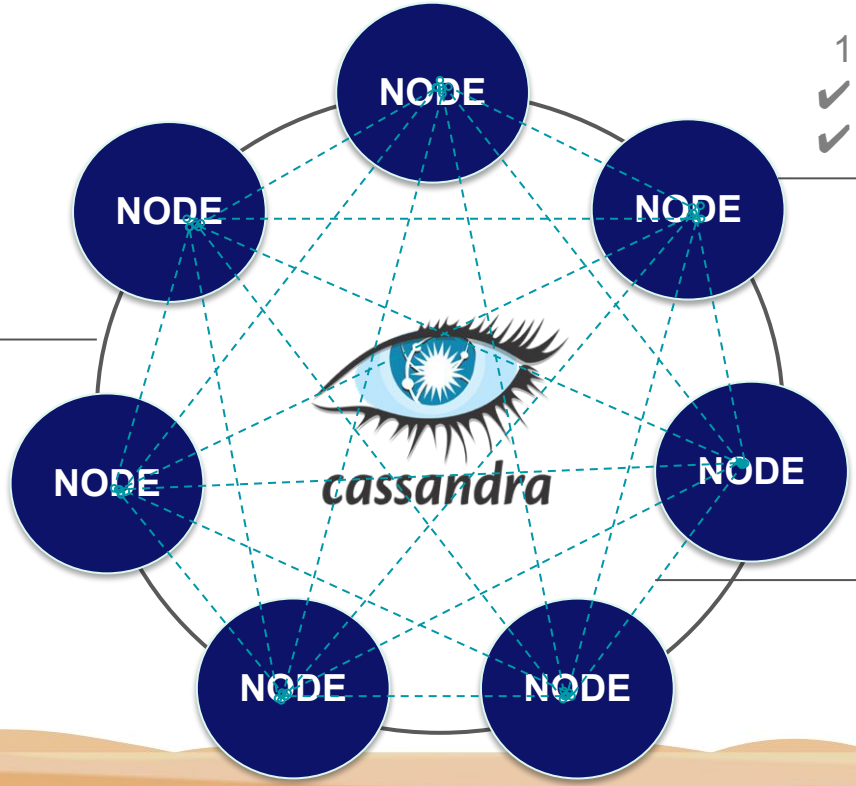
Developer Workshop Series Week 1

- ➔ Bootstrapping
- ➔ Apache Cassandra™ Why, What & When
- ➔ Read and Write path
- ➔ Uber High Level Data Modeling
- ➔ What's NEXT?

Apache Cassandra™ = NoSQL Distributed Database

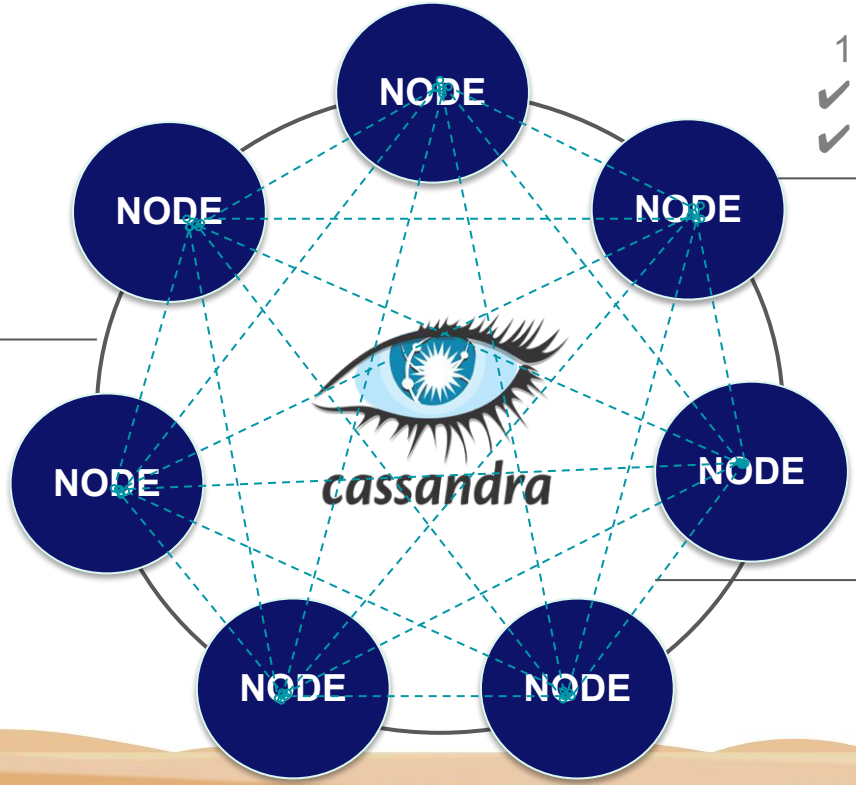


1 Installation = 1 NODE
- ✔ Capacity = ~ 2-4TB
- ✔ Throughput = LOTS Tx/sec/core

DataCenter | Ring

Communication:
- ✔ Gossiping

Apache Cassandra™ = NoSQL Distributed Database



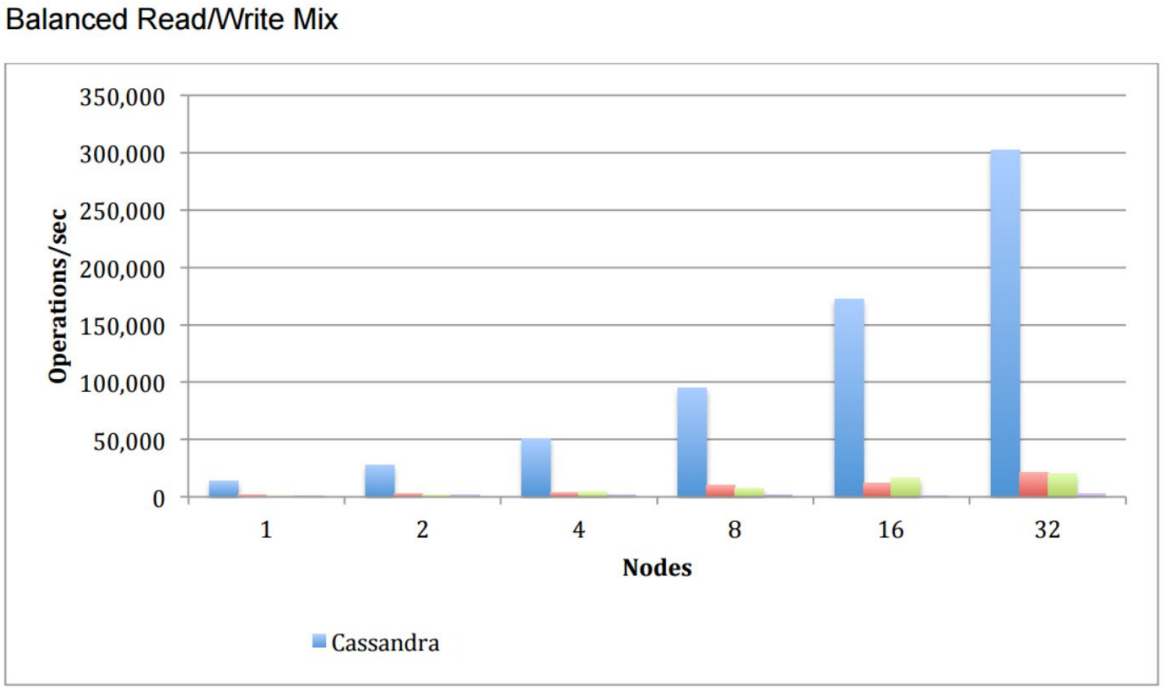
1 Installation = 1 NODE
- ✔ Capacity = ~ 2-4TB
- ✔ Throughput = ~ 6-12k Tx/sec/core

DataCenter | Ring

Communication:
- ✔ Gossiping

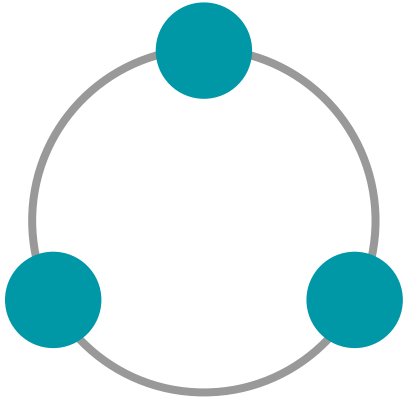
Scales Linearly

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

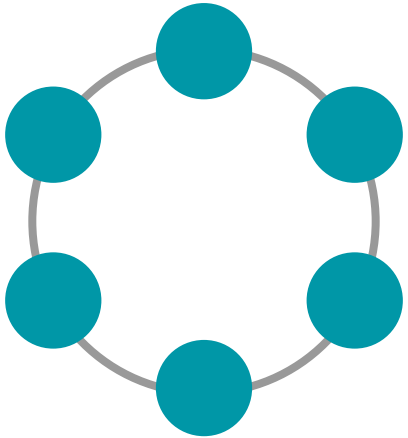


Horizontal vs. Vertical Scaling

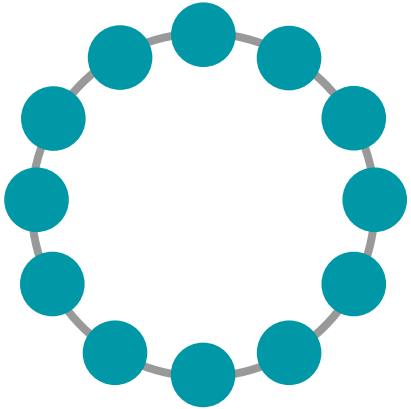
- Vertical scaling requires one large expensive machine
- Horizontal scaling requires multiple less-expensive commodity hardware



100,000 transactions/second



200,000 transactions/second



400,000 transactions/second

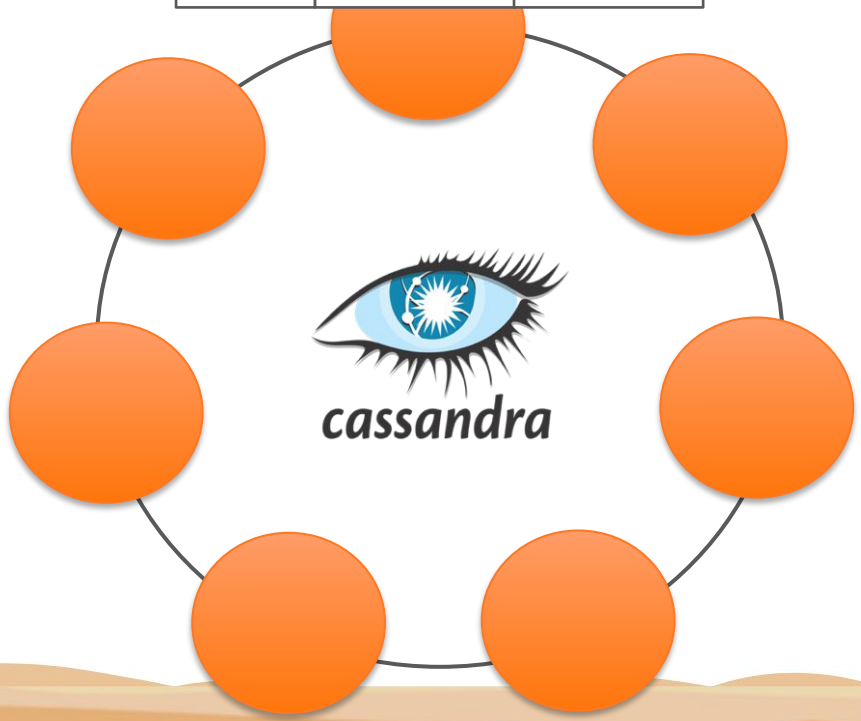
Data is Distributed

Country	City	Population
USA	New York	8.000.000
USA	Los Angeles	4.000.000
FR	Paris	2.230.000
DE	Berlin	3.350.000
UK	London	9.200.000
AU	Sydney	4.900.000
DE	Nuremberg	500.000
CA	Toronto	6.200.000
CA	Montreal	4.200.000
FR	Toulouse	1.100.000
JP	Tokyo	37.430.000
IN	Mumbai	20.200.000

Partition Key

Data is Distributed

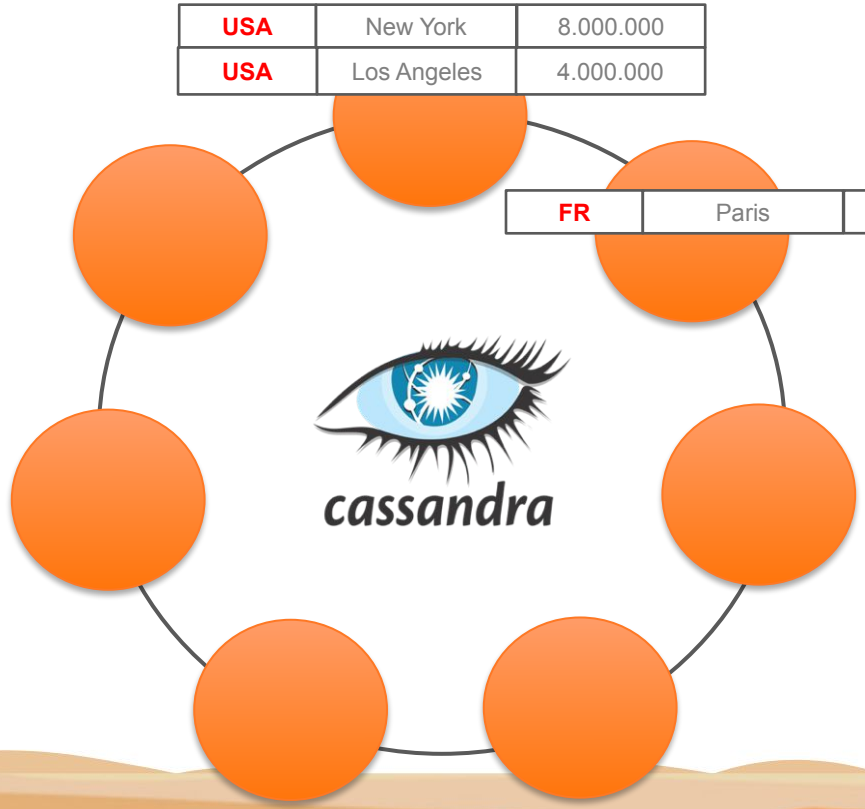
USA	New York	8.000.000
USA	Los Angeles	4.000.000



Country	City	Population
FR	Paris	2.230.000
DE	Berlin	3.350.000
UK	London	9.200.000
AU	Sydney	4.900.000
DE	Nuremberg	500.000
CA	Toronto	6.200.000
CA	Montreal	4.200.000
FR	Toulouse	1.100.000
JP	Tokyo	37.430.000
IN	Mumbai	20.200.000

Partition Key

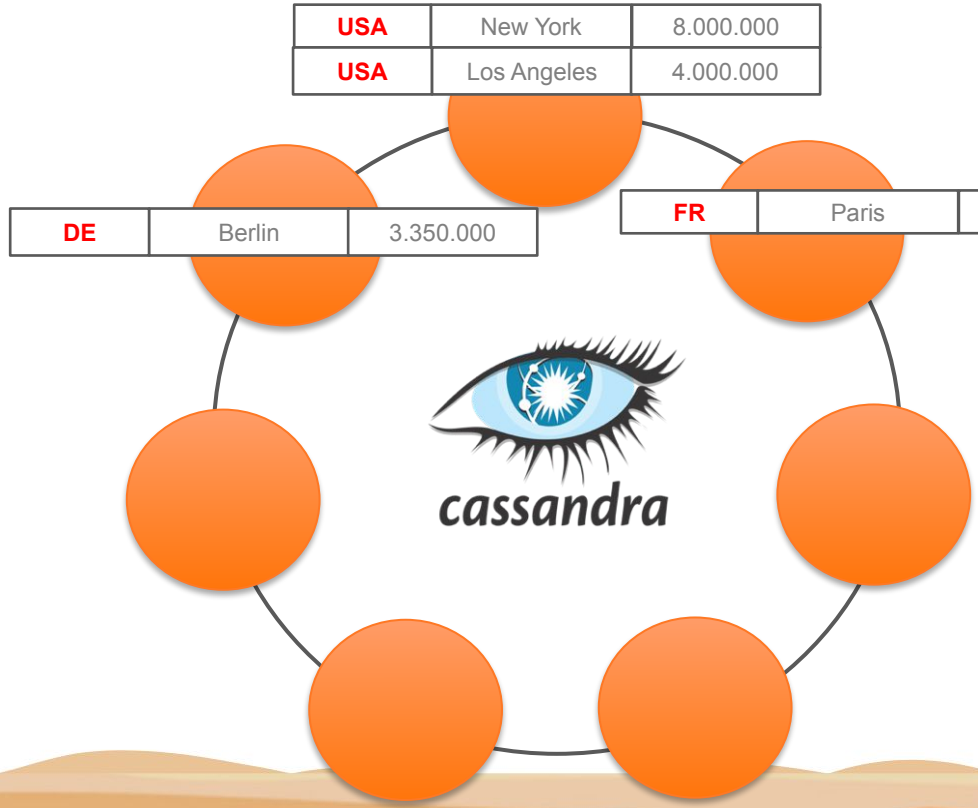
Data is Distributed



Country	City	Population
DE	Berlin	3.350.000
UK	London	9.200.000
AU	Sydney	4.900.000
DE	Nuremberg	500.000
CA	Toronto	6.200.000
CA	Montreal	4.200.000
FR	Toulouse	1.100.000
JP	Tokyo	37.430.000
IN	Mumbai	20.200.000

Partition Key

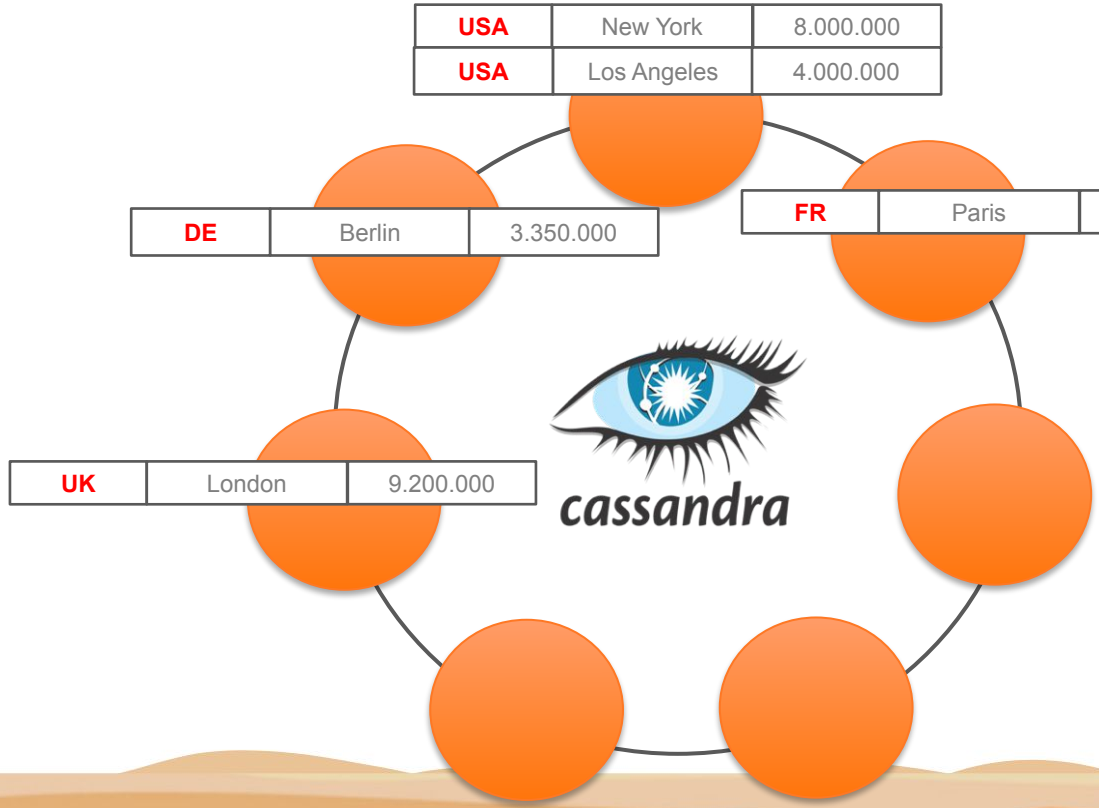
Data is Distributed



Country	City	Population
UK	London	9.200.000
AU	Sydney	4.900.000
DE	Nuremberg	500.000
CA	Toronto	6.200.000
CA	Montreal	4.200.000
FR	Toulouse	1.100.000
JP	Tokyo	37.430.000
IN	Mumbai	20.200.000

Partition Key

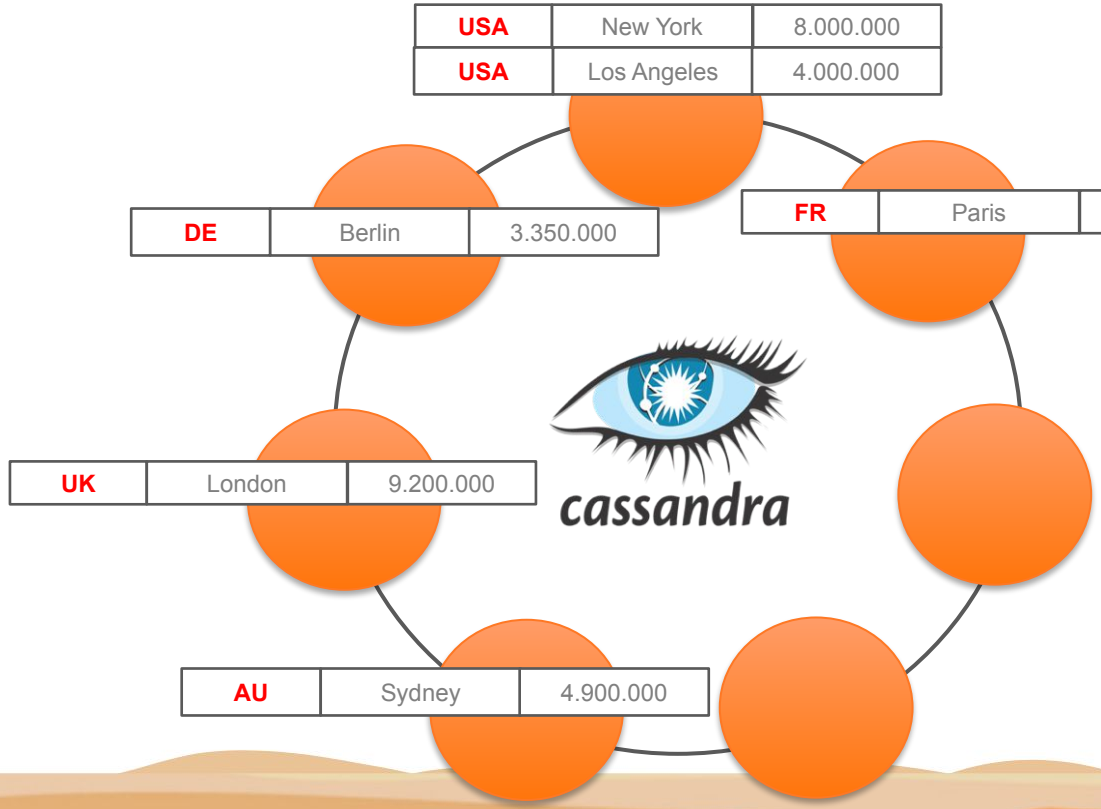
Data is Distributed



Country	City	Population
AU	Sydney	4.900.000
DE	Nuremberg	500.000
CA	Toronto	6.200.000
CA	Montreal	4.200.000
FR	Toulouse	1.100.000
JP	Tokyo	37.430.000
IN	Mumbai	20.200.000

Partition Key

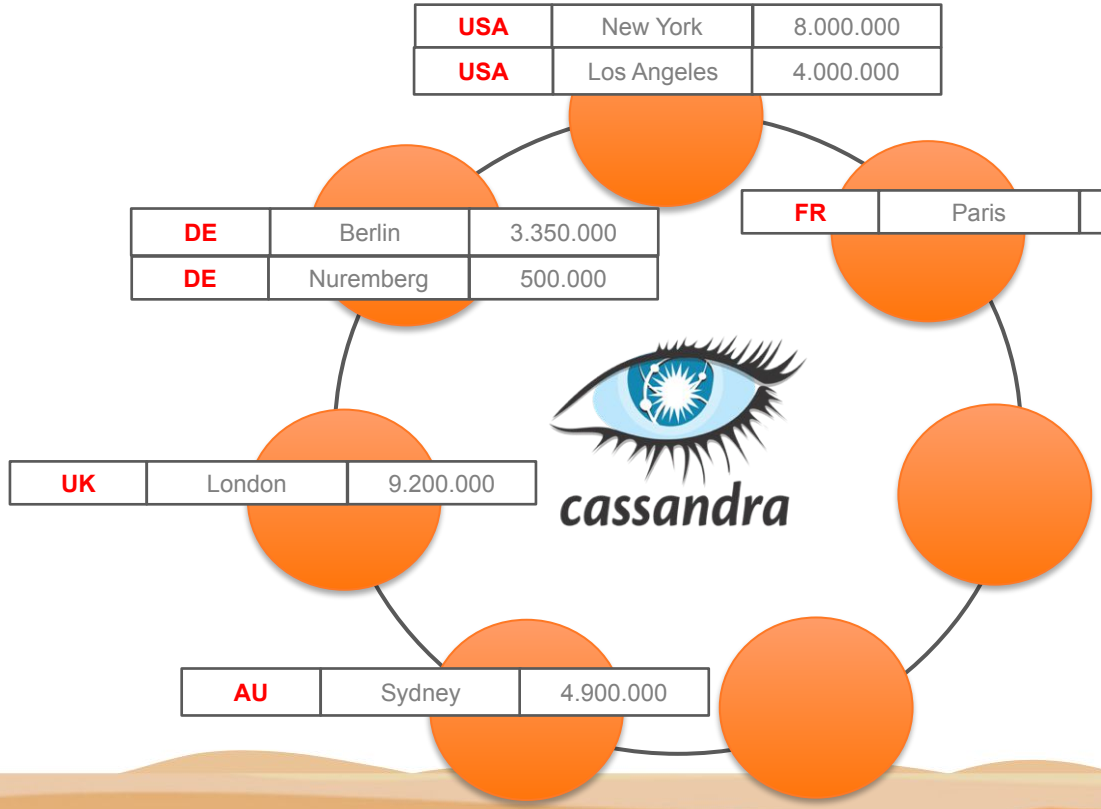
Data is Distributed



Country	City	Population
DE	Nuremberg	500.000
CA	Toronto	6.200.000
CA	Montreal	4.200.000
FR	Toulouse	1.100.000
JP	Tokyo	37.430.000
IN	Mumbai	20.200.000

Partition Key

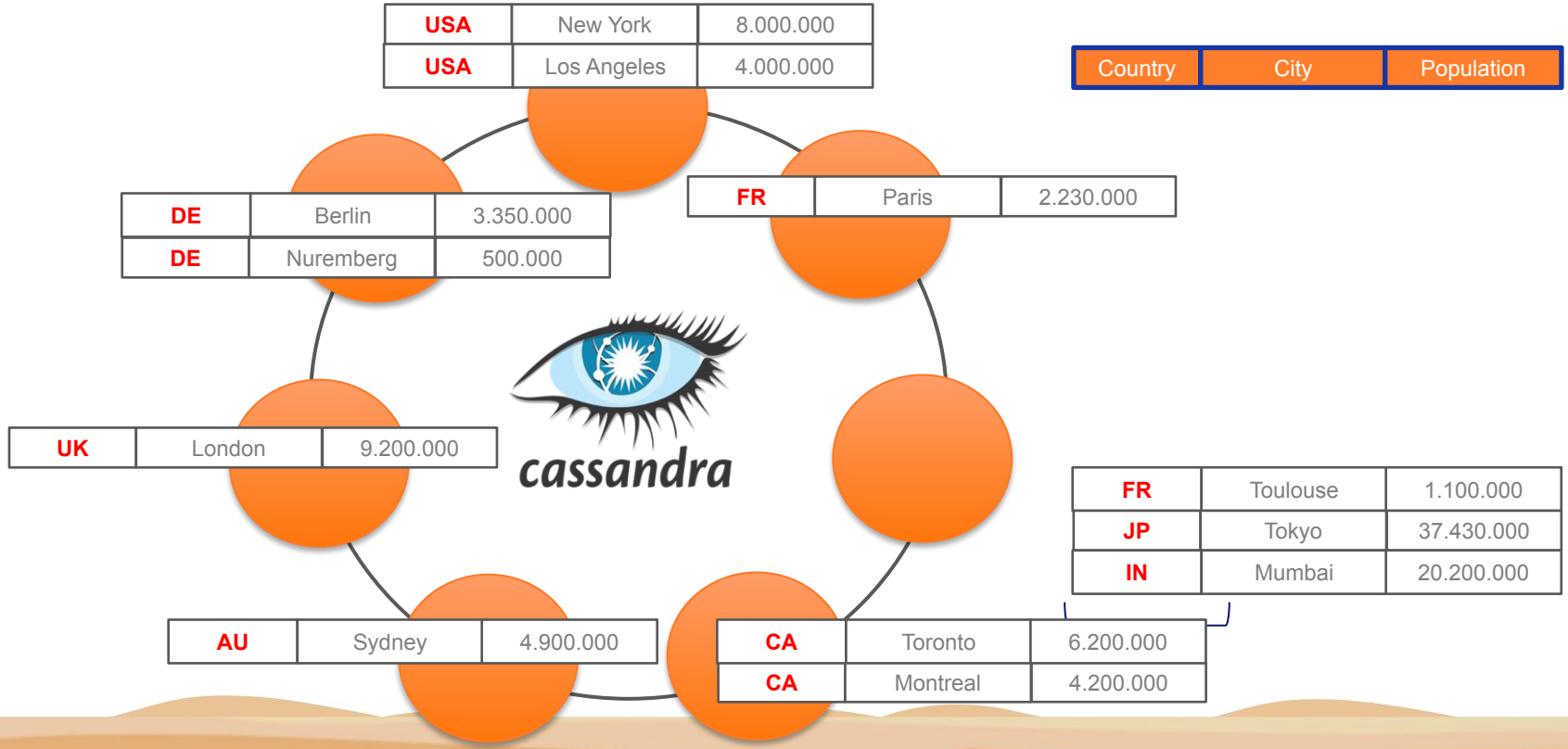
Data is Distributed



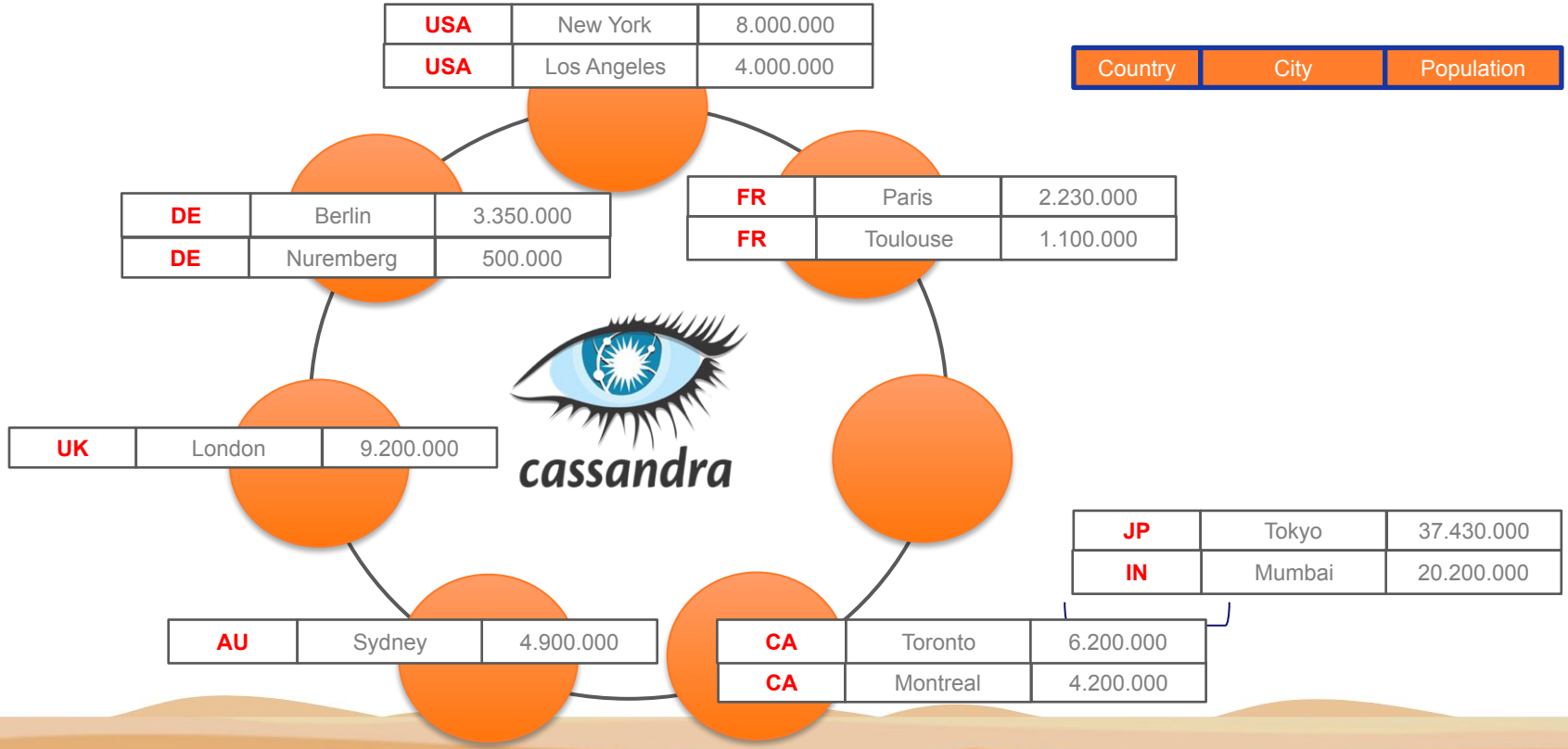
Country	City	Population
CA	Toronto	6.200.000
CA	Montreal	4.200.000
FR	Toulouse	1.100.000
JP	Tokyo	37.430.000
IN	Mumbai	20.200.000

Partition Key

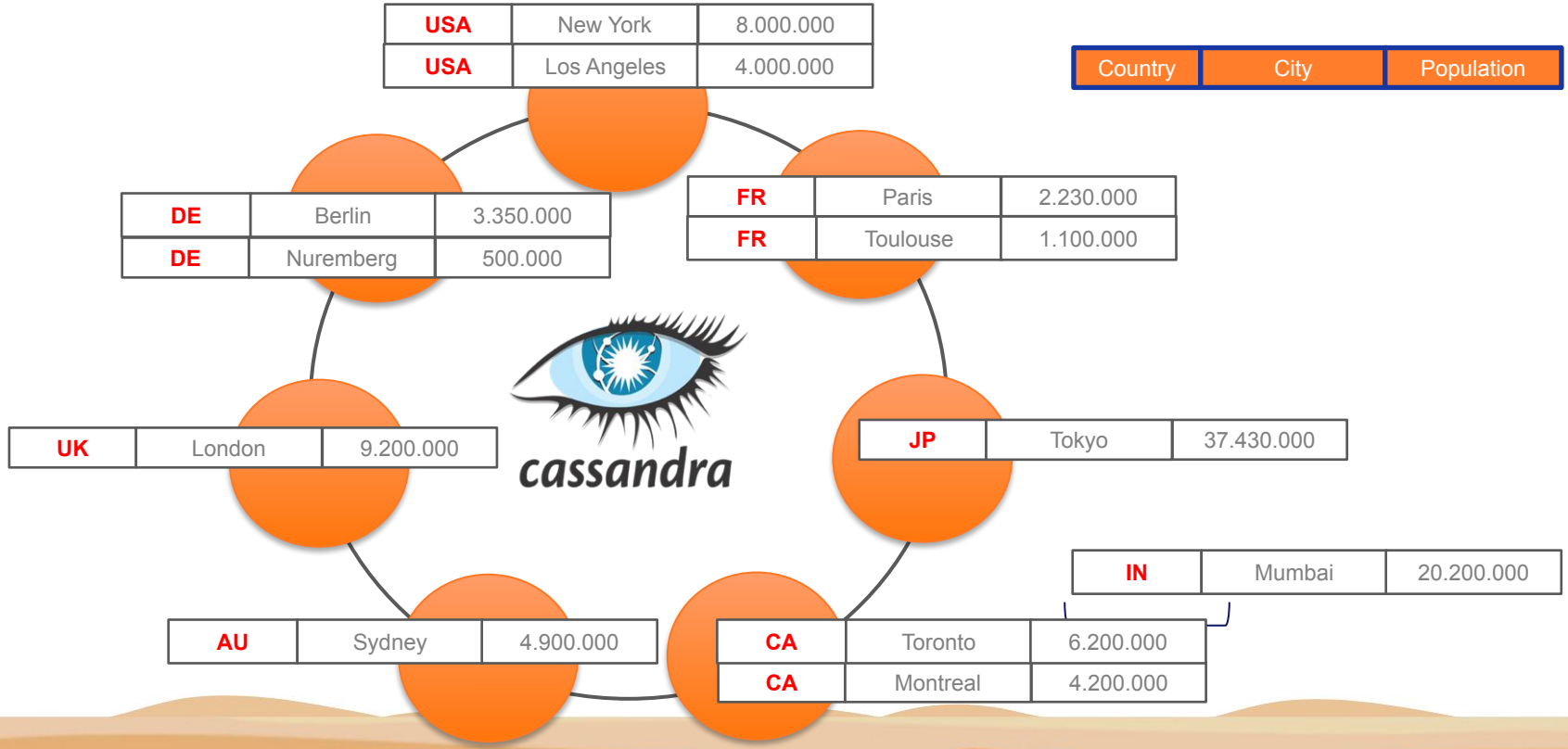
Data is Distributed



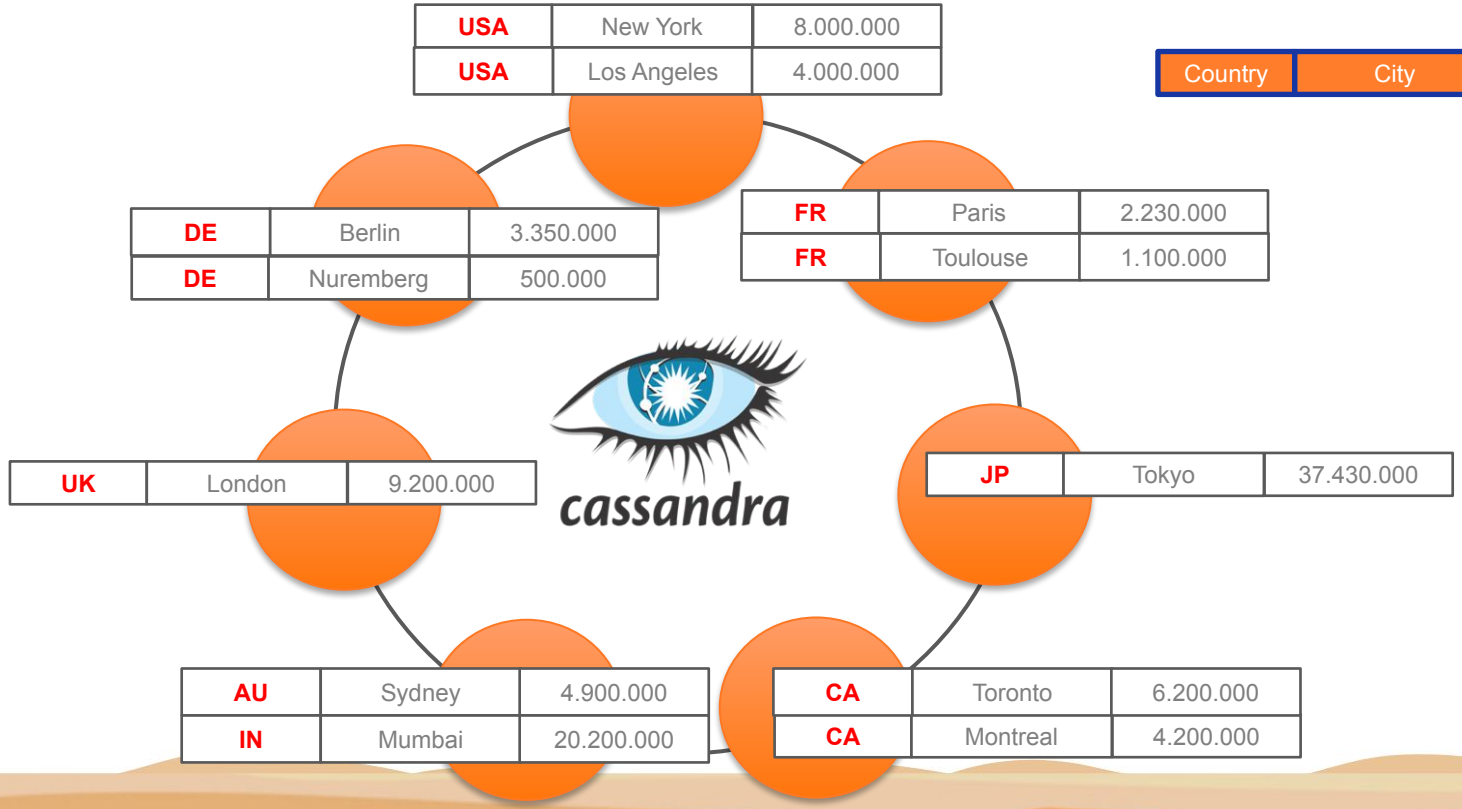
Data is Distributed



Data is Distributed

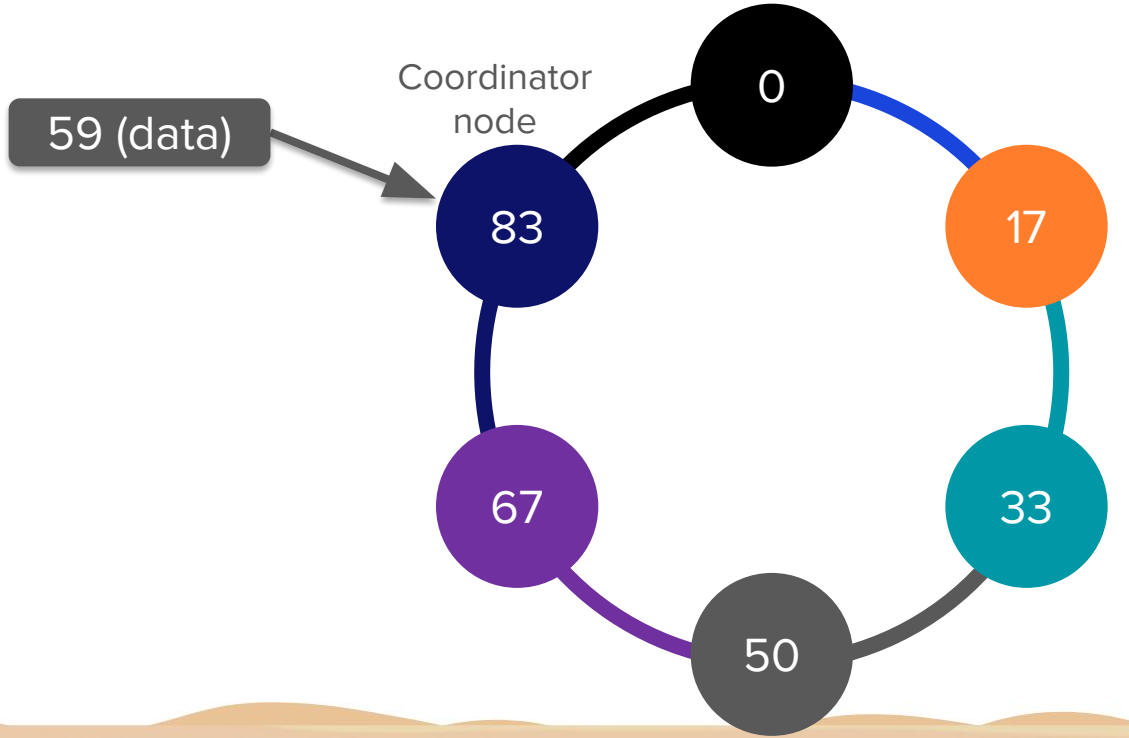


Data is Distributed

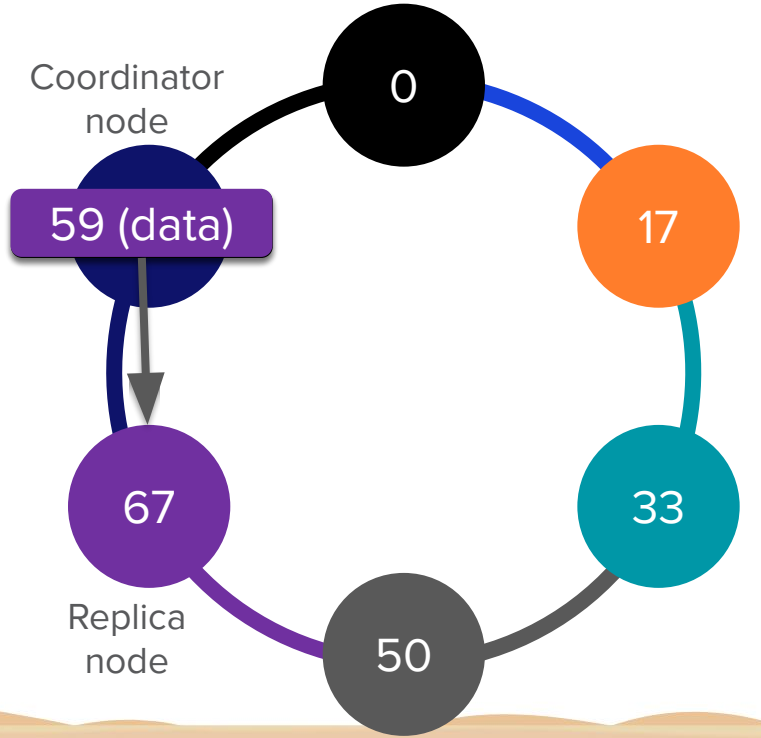


Replication

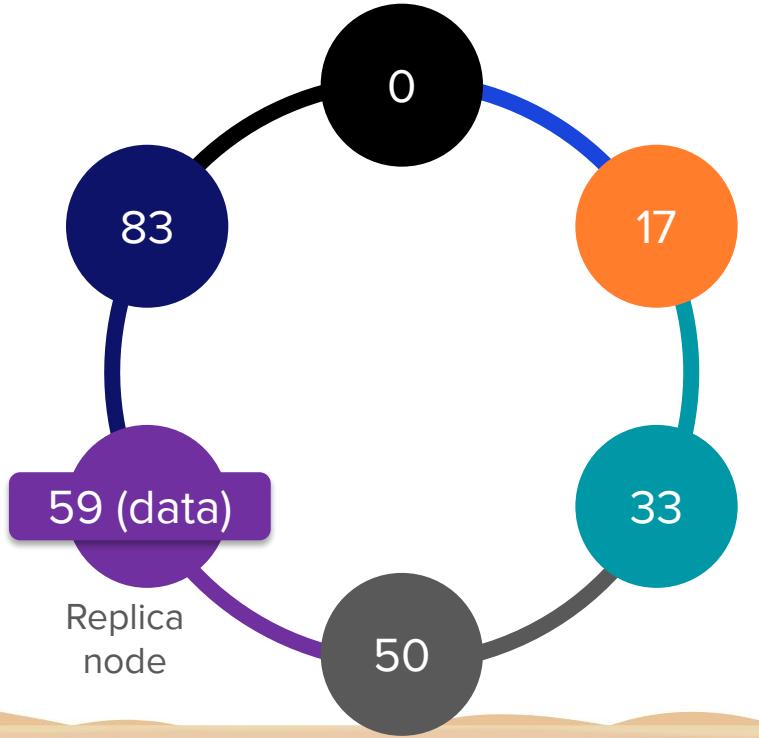
How the Ring Works



How the Ring Works

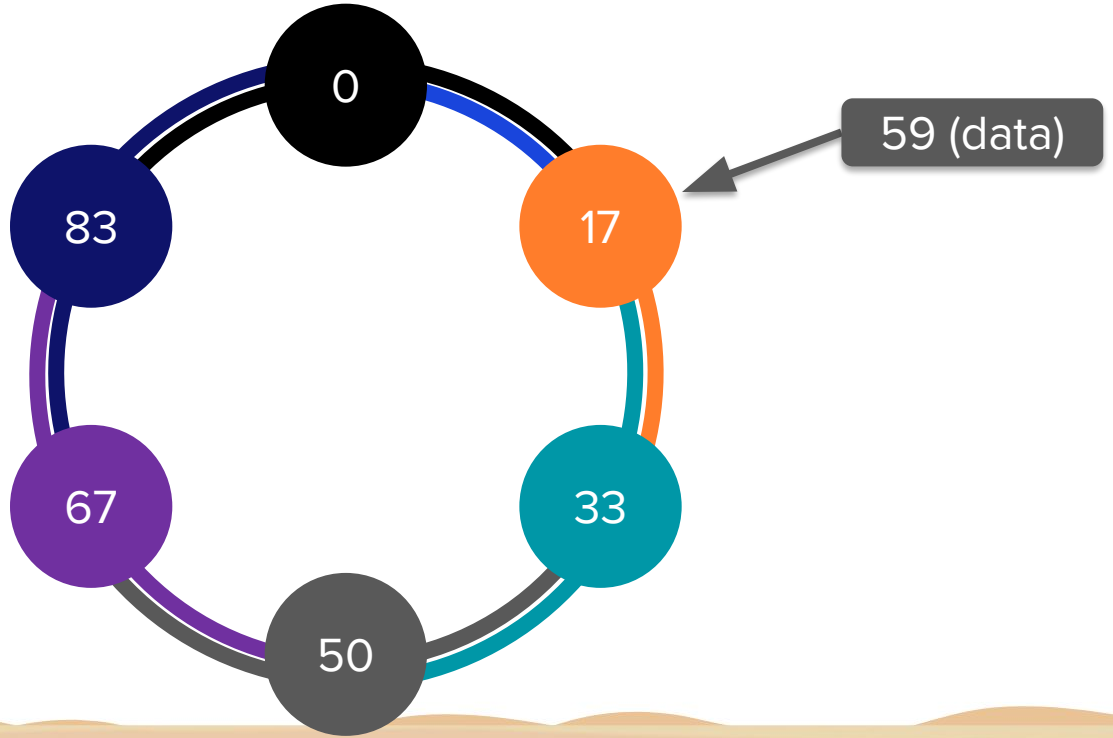


How the Ring Works



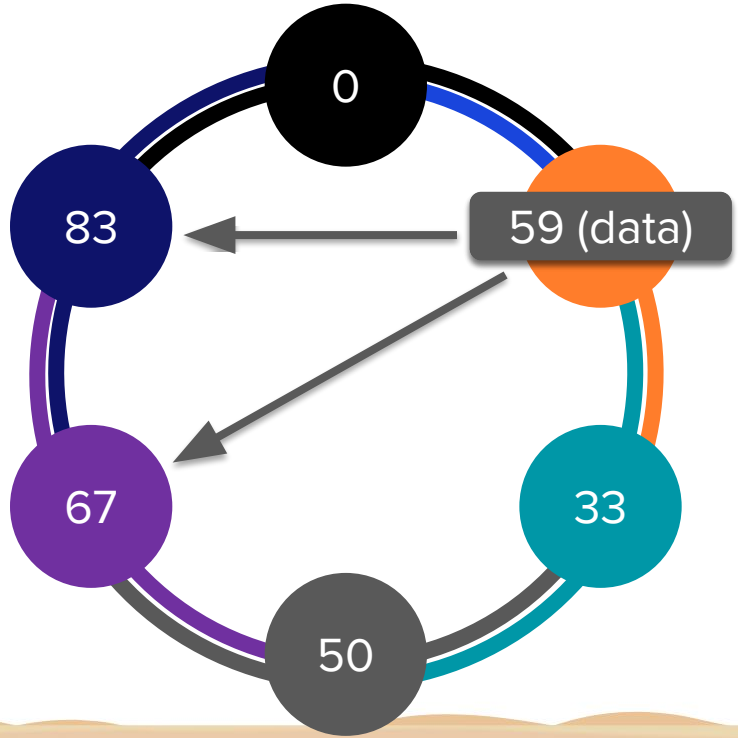
Replication within the Ring

RF = 2



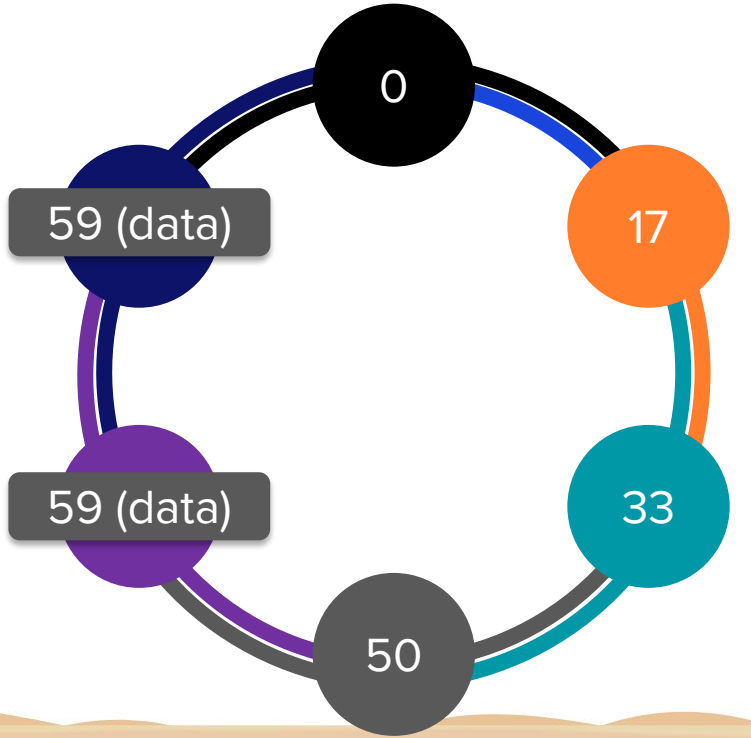
Replication within the Ring

RF = 2



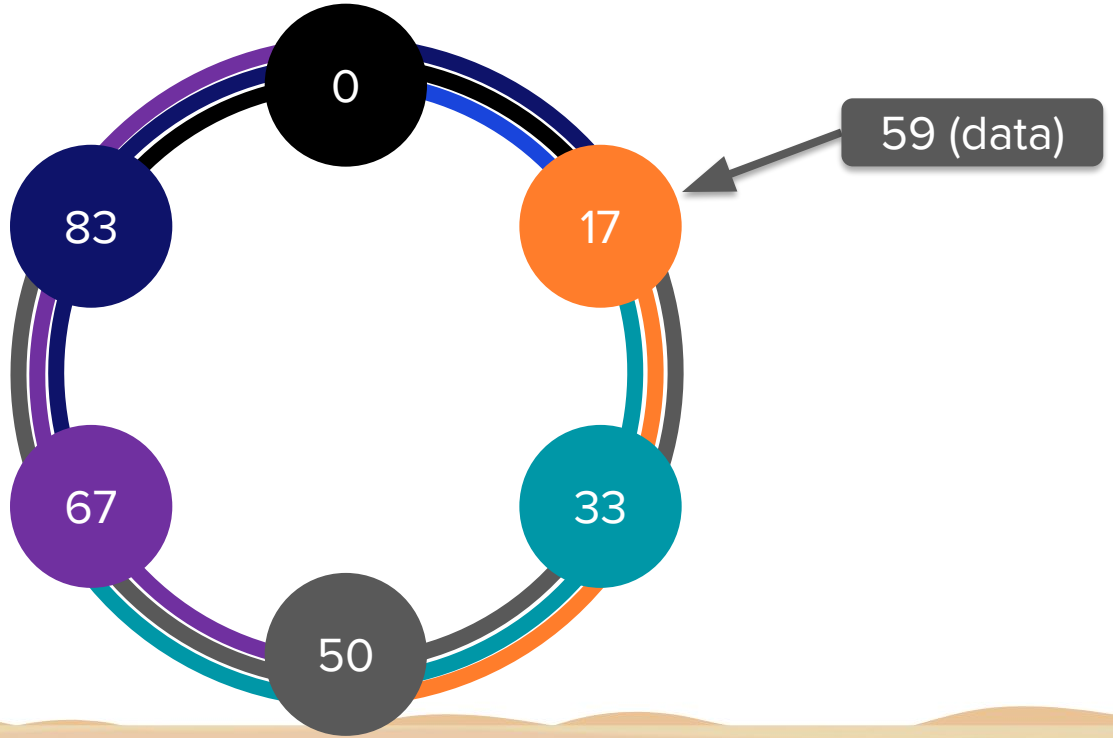
Replication within the Ring

RF = 2



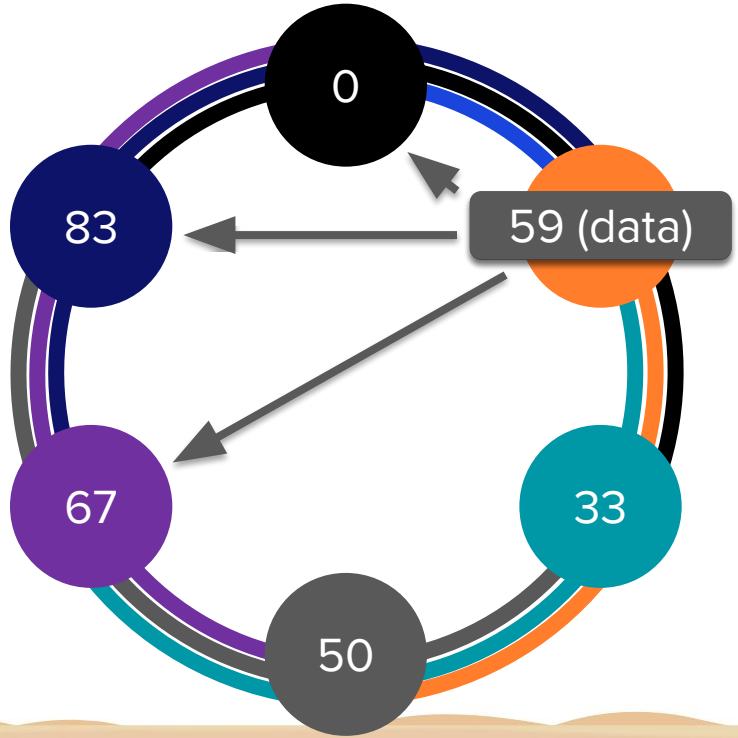
Replication within the Ring

RF = 3



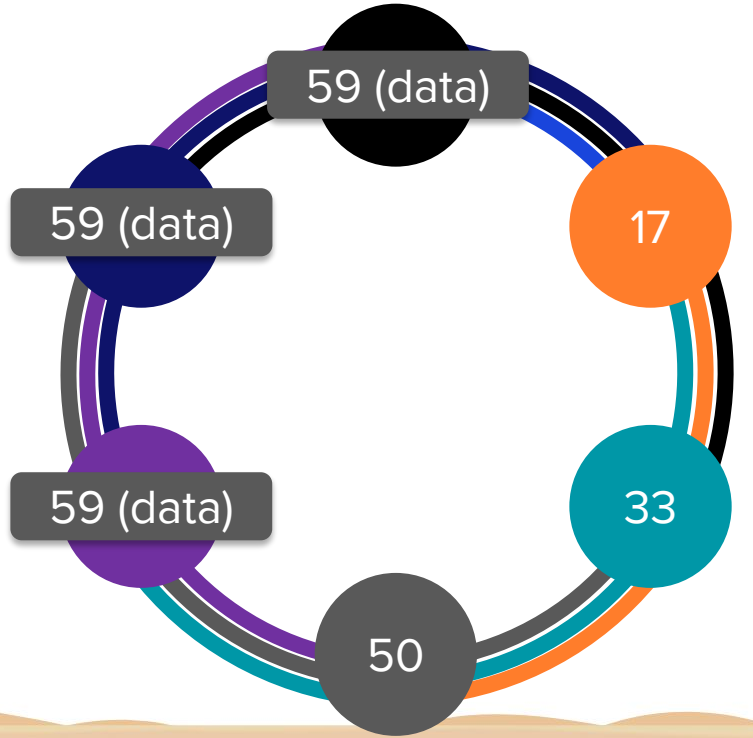
Replication within the Ring

RF = 3



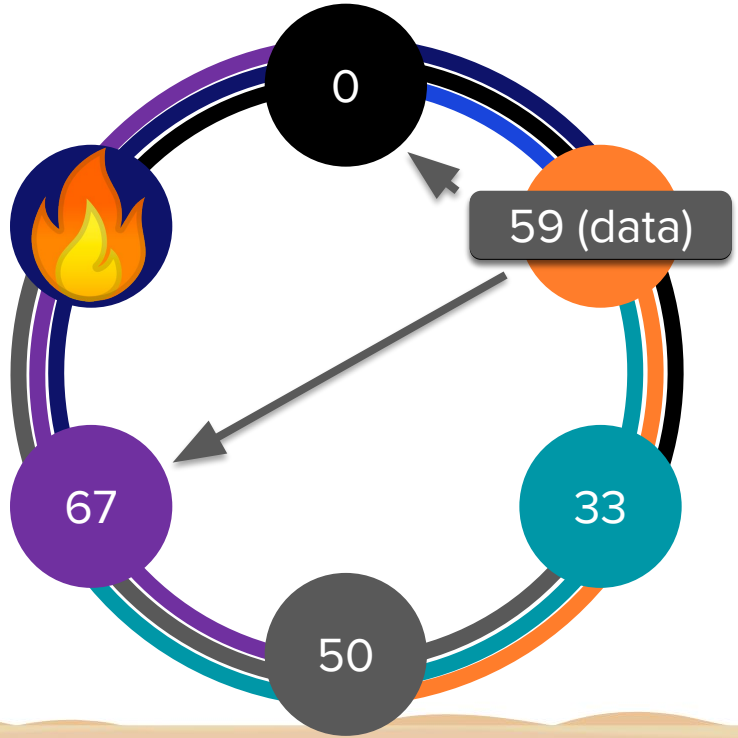
Replication within the Ring

RF = 3



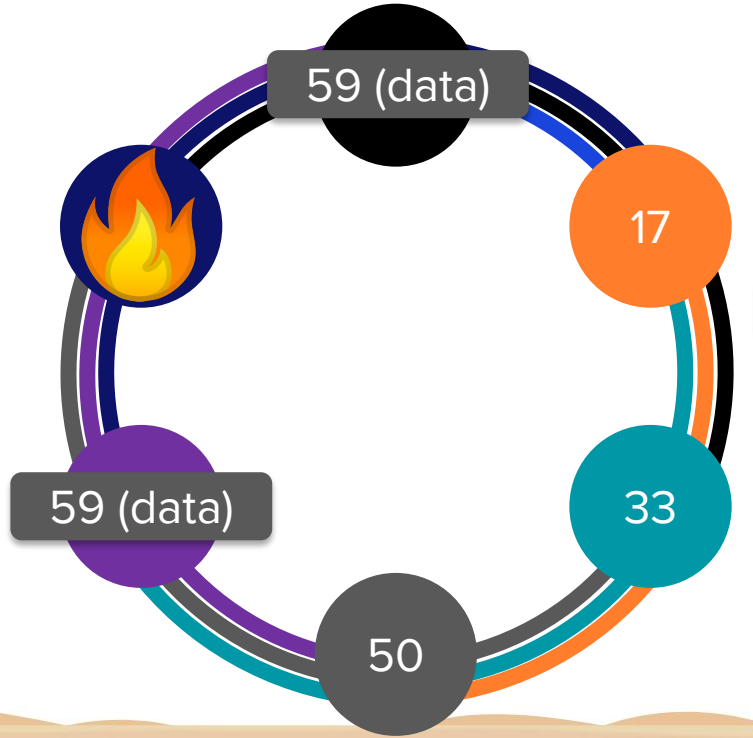
Node Failure

RF = 3



Node Failure

RF = 3

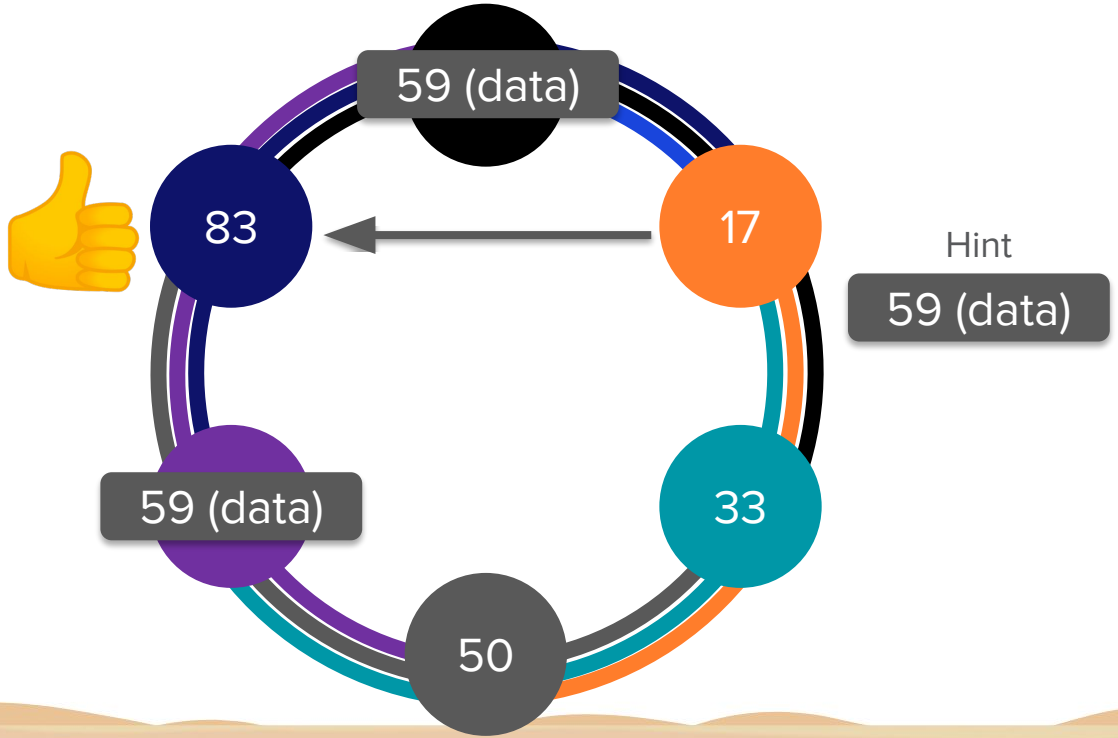


Hint

59 (data)

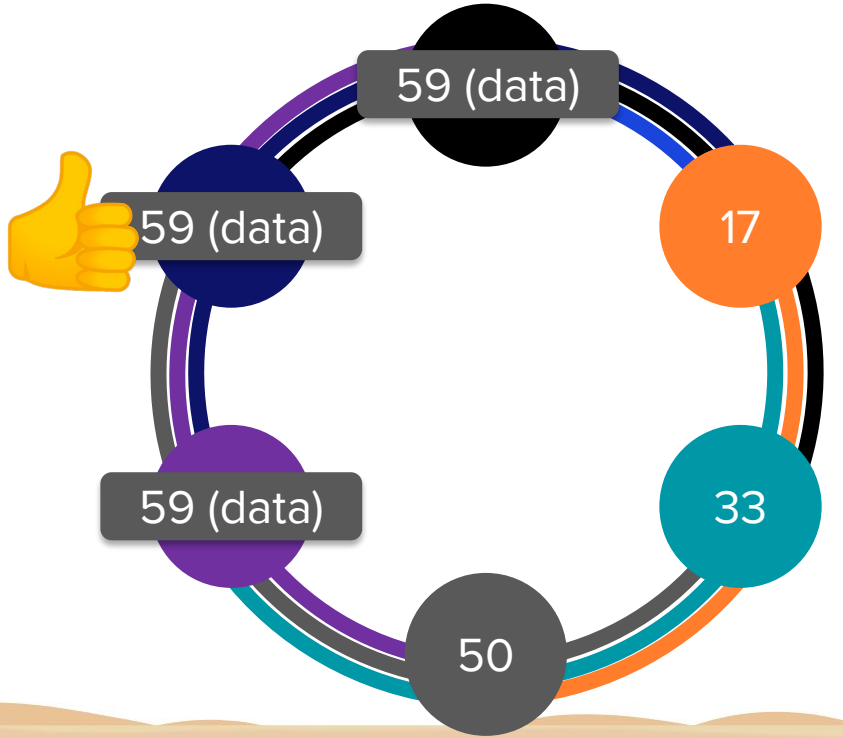
Node Failure

RF = 3



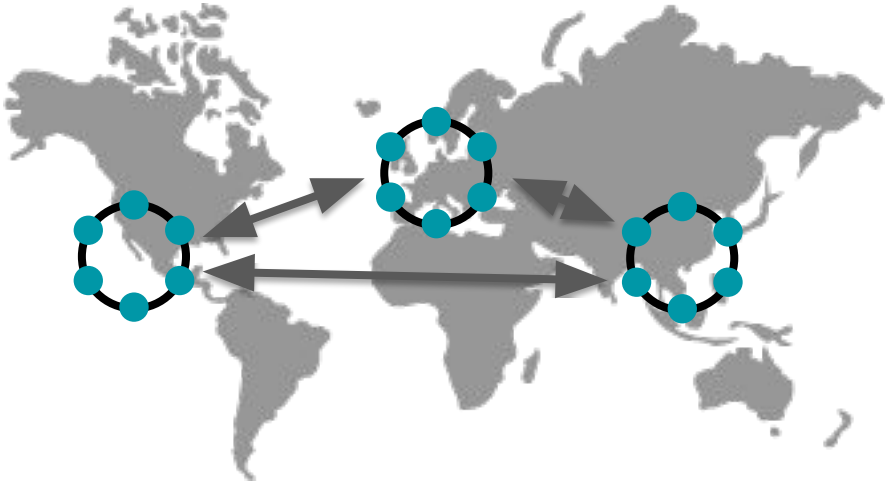
Node Failure – Recovered!

RF = 3

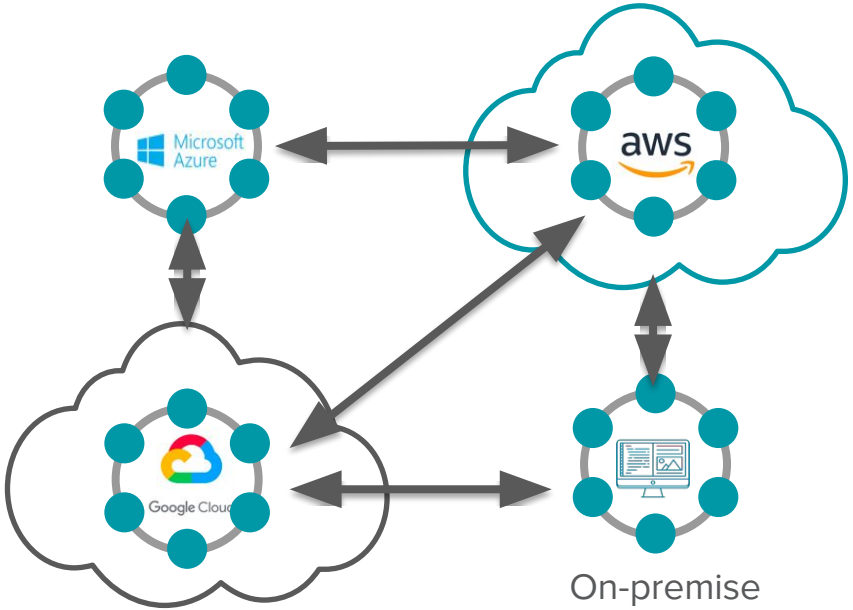


Data Distributed Everywhere

- Geographic Distribution

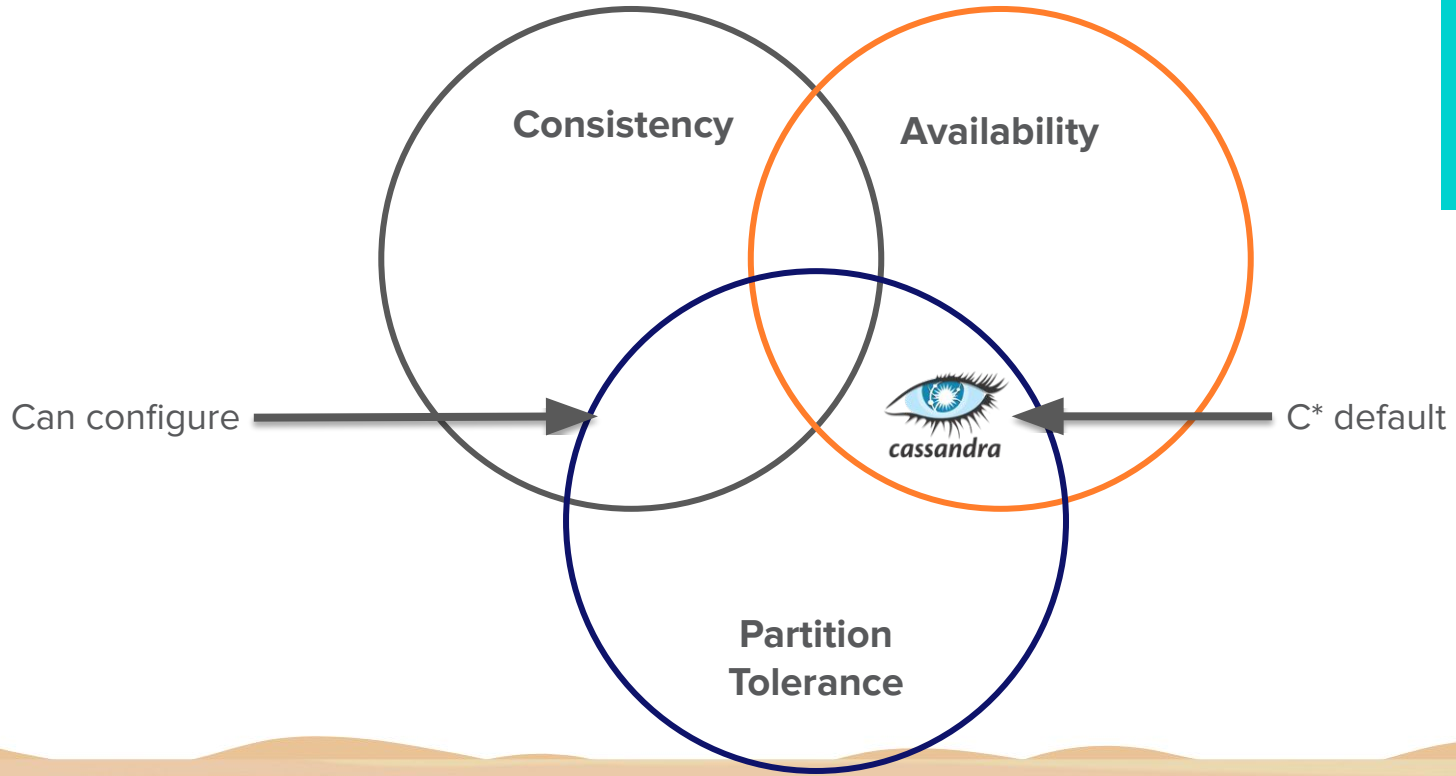


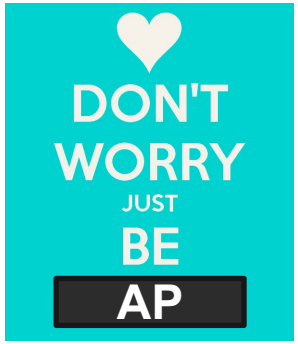
- Hybrid-Cloud and Multi-Cloud



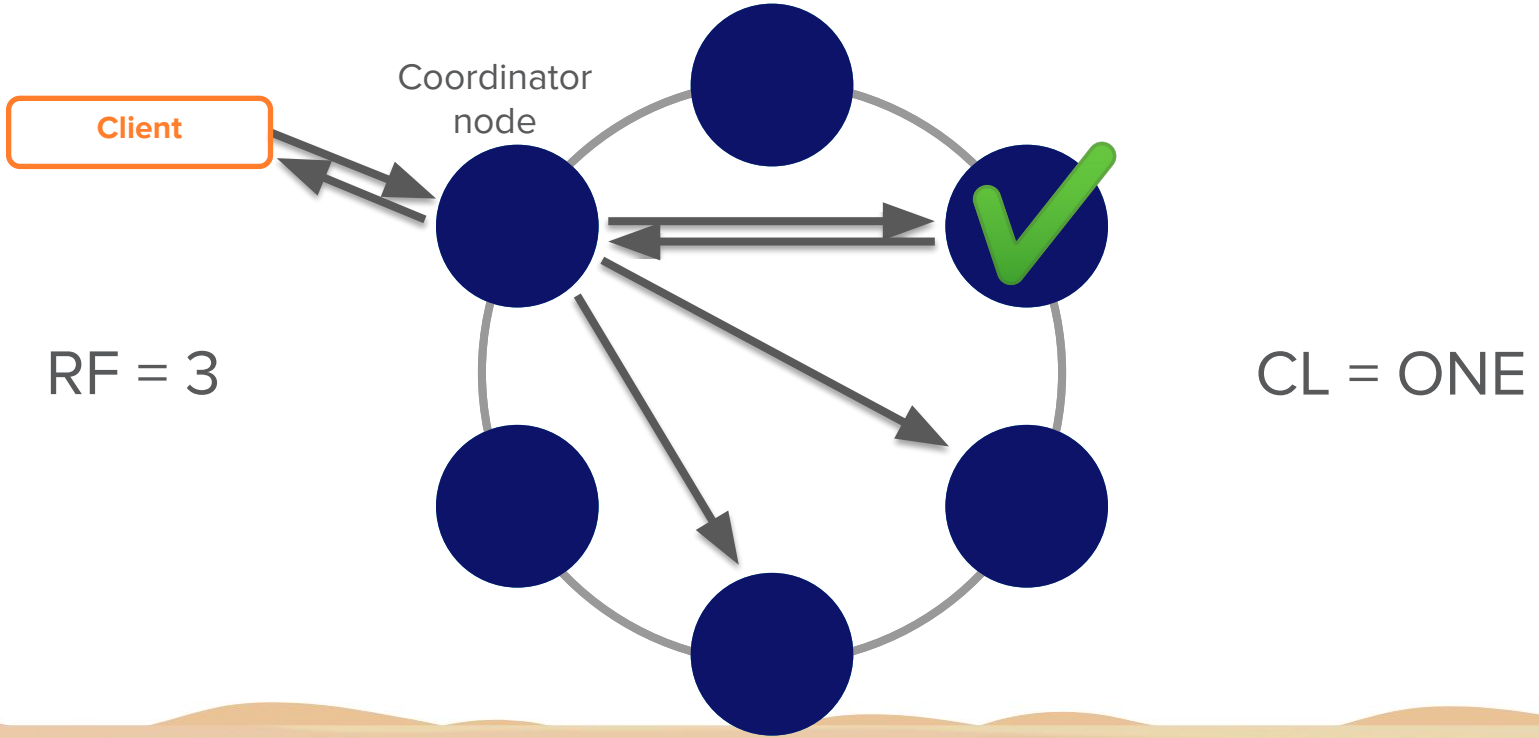
Consistency

CAP Theorem

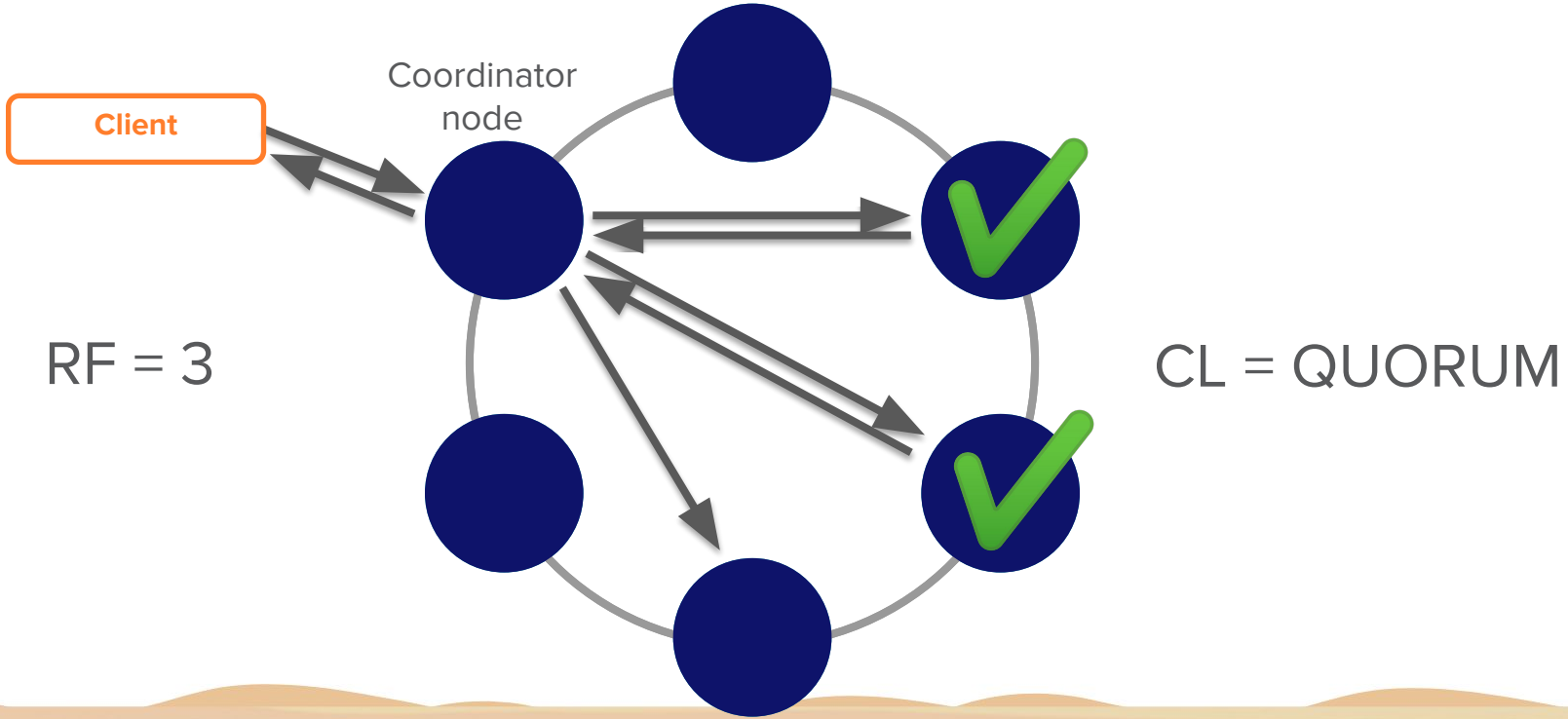

Consistency
Availability
Partition Tolerance
cassandra
Can configure
C* default


DON'T
WORRY
JUST
BE
AP

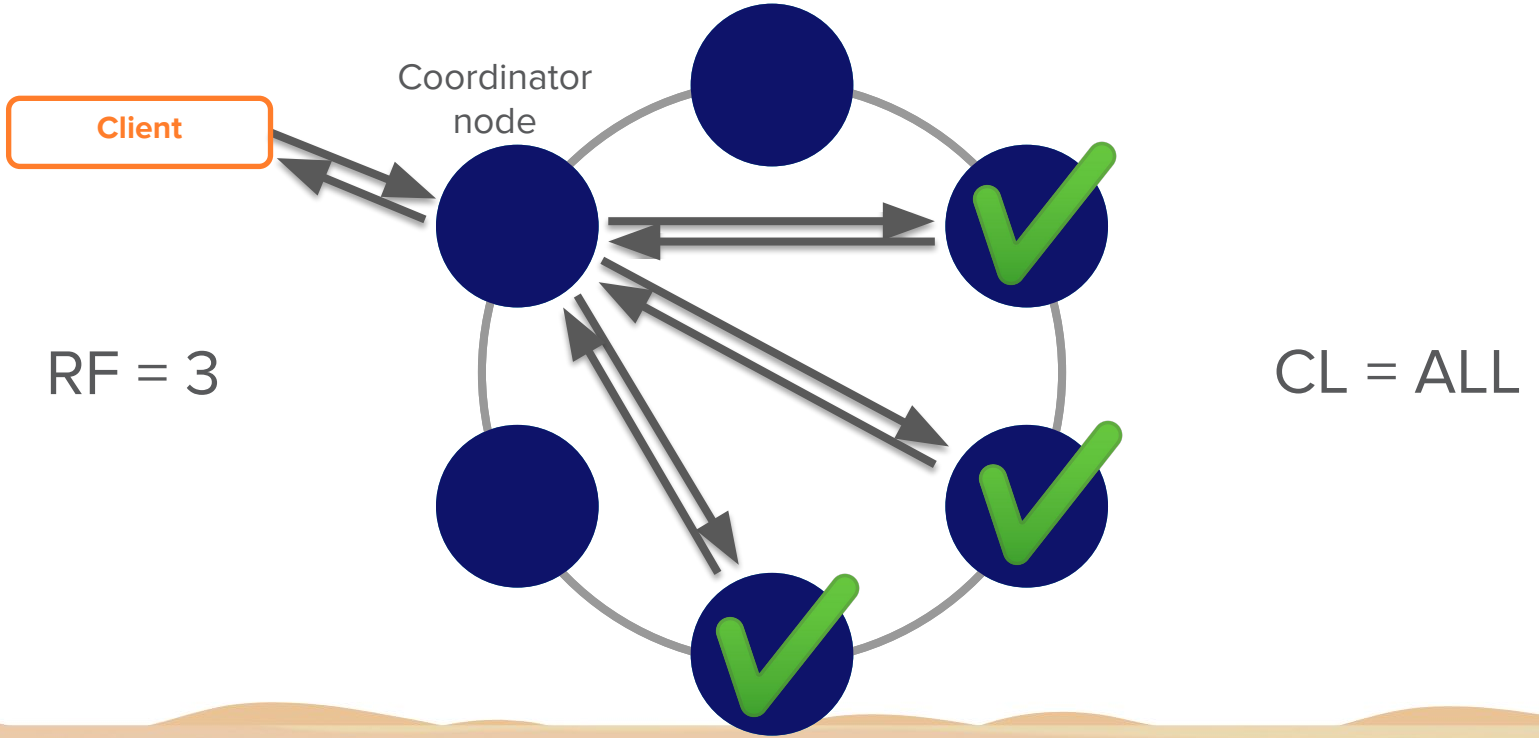
Consistency Levels



Consistency Levels



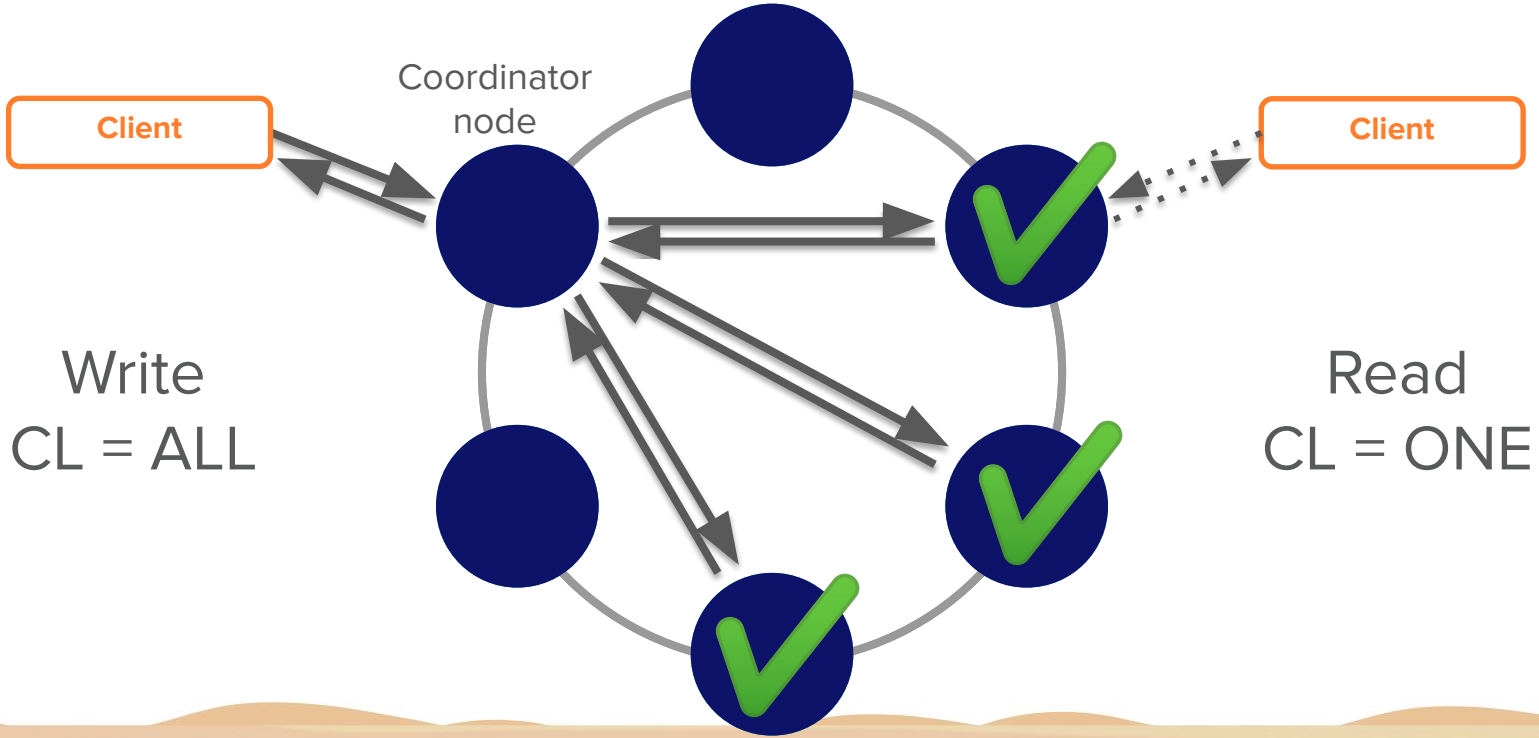
Consistency Levels



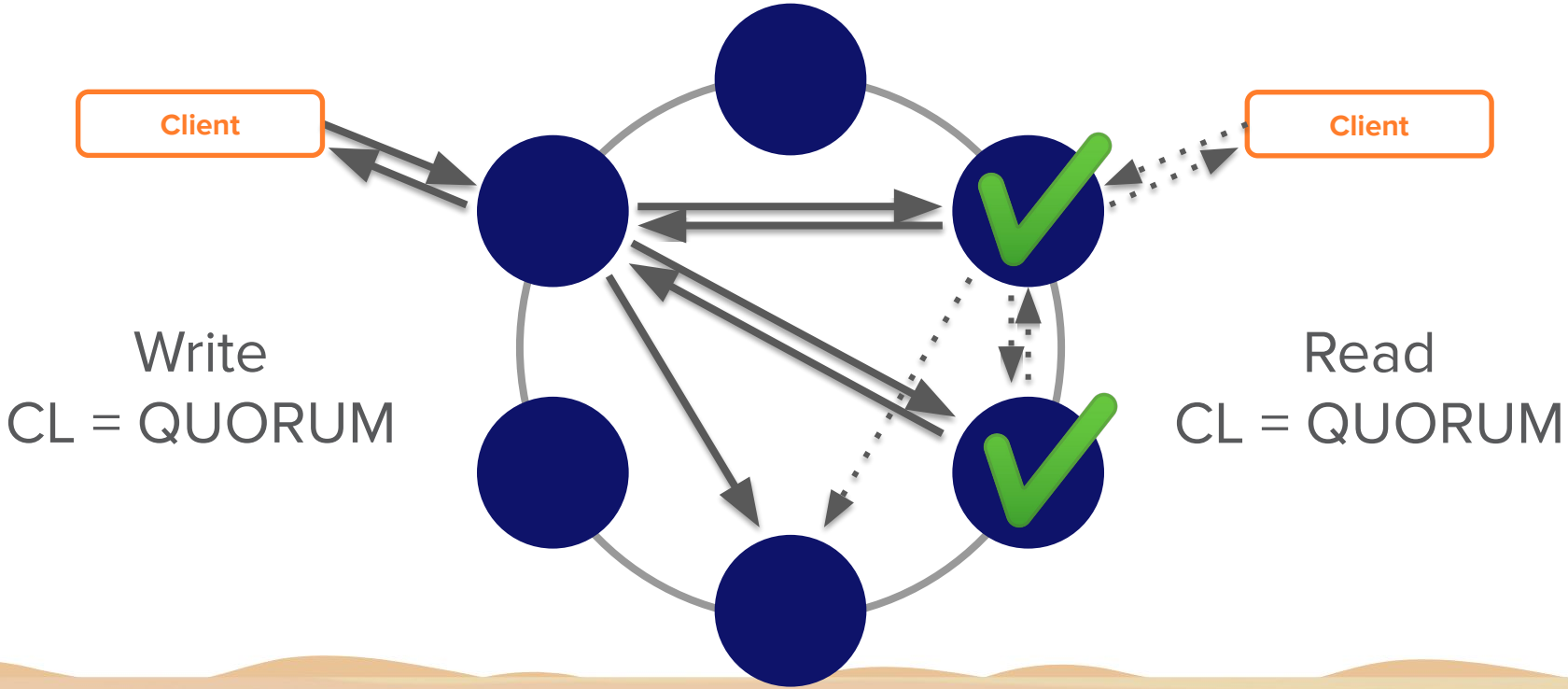
Immediate Consistency

$$CL_{read} + CL_{write} > RF$$

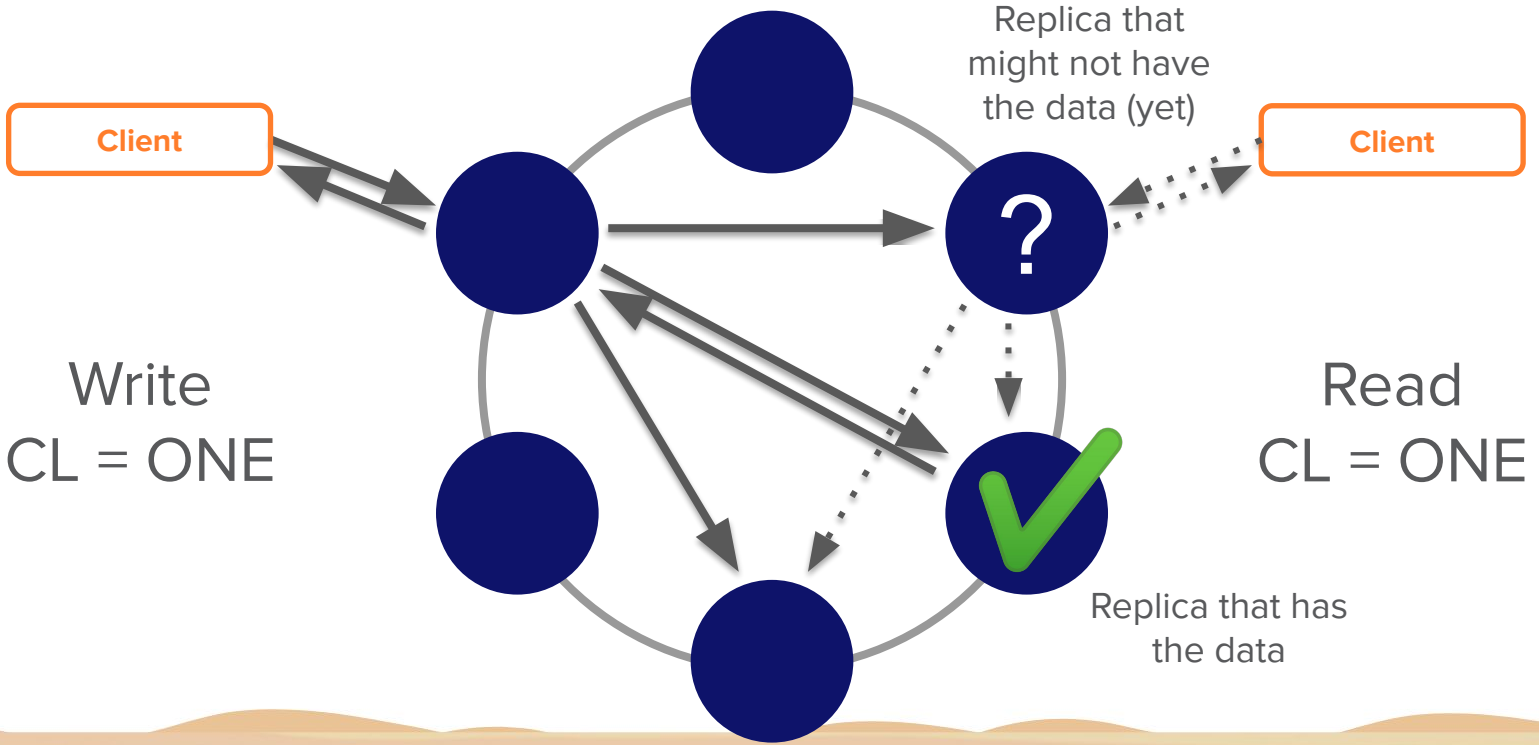
Immediate Consistency – One Way



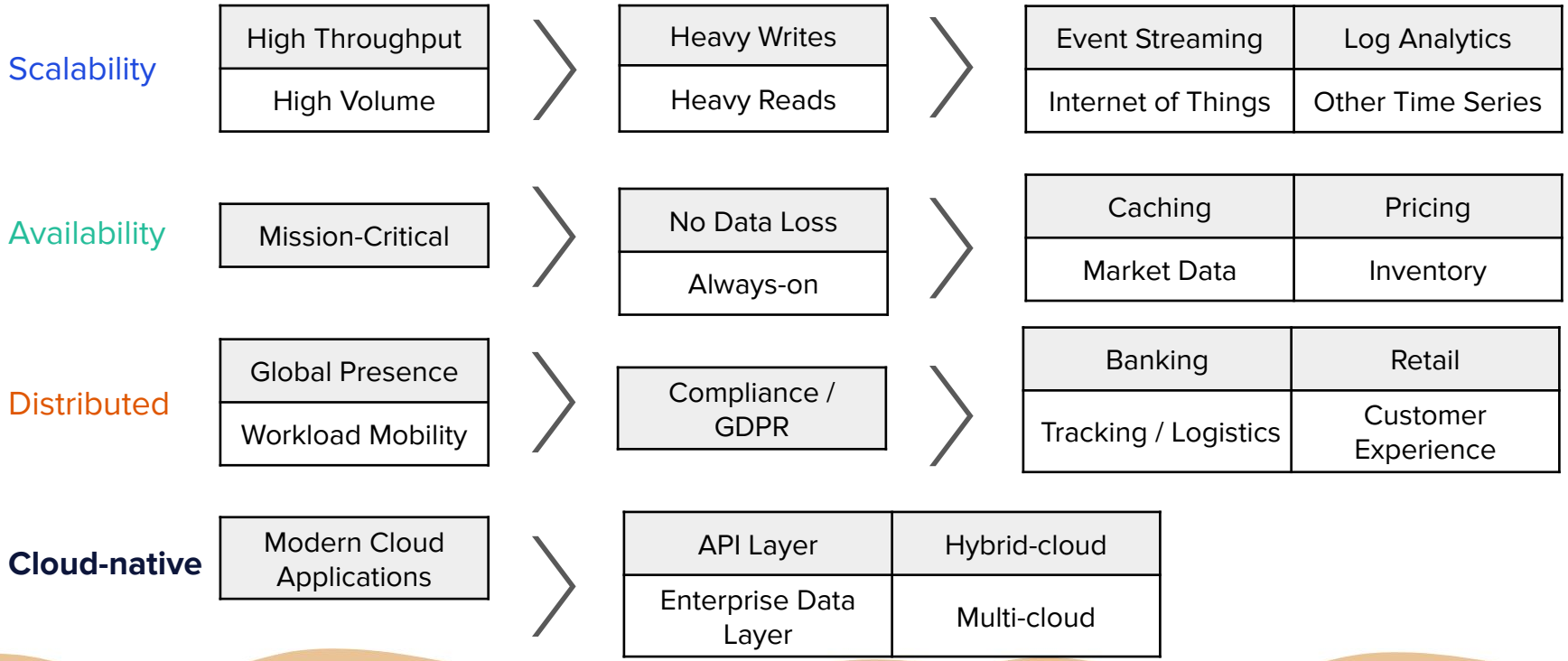
Immediate Consistency – A Better Way



Weak(er) Consistency



Understanding Use Cases



Developer Workshop Series Week 1

- ➔ Bootstrapping
- ➔ Apache Cassandra™ Why, What & When
- ➔ Read and Write path
- ➔ Uber High Level Data Modeling
- ➔ What's NEXT?

Write path

Write Path

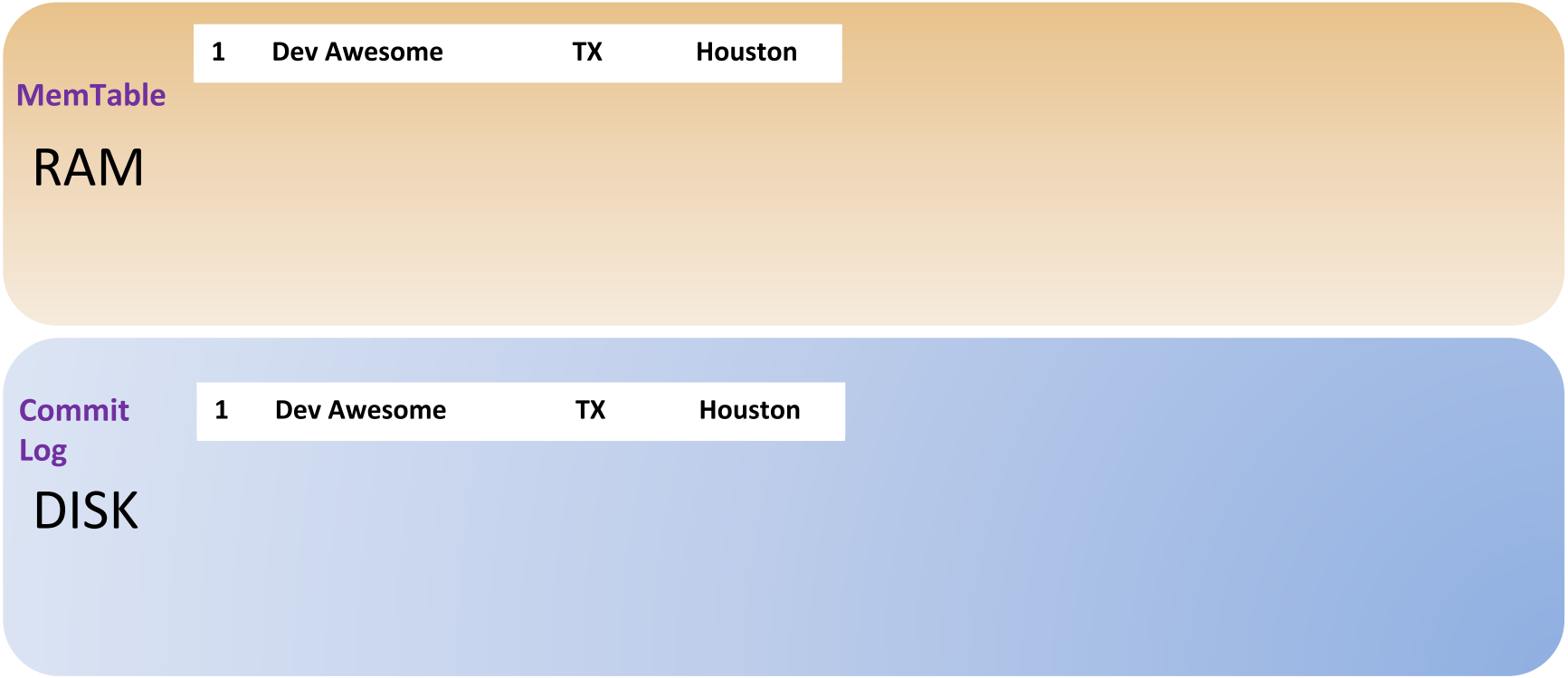
Write Path

RAM

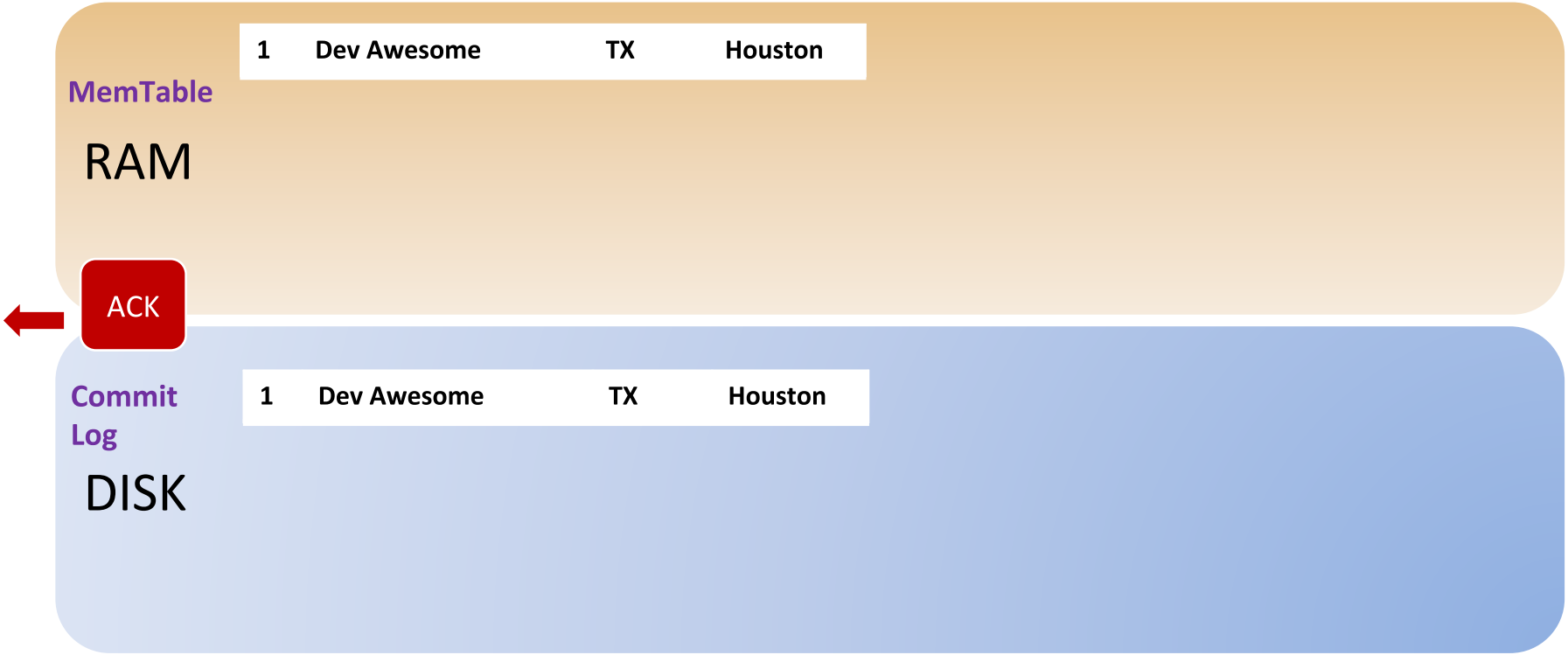
1	Dev Awesome	TX	Houston

DISK

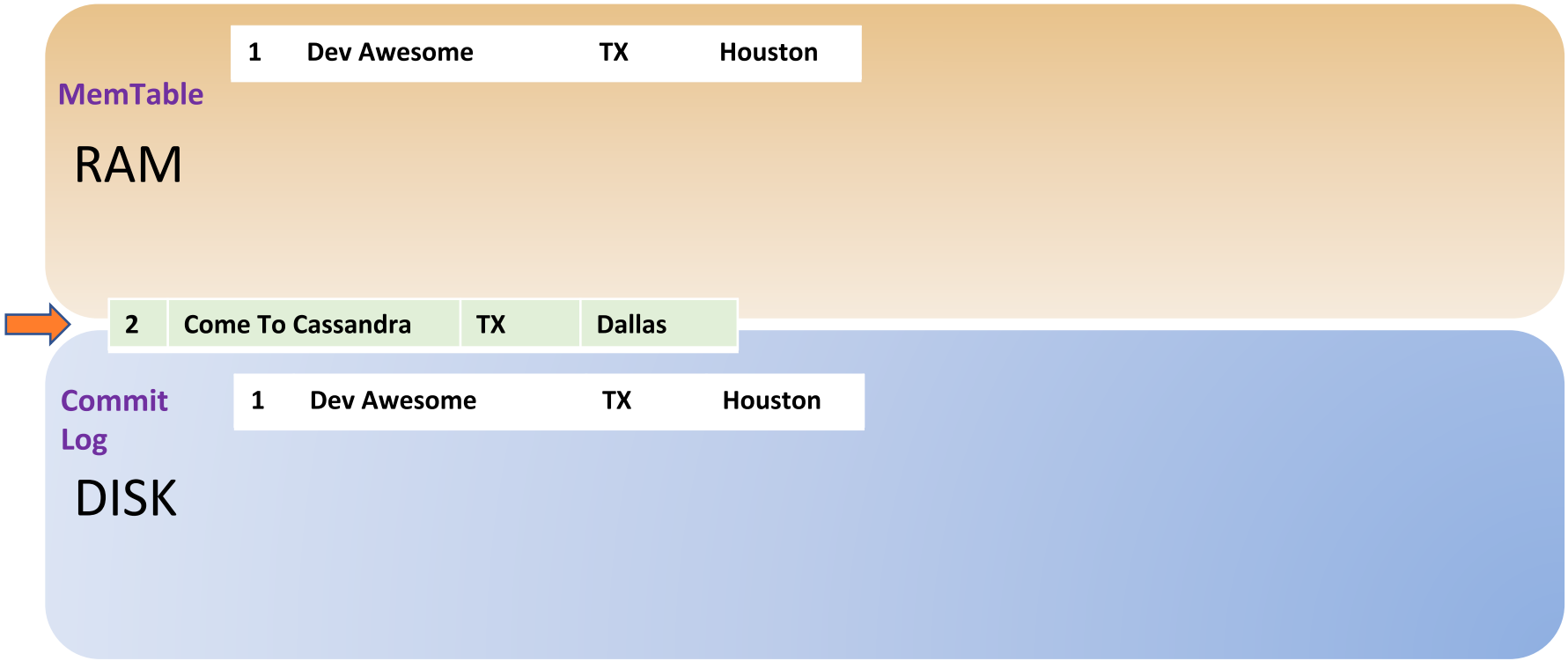
Write Path


MemTable
1 Dev Awesome TX Houston
RAM
Commit Log
1 Dev Awesome TX Houston
DISK

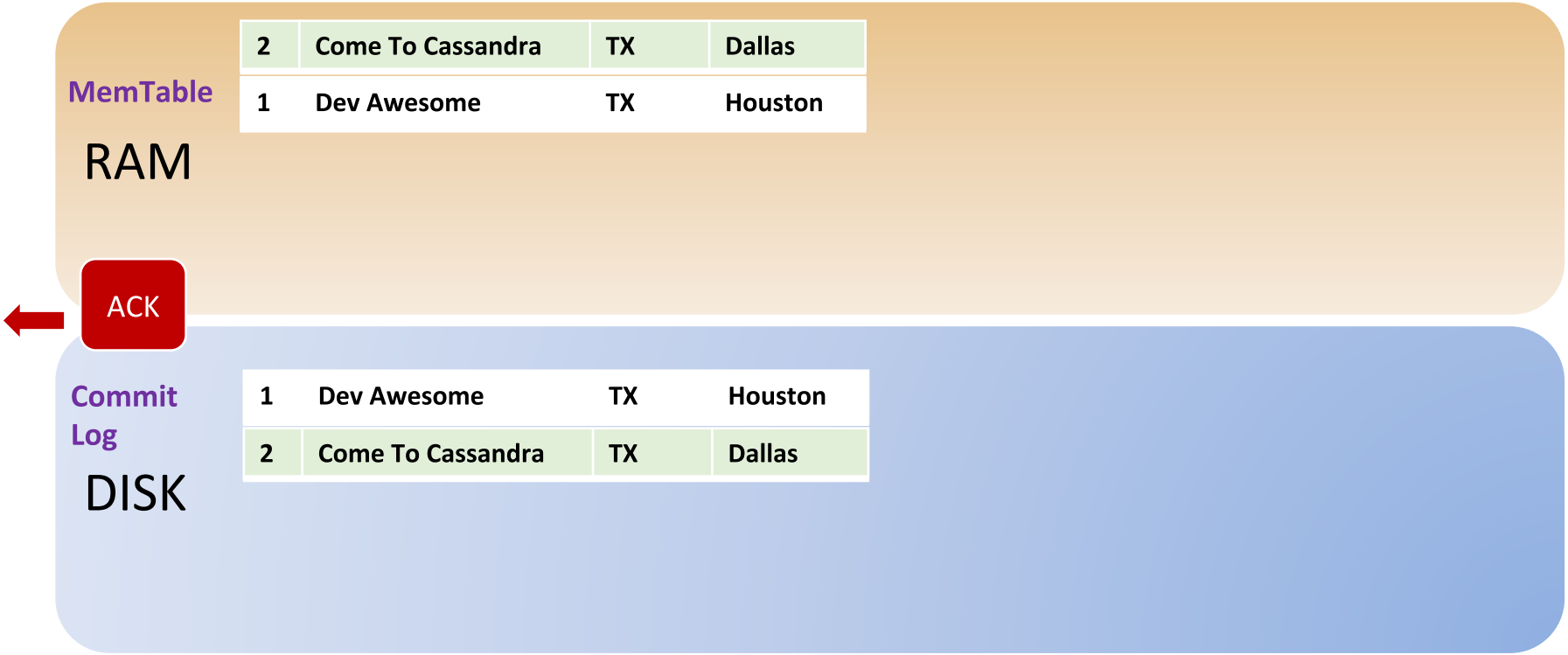
Write Path


1 Dev Awesome TX Houston
MemTable
RAM
ACK
Commit Log
1 Dev Awesome TX Houston
DISK

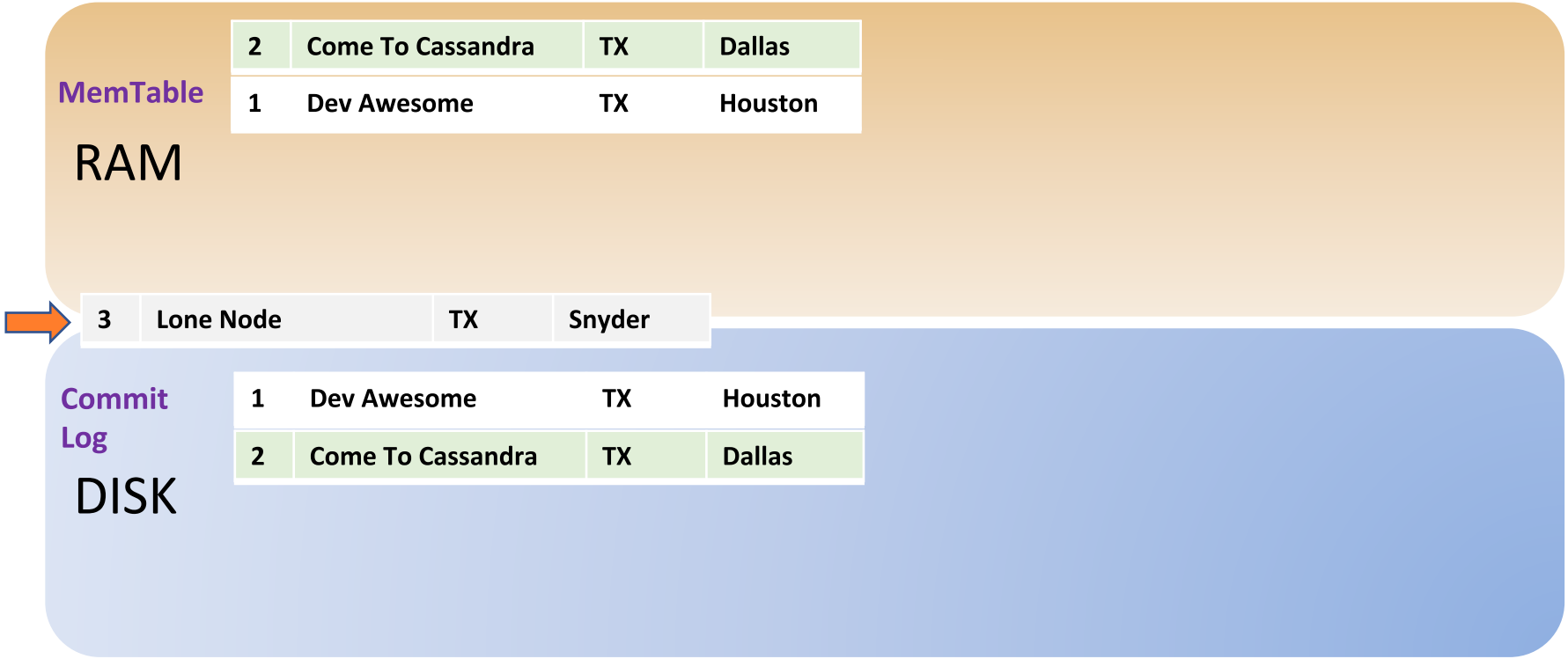
Write Path


MemTable
1 Dev Awesome TX Houston
RAM
2 Come To Cassandra TX Dallas
Commit Log
1 Dev Awesome TX Houston
DISK

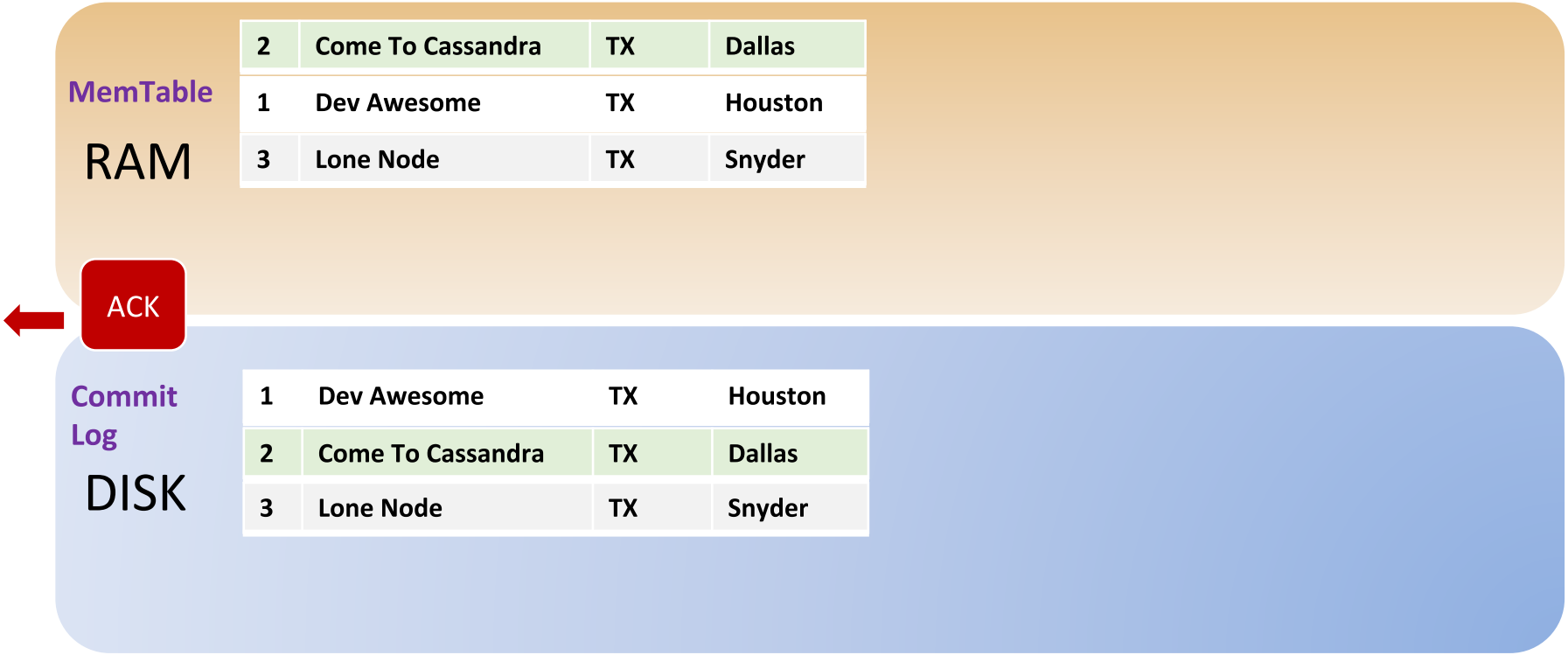
Write Path


MemTable
RAM
2 Come To Cassandra TX Dallas
1 Dev Awesome TX Houston
ACK
Commit Log
DISK
1 Dev Awesome TX Houston
2 Come To Cassandra TX Dallas

Write Path


MemTable
RAM
2 Come To Cassandra TX Dallas
1 Dev Awesome TX Houston
3 Lone Node TX Snyder
Commit Log
DISK
1 Dev Awesome TX Houston
2 Come To Cassandra TX Dallas

Write Path


MemTable
RAM
2 Come To Cassandra TX Dallas
1 Dev Awesome TX Houston
3 Lone Node TX Snyder
ACK
Commit Log
DISK
1 Dev Awesome TX Houston
2 Come To Cassandra TX Dallas
3 Lone Node TX Snyder

Write Path

MemTable

RAM

2	Come To Cassandra	TX	Dallas
1	Dev Awesome	TX	Houston
3	Lone Node	TX	Snyder

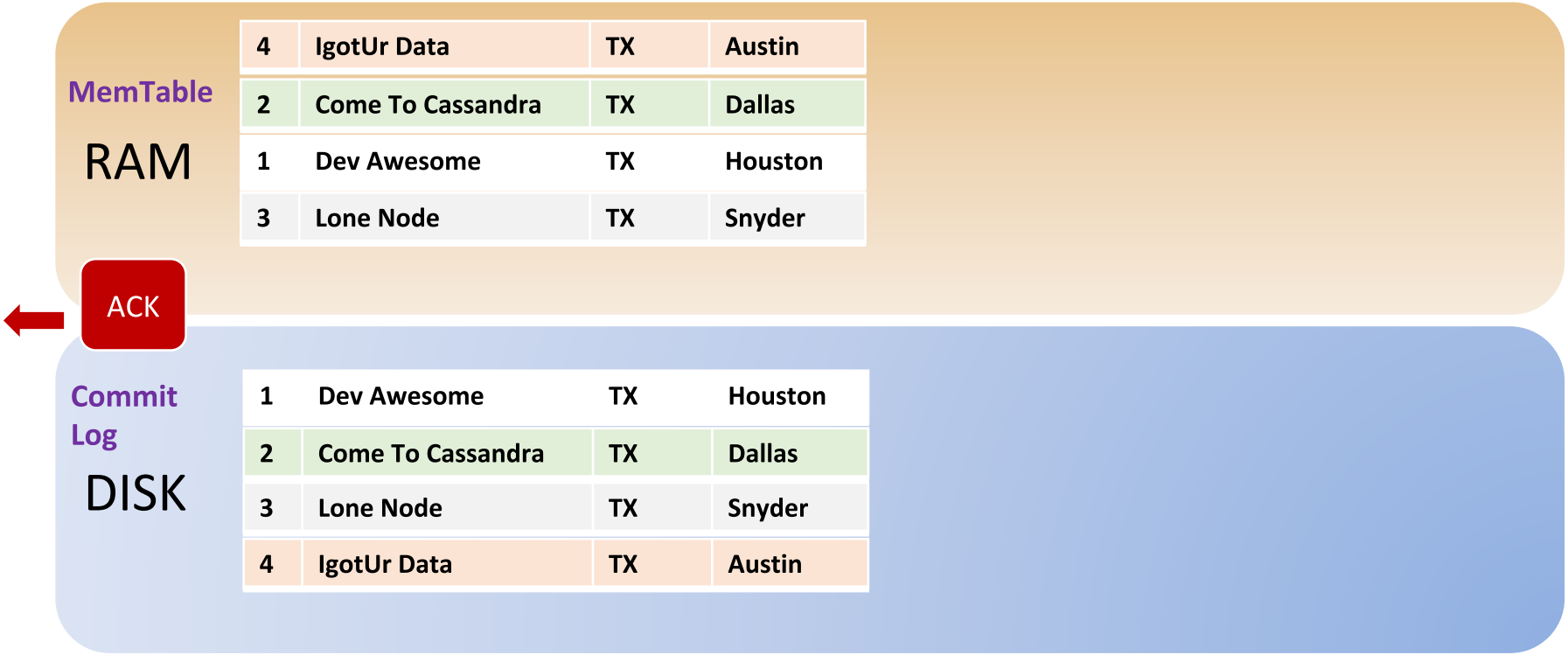
4	IgotUr Data	TX	Austin

Commit Log

DISK

1	Dev Awesome	TX	Houston
2	Come To Cassandra	TX	Dallas
3	Lone Node	TX	Snyder

Write Path


MemTable
RAM
4 IgotUr Data TX Austin
2 Come To Cassandra TX Dallas
1 Dev Awesome TX Houston
3 Lone Node TX Snyder
ACK
Commit Log
DISK
1 Dev Awesome TX Houston
2 Come To Cassandra TX Dallas
3 Lone Node TX Snyder
4 IgotUr Data TX Austin

Write Path

MemTable

RAM

4	IgotUr Data	TX	Austin
2	Come To Cassandra	TX	Dallas
1	Dev Awesome	TX	Houston
3	Lone Node	TX	Snyder

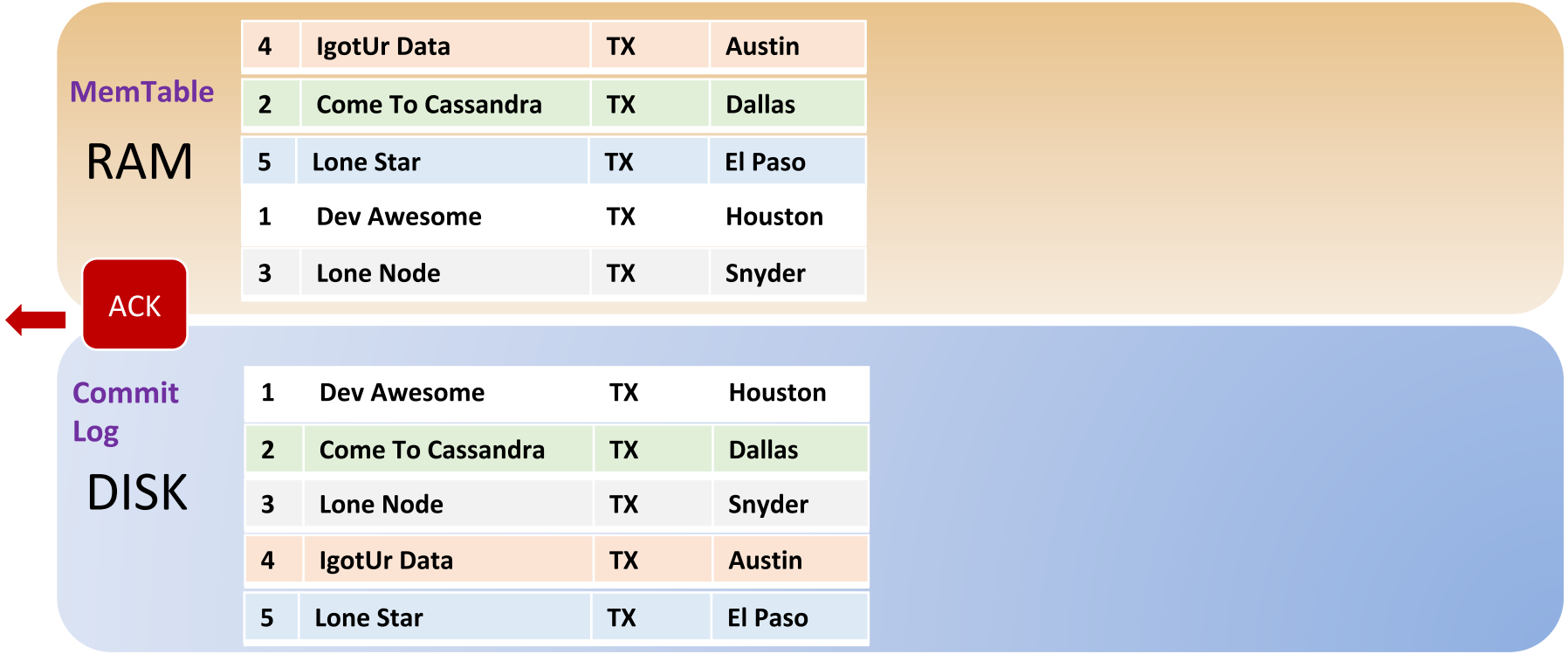
5	Lone Star	TX	El Paso

Commit Log

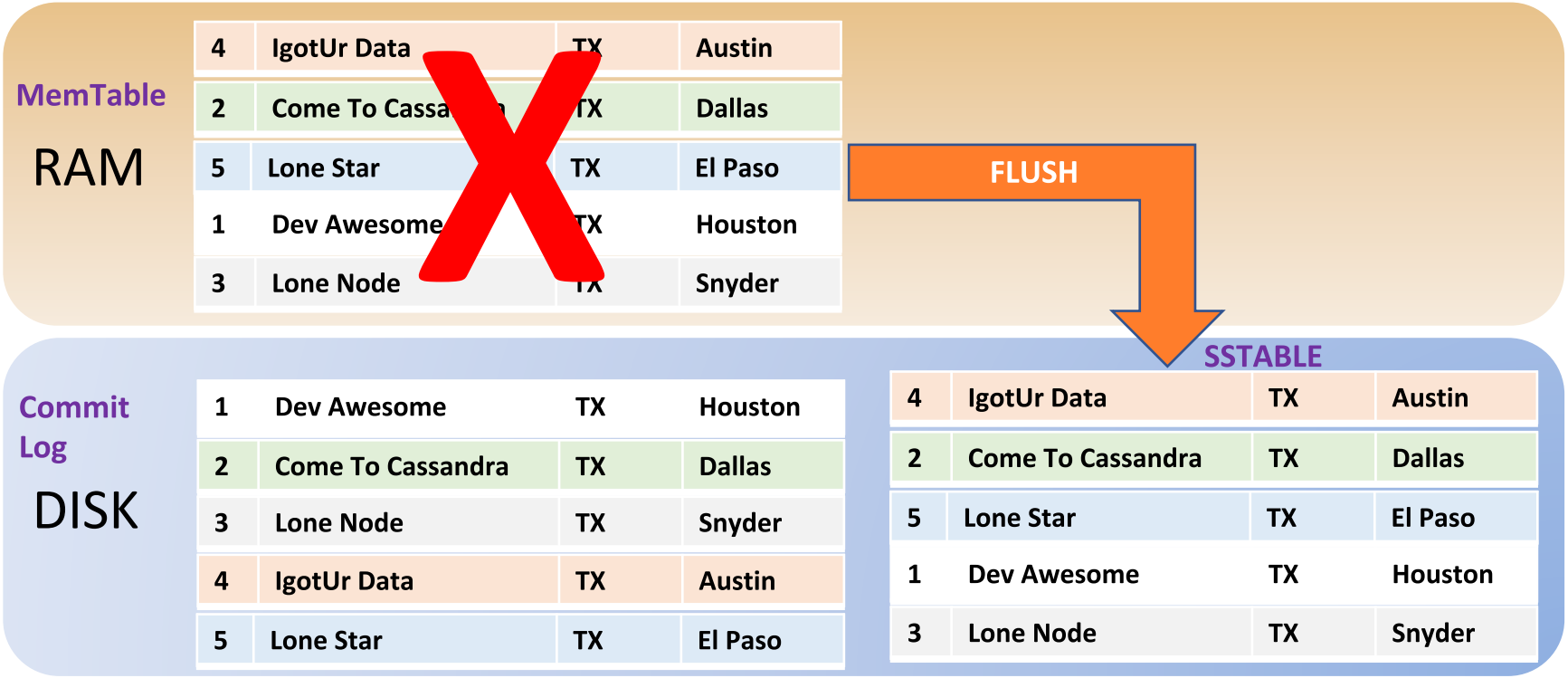
DISK

1	Dev Awesome	TX	Houston
2	Come To Cassandra	TX	Dallas
3	Lone Node	TX	Snyder
4	IgotUr Data	TX	Austin

Write Path


MemTable
RAM
4 IgotUr Data TX Austin
2 Come To Cassandra TX Dallas
5 Lone Star TX El Paso
1 Dev Awesome TX Houston
3 Lone Node TX Snyder
ACK
Commit Log
DISK
1 Dev Awesome TX Houston
2 Come To Cassandra TX Dallas
3 Lone Node TX Snyder
4 IgotUr Data TX Austin
5 Lone Star TX El Paso

Write Path


MemTable
RAM
4 IgotUr Data TX Austin
2 Come To Cassa TX Dallas
5 Lone Star TX El Paso
1 Dev Awesome TX Houston
3 Lone Node TX Snyder
FLUSH
Commit Log
DISK
1 Dev Awesome TX Houston
2 Come To Cassandra TX Dallas
3 Lone Node TX Snyder
4 IgotUr Data TX Austin
5 Lone Star TX El Paso
SSTABLE
4 IgotUr Data TX Austin
2 Come To Cassandra TX Dallas
5 Lone Star TX El Paso
1 Dev Awesome TX Houston
3 Lone Node TX Snyder

Write Path

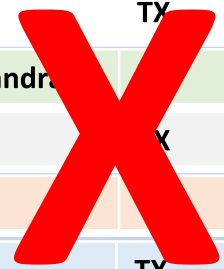
MemTable

RAM

Commit Log

DISK

1	Dev Awesome	TX	Houston
2	Come To Cassandra		Dallas
3	Lone Node		Snyder
4	IgotUr Data		Austin
5	Lone Star	TX	El Paso



SSTABLE

4	IgotUr Data	TX	Austin
2	Come To Cassandra	TX	Dallas
5	Lone Star	TX	El Paso
1	Dev Awesome	TX	Houston
3	Lone Node	TX	Snyder

Write Path

MemTable

RAM

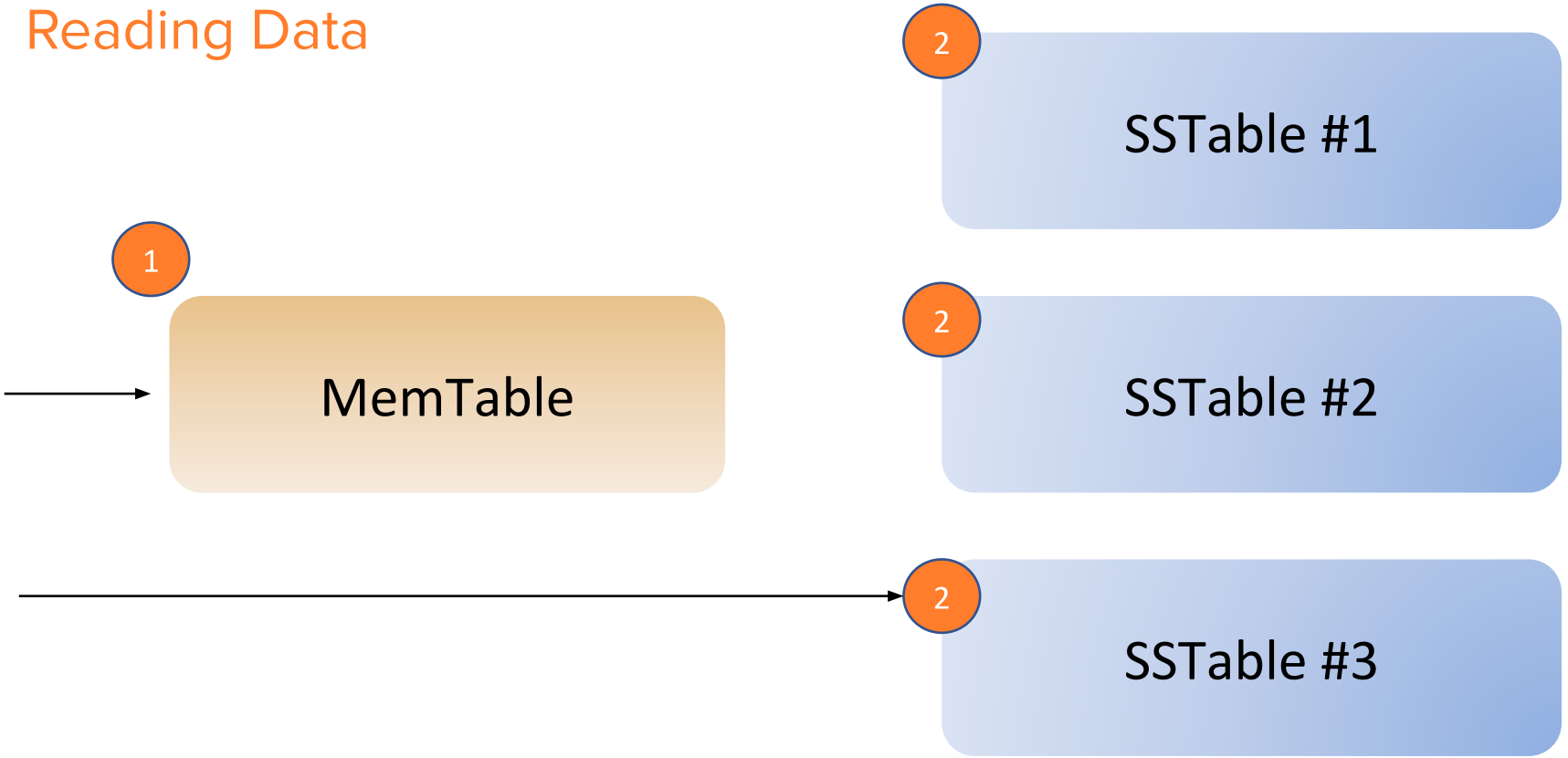
DISK

SSTABLE (IMMUTABLE)

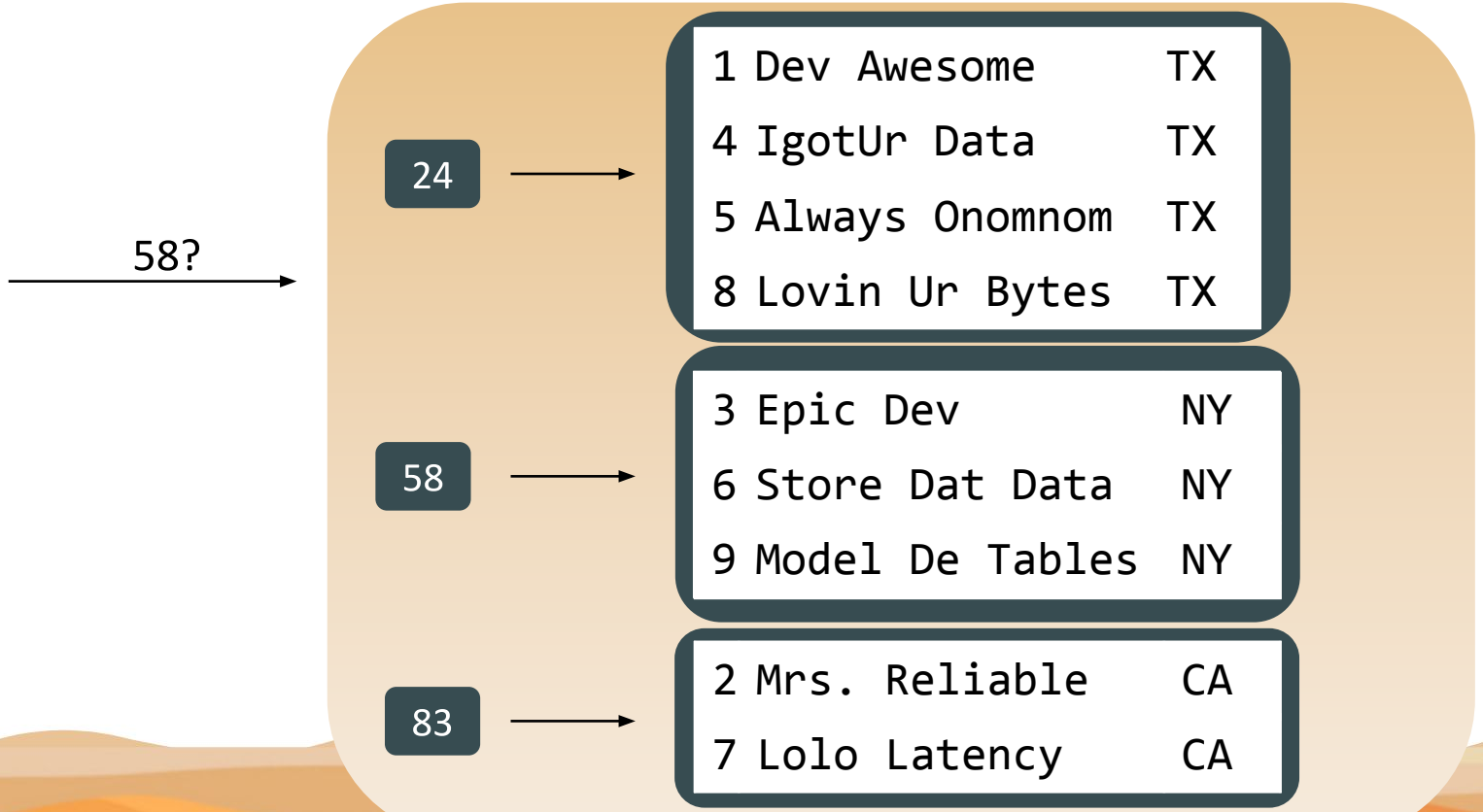
4	IgotUr Data	TX	Austin
2	Come To Cassandra	TX	Dallas
5	Lone Star	TX	El Paso
1	Dev Awesome	TX	Houston
3	Lone Node	TX	Snyder

Read path

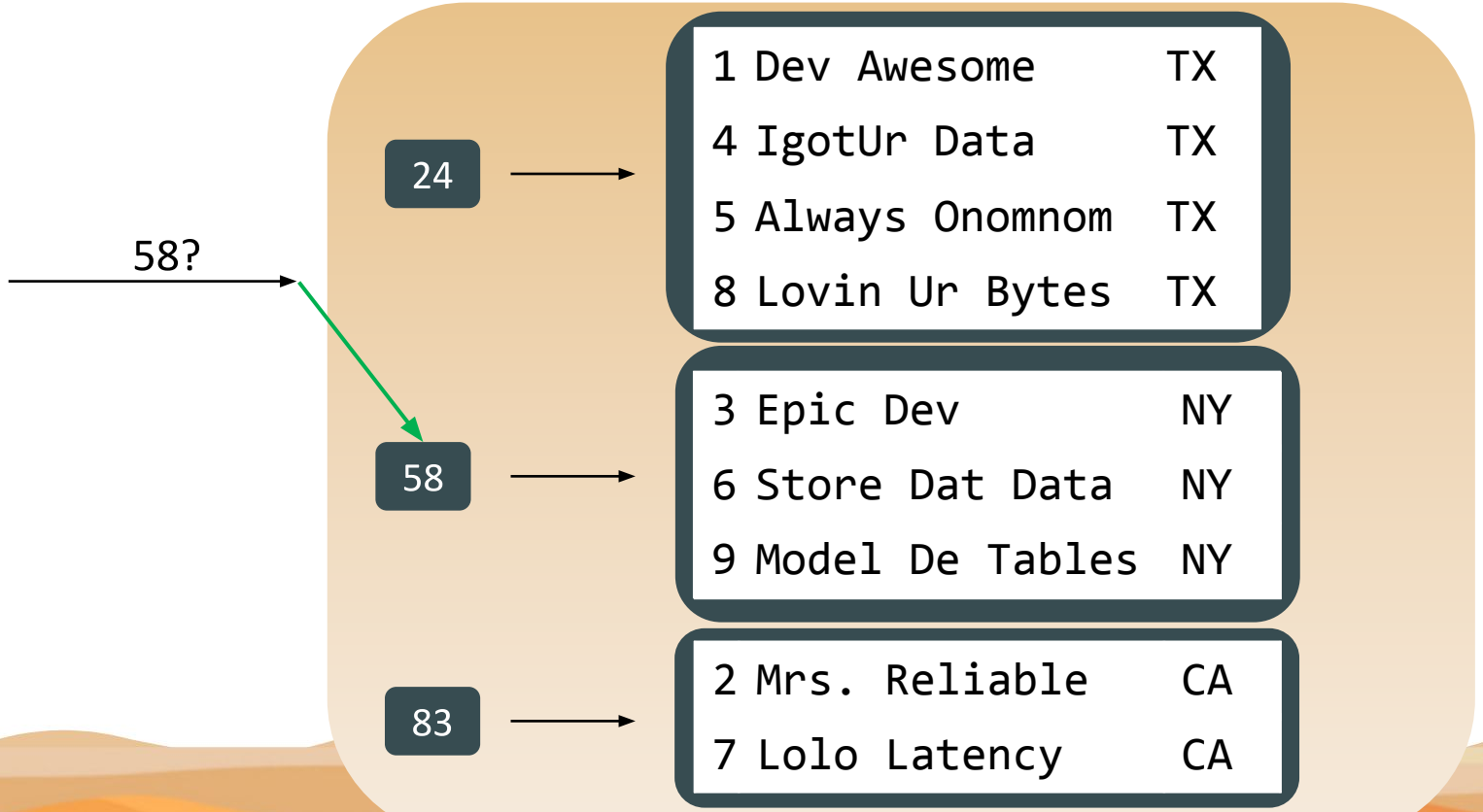
Reading Data



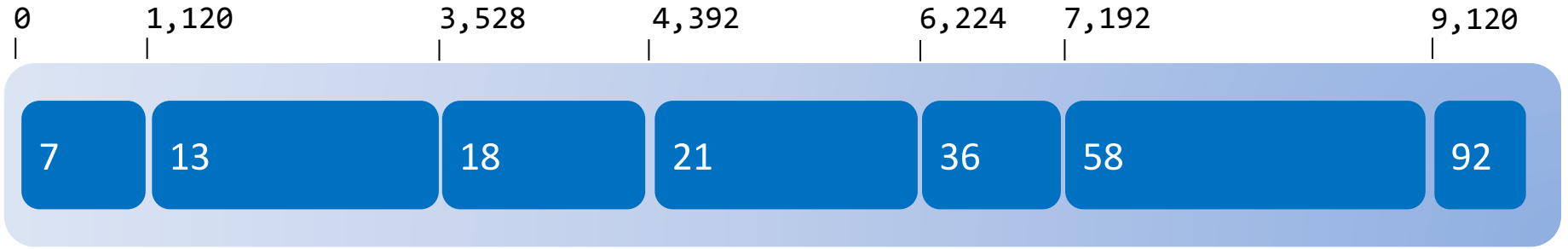
Reading a MemTable



Reading a MemTable



Reading a SSTable

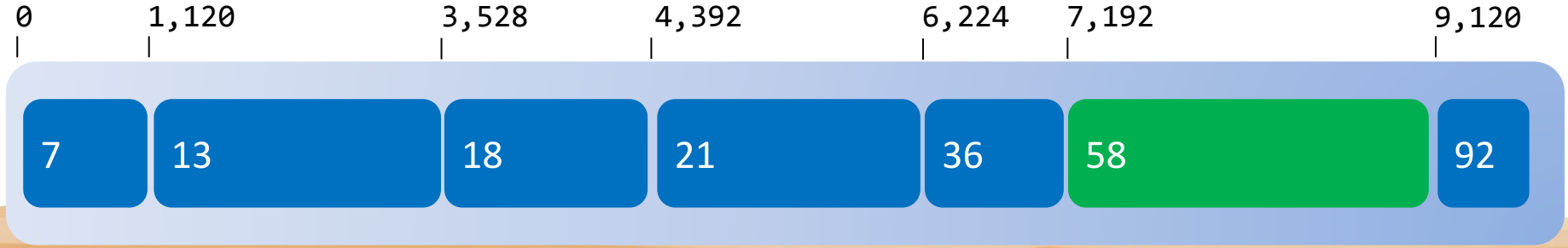


- A SSTable holds ordered partitions
- A partition can be split in multiple SSTables
- We can mark offset of each partition

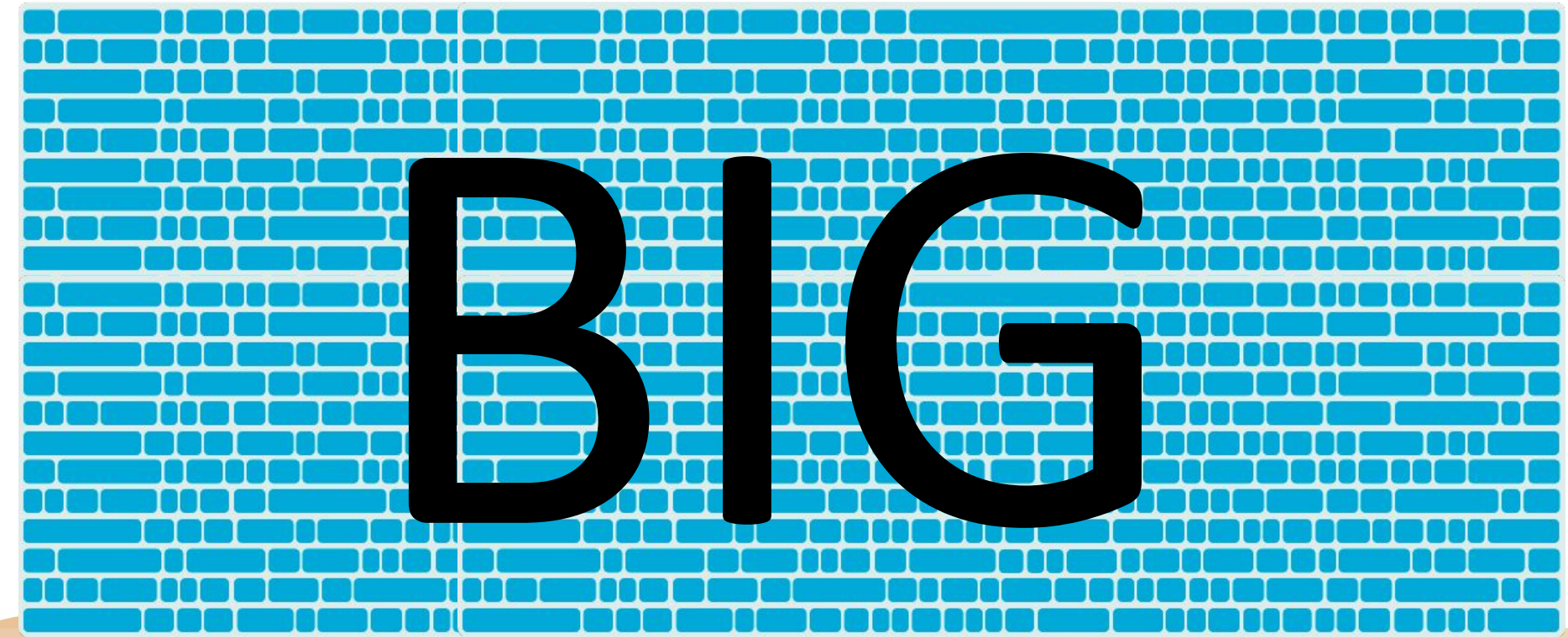
Partition Index (on Disk)

Token	Byte Offset
7	0
13	1,120
18	3,528
21	4,392
36	6,224
58	7,192
92	9,120

58?



...but a SSTABLE is ...

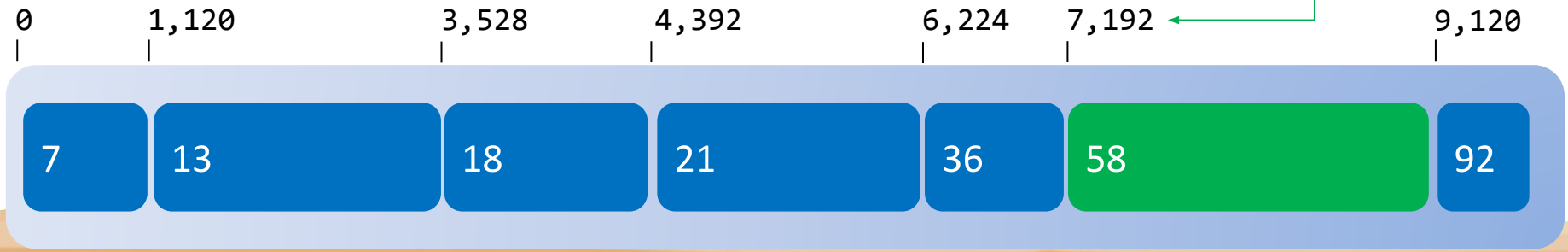


Partition Summary (RAM)

Token	Byte Offset
	0
	24
56 - 100	40

58?

	Token	Byte Offset
0 →	7	0
	13	1,120
	18	3,528
24 →	21	4,392
	36	6,224
40 →	58	7,192
	92	9,120



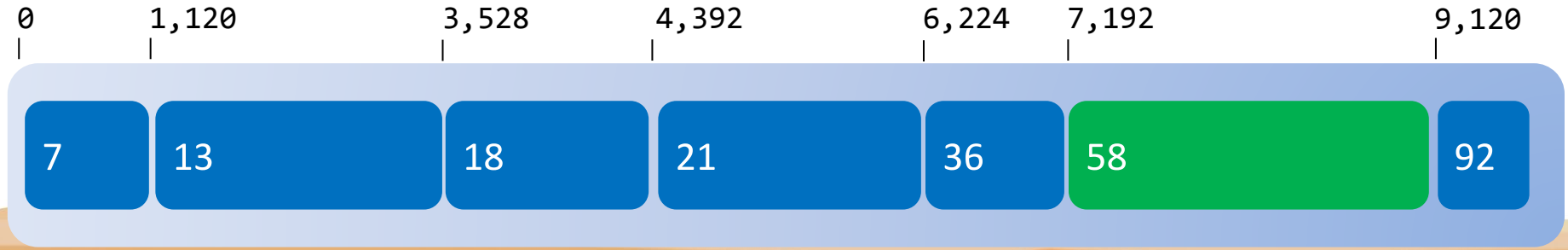
Key Cache (RAM)

Token	Byte Offset
	0
	24
	40

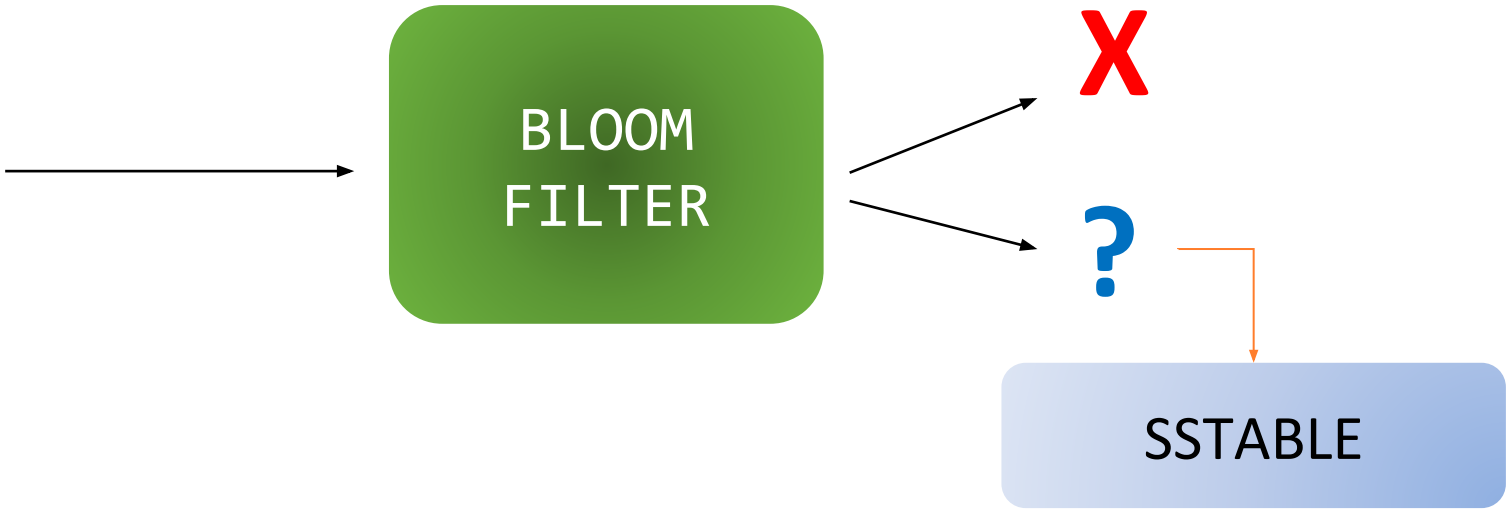
Token	Byte Offset
7	0
13	1,120
18	3,528
21	4,392
36	6,224
58	7,192
92	9,120

58?

Token	Byte Offset
58	7192



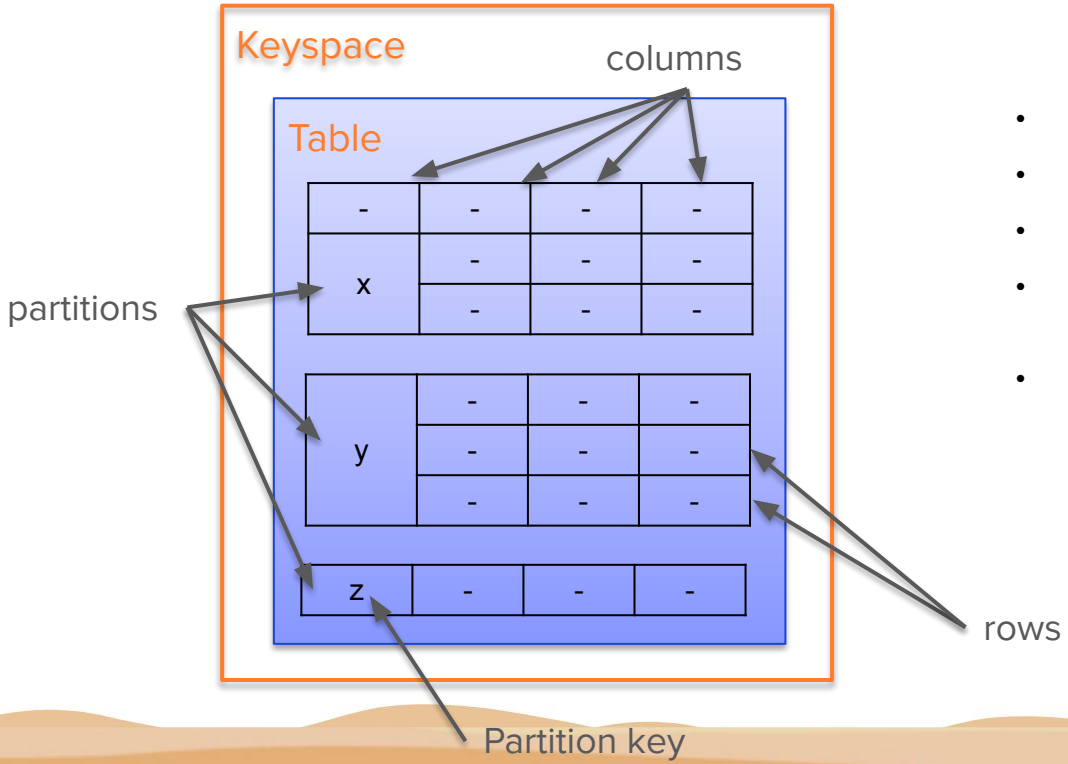
Bloom Filter



Developer Workshop Series Week 1

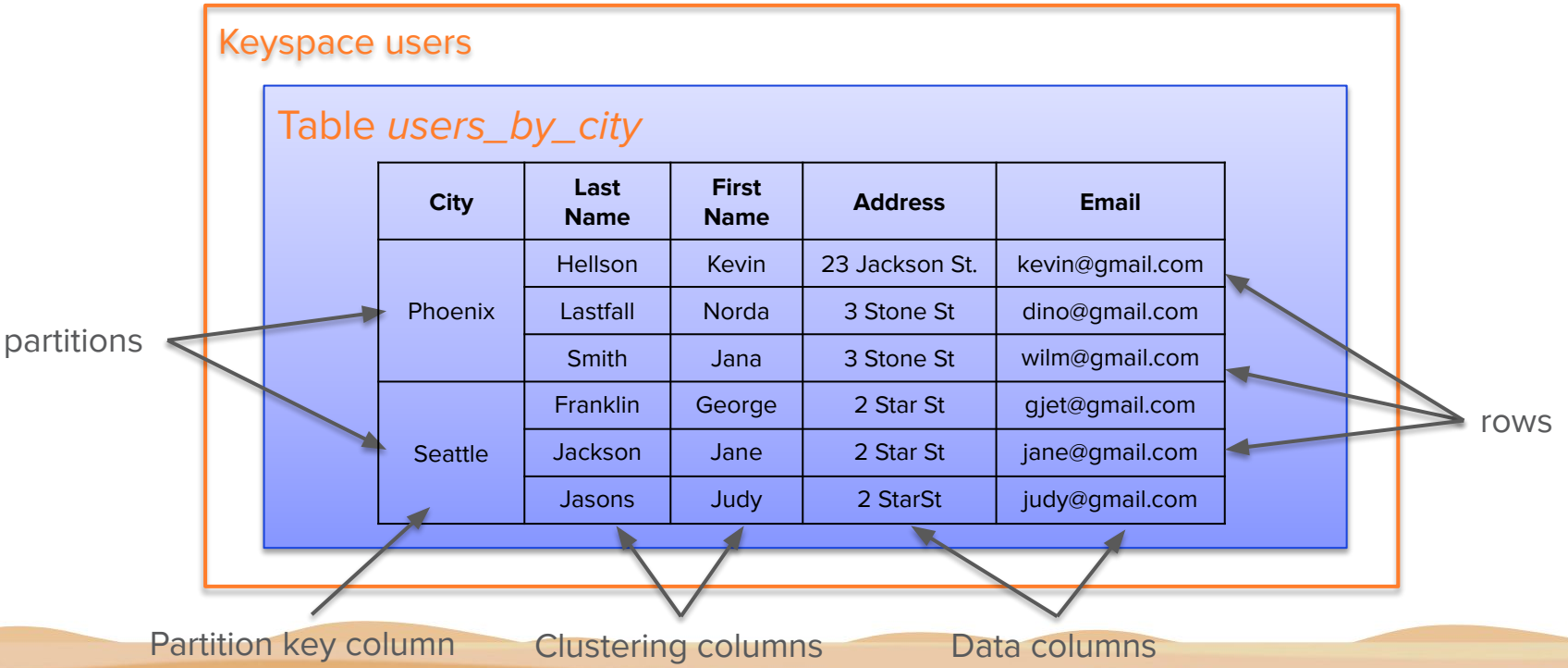
- ➔ Bootstrapping
- ➔ Apache Cassandra™ Why, What & When
- ➔ Read and Write path
- ➔ Uber High Level Data Modeling
- ➔ What's NEXT?

Cassandra Structure - Partition



- 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 – Users organized by city



City	Last Name	First Name	Address	Email
Phoenix	Hellson	Kevin	23 Jackson St.	kevin@gmail.com
	Lastfall	Norda	3 Stone St	dino@gmail.com
	Smith	Jana	3 Stone St	wilm@gmail.com
Seattle	Franklin	George	2 Star St	gjet@gmail.com
	Jackson	Jane	2 Star St	jane@gmail.com
	Jasons	Judy	2 StarSt	judy@gmail.com

Tables Hold Many Partitions

City	Last Name	First Name	Address	Email
Phoenix	Hellson	Kevin	23 Jackson St.	kevin@gmail.com
	Lastfall	Norda	3 Stone St	dino@gmail.com
	Smith	Jana	3 Stone St	wilm@gmail.com

Table *users_by_city*

Tables Hold Many Partitions

City	Last Name	First Name	Address	Email
Seattle	Franklin	George	2 Star St	gjet@gmail.com
	Jackson	Jane	2 Star St	jane@gmail.com
	Jasons	Judy	2 StarSt	judy@gmail.com

Table *users_by_city*

City	Last Name	First Name	Address	Email
Phoenix	----	----	----	----
	----	----	----	----
	----	----	----	----

Tables Hold Many Partitions

City	Last Name	First Name	Address	Email
Charlotte	Azrael	Chris	5 Blue St	chris@gmail.com
	Stilson	Brainy	7 Azure ln	brain@gmail.com
	Smith	Cristina	4 Teal Cir	clu@gmail.com
	Sage	Grant	9 Royal St	grant@gmail.com
	Seterson	Peter	2 Navy Ct	peter@gmail.com

Table *users_by_city*

City	Last Name	First Name	Address	Email
Phoenix	----	----	----	----
	----	----	----	----
	----	----	----	----

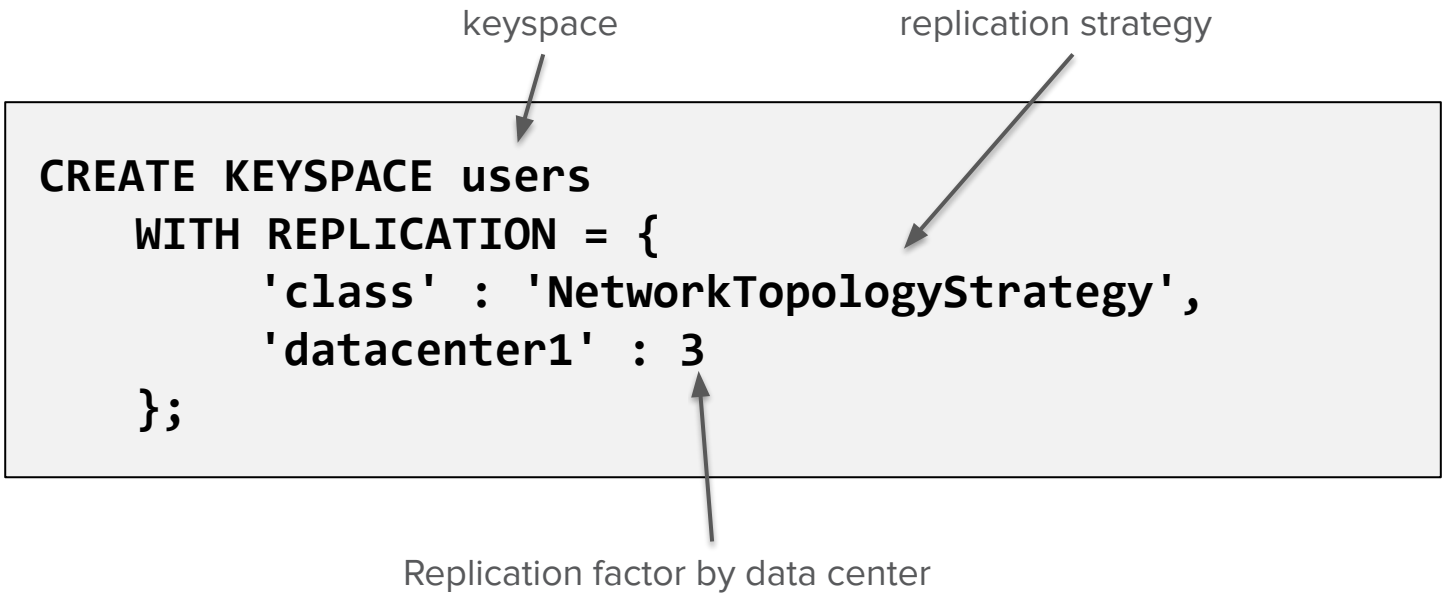
Seattle	----	----	----	----
	----	----	----	----
	----	----	----	----

Tables Hold Many Partitions

Table *users_by_city*

City	Last Name	First Name	Address	Email
Phoenix	----	----	----	----
	----	----	----	----
	----	----	----	----
Seattle	----	----	----	----
	----	----	----	----
	----	----	----	----
Charlotte	----	----	----	----
	----	----	----	----
	----	----	----	----
	----	----	----	----
	----	----	----	----

Creating a Keyspace in CQL



Creating a Table in CQL