Administration & Operations

Architecture

AEROSPIKE

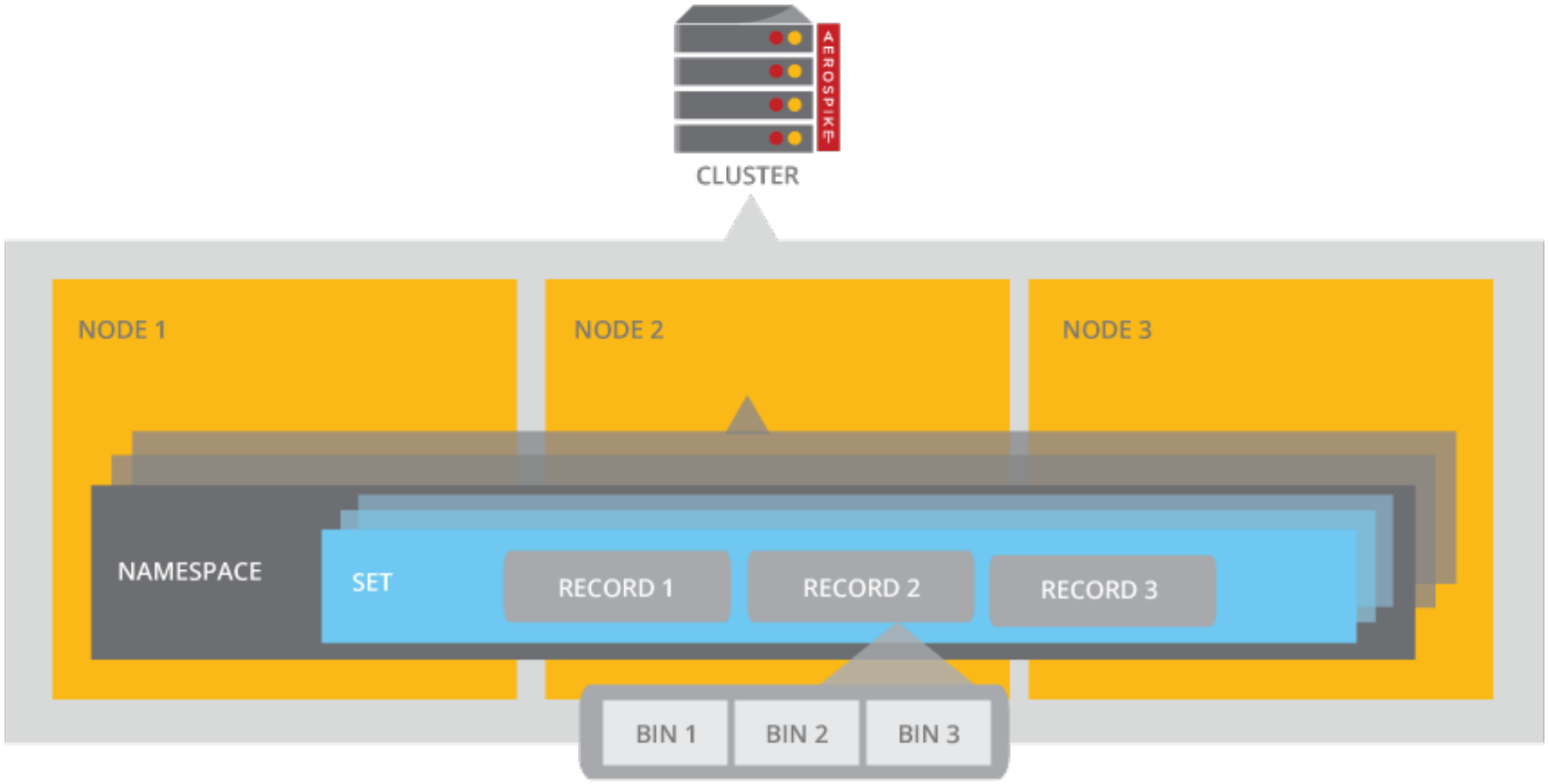
Objectives

This module covers the following:

- Data hierarchy.
- High level architecture.
- Data partitioning.

Data Hierarchy

Data Hierarchy



Nodes

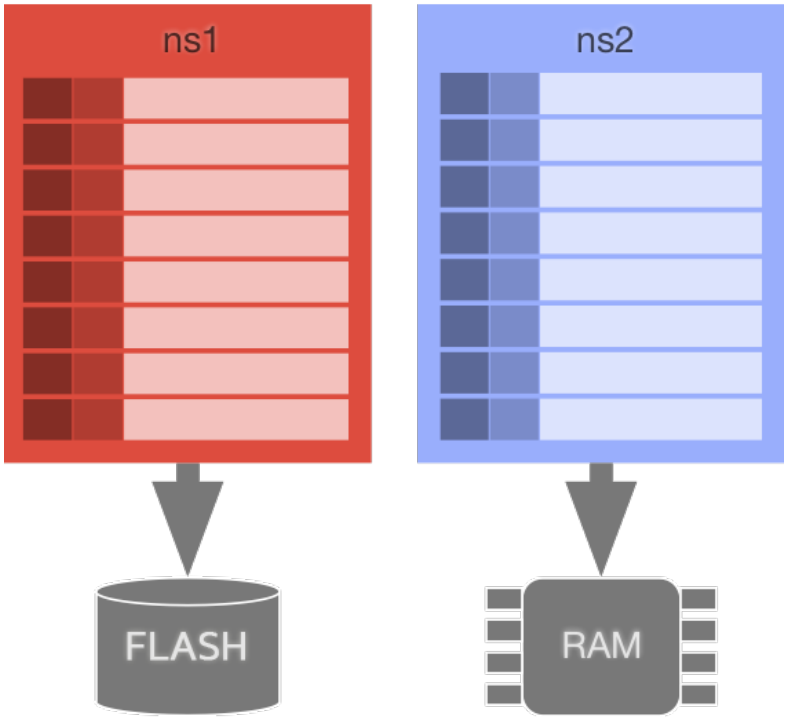
- Each node should have **identical hardware**.
- Should have **identical configuration**.
- Data (and their associated traffic) will be **evenly balanced** across the nodes.
- Big **differences** between nodes implies a **problem**.
- Node **capacity** should take into account **node failure** patterns.

If you have an 8 node cluster distributed evenly on 4 racks, make sure that 6 nodes can handle the data and traffic volume in the event you lose a single rack.

Namespace

Similar to a table space or database

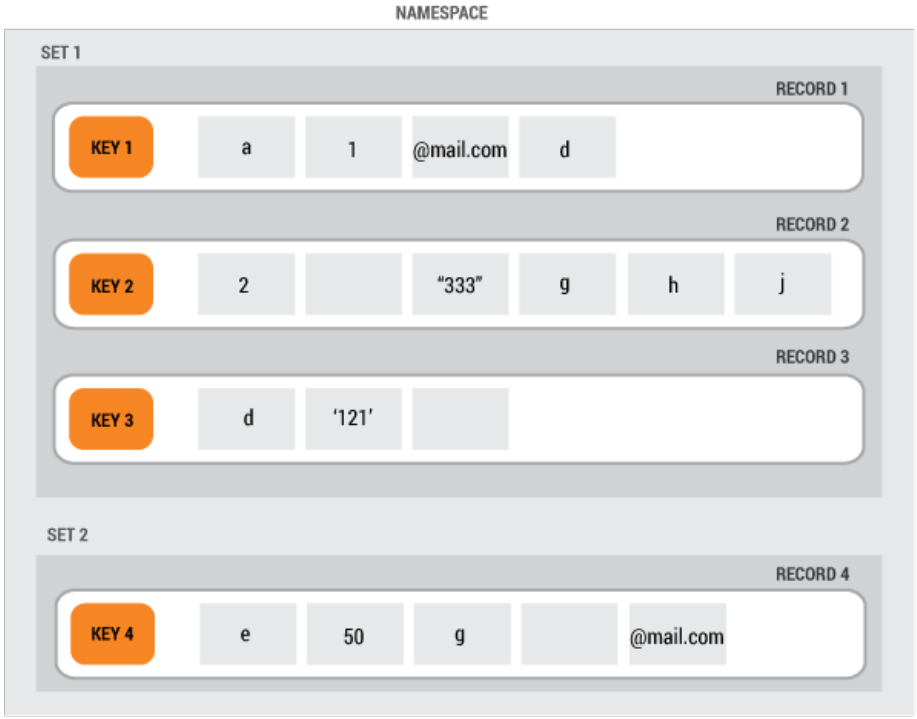
- **Storage** definition - DRAM or Flash
  - Hybrid
    - RAM for index and flash for data
  - RAM + disk for persistence only
    - RAM for index + data
  - RAM only
    - RAM for index + data
- **Policy** container
  - Replication factor
    - change requires cluster-wide restart
  - RAM and disk configuration
  - High-watermark
  - Default TTL - The client can override
    - TTL = 0 records never expire.
- **Difficult** to **add** or **remove**
  - Cluster wide restart - downtime



Set

Sets are **similar** to tables

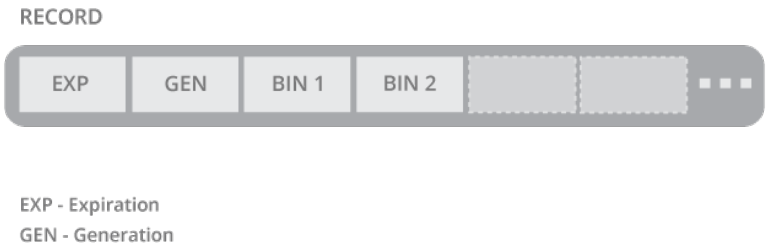
- But it has **no schema**
- Arbitrary grouping of records
- Inherits policy from namespace
- Prefix to **primary key**
- Set name <= 63 characters
- 1023 per namespace
- Cannot be deleted or renamed



Record

A record is a “**row**” of Key-value – (Object)

- **Value**: one or more bins
- Bin has a **name and type**
  - Bin types: String, integer, double, BLOB, list, map, GeoJSON
- Bins can be added at any time
- **Generation** counter
  - Optimistic Concurrency
  - After 64K - reset to Zero
- **Time-to-live** = auto expiration
- Reads/Writes are **atomic**
- Any change = **complete rewrite**
- Data stored **contiguously** on the same node.



Bins

- Bins have a:
  - Name – 14 characters or less
  - Type – one of the following
- A Bin can have a different type in another record
- One or more bins updated in a **single operation**
  - Increment (add)
  - Operate

Types

- String
- Integer
- Double
- Blob
- List
- Map
- GeoJSON

Id	lname	fname	address	favorites
1	Able	John	123 First	cats, dogs, mice
2	Baker	916570	234 Second	
3	Charlie			
4	Delta	Moe	456 Fourth	stake, ice cream, apples

There is a limit of 32K bin names in a namespace.

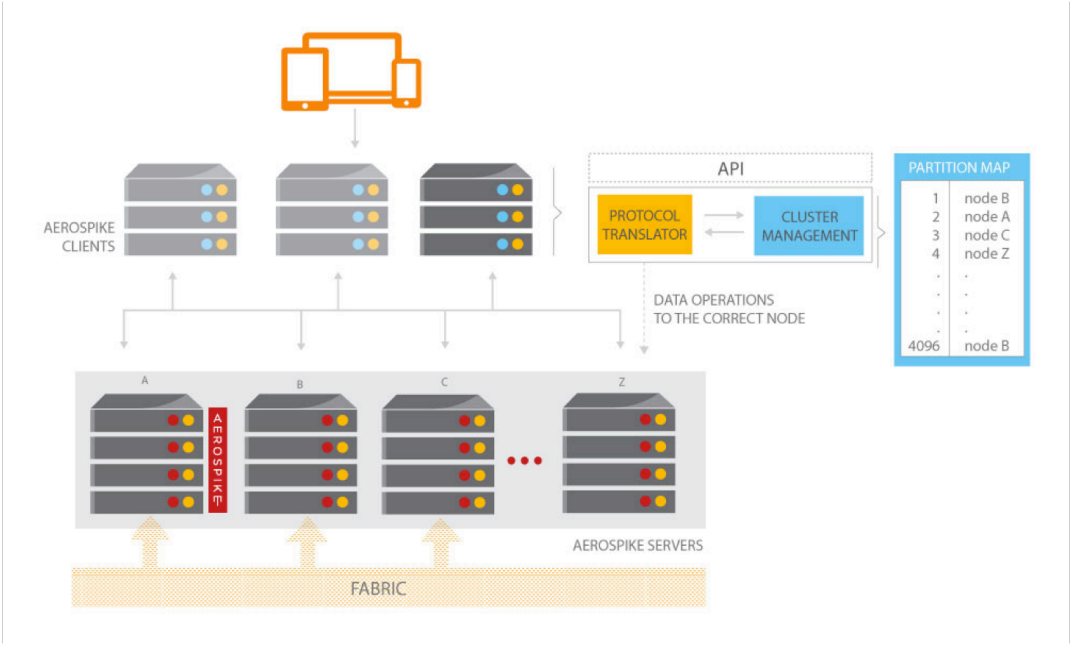
Bin names cannot be deleted or renamed.

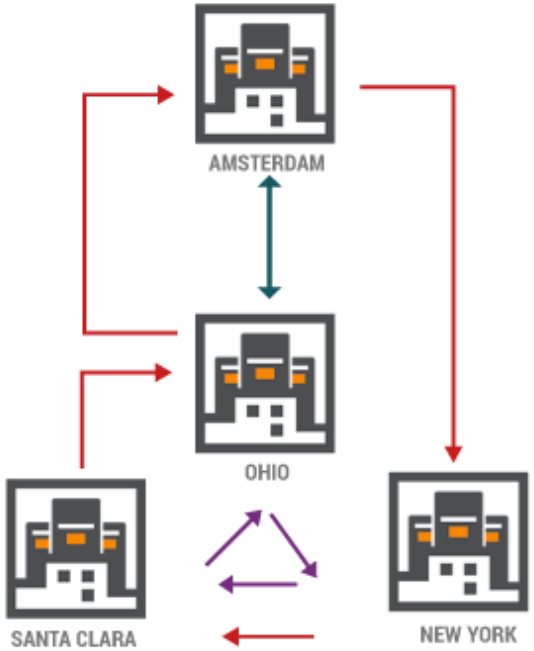
Database Hierarchy

Term	RDBMs	Definition	Notes
Cluster	-	An Aerospike cluster services a single database service.	While a company may deploy multiple clusters, applications will only connect to a single cluster.
Node	-	A single instance of an Aerospike database. A node will act as a part of the whole cluster.	For production deployments, a host should only have a single node. For development, you may place more than one node on a host.
Namespace	Database	An area of storage related to the media. Can be either RAM or flash (SSD based). Setting up new/removing namespaces requires a cluster-wide restart.	
Set	Table	An unstructured grouping of data that has some commonality.	Similar to “tables” in a relational database, but do not require a schema.
Record	Row	A key and all data related to that key.	Aerospike always stores all data for a record on the same node.
Bin	Column	One part of data related to a key.	Bins in Aerospike are typed, but the same bin in different records can have different types. Bins are not required. Single bin optimizations are allowed.

Architecture

Architecture – The Big Picture





1) **No Hotspots** – Distributed Hash Table simplifies data partitioning

2) Smart Client – **1 hop** to data, no load balancers

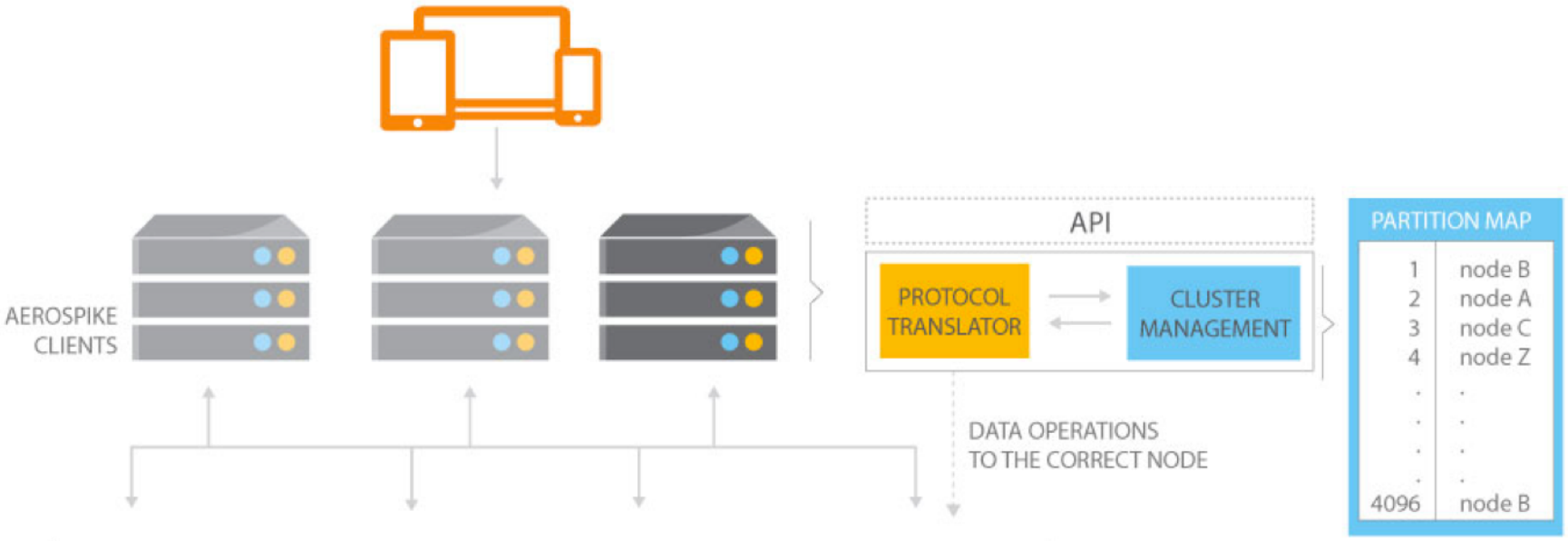
3) **Shared Nothing Architecture**, every node is identical

4) Smart Cluster, **Zero Touch** – auto-failover, rebalancing, rolling upgrades

5) Operations and long-running tasks prioritized in real-time

6) **XDR** – sync replication across data centers ensures **Zero Downtime**

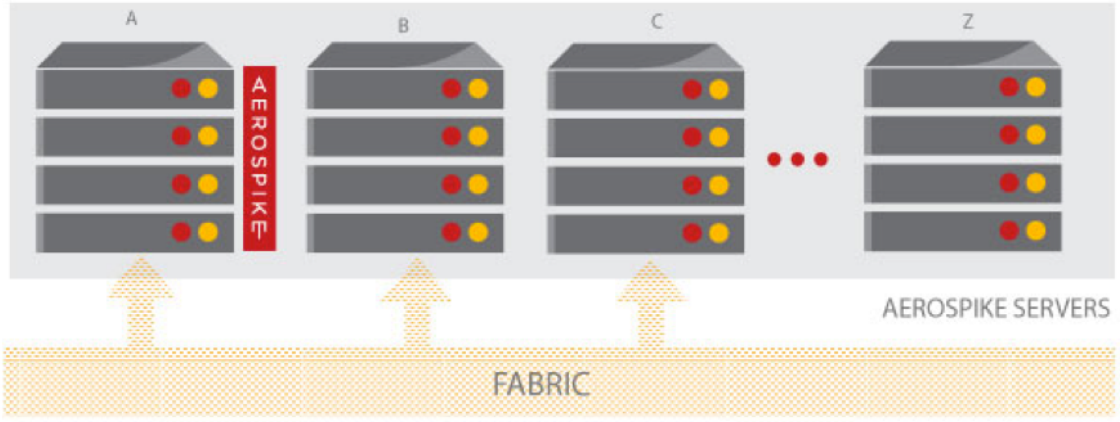
Smart Client™



- The Aerospike Client is implemented as a **library**, JAR or DLL, and consists of 2 parts:
  - Operation APIs – These are the operations that you can execute on the cluster – **CRUD+** etc.
  - First class **observer** of the Cluster – Monitoring the state of each node and aware on new nodes or node failures.

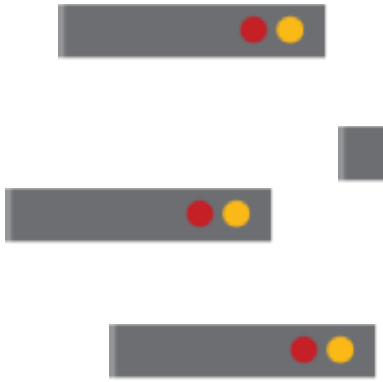
The Cluster (servers)

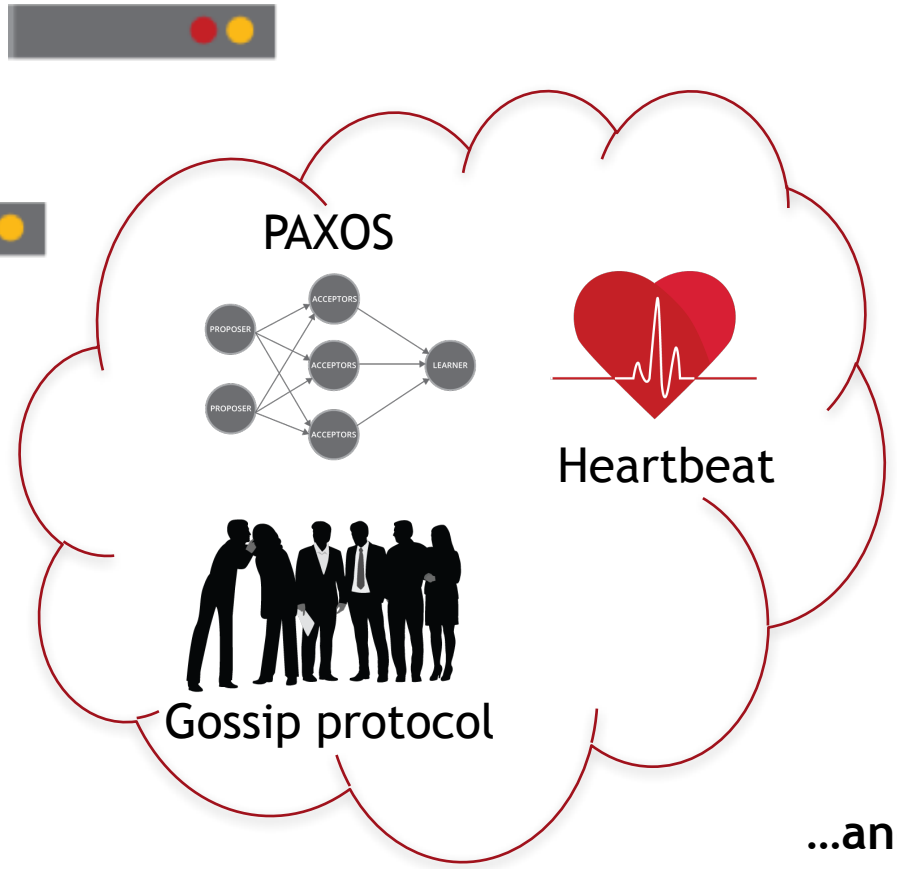
- Federation of **local servers**
  - XDR to remote cluster
- Automatic **load balancing**
- Automatic **fail over**
- Detects **new nodes** (multicast or mesh)
- **Rebalances** data (measured rate)
- Adds nodes **under load**
- **Locally attached** storage



Cluster formation

Individual nodes go in…





…and the Cluster forms

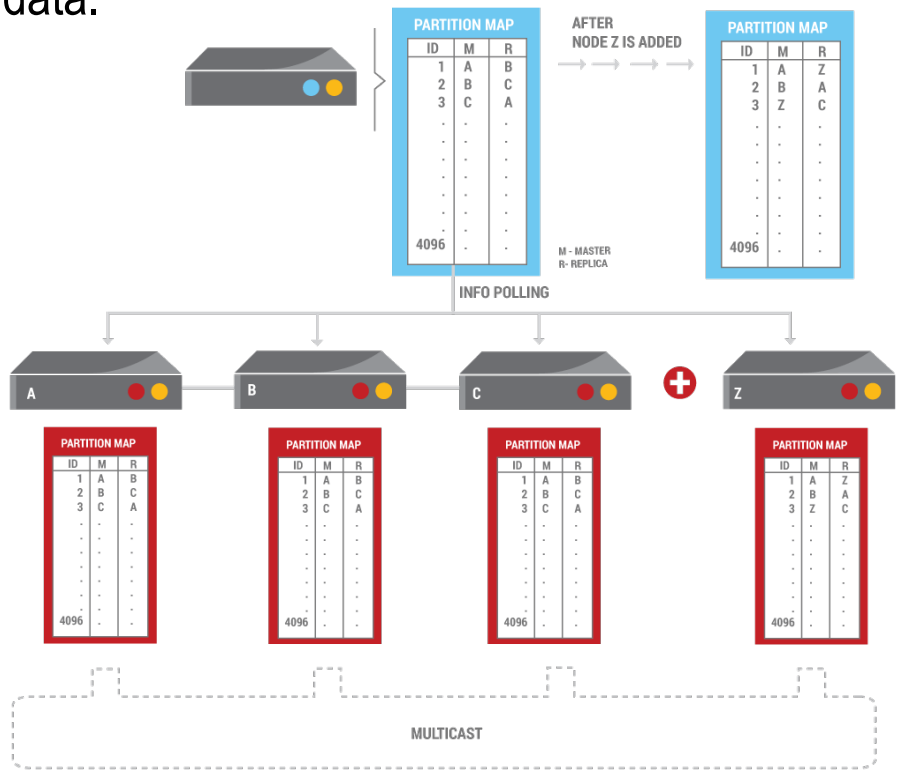
Distributing Data: The Partition Map

Distributing data can be done in many ways. Aerospike has chosen a method that:

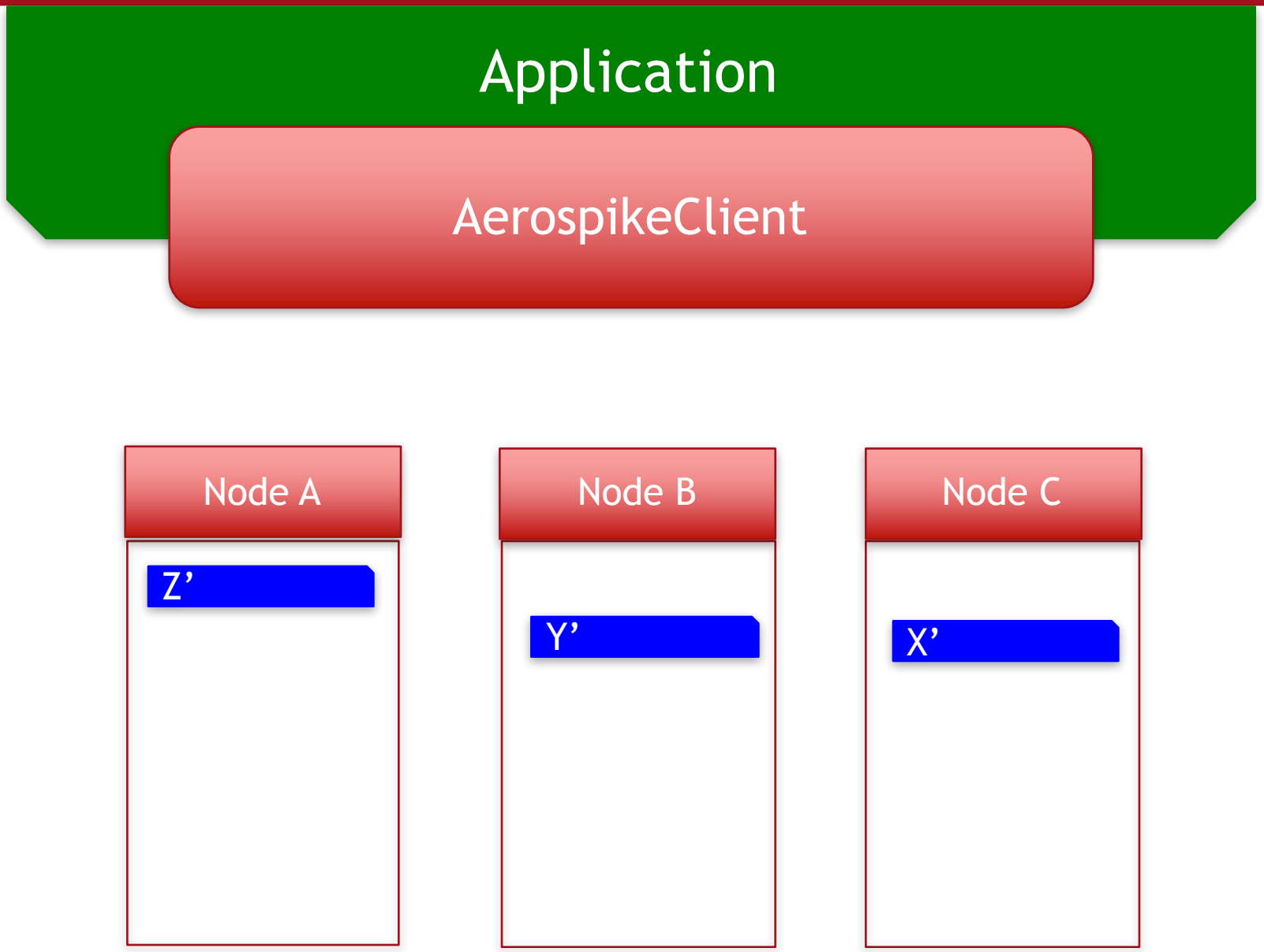
1. Automatically **balances** data across **nodes**.
2. Makes it easy to **migrate (rebalance)** should a node crash or be added.
3. Does not require the developer to understand how the data is **distributed**.
4. Takes into account **replica** copies of the data.

1 Hop to data

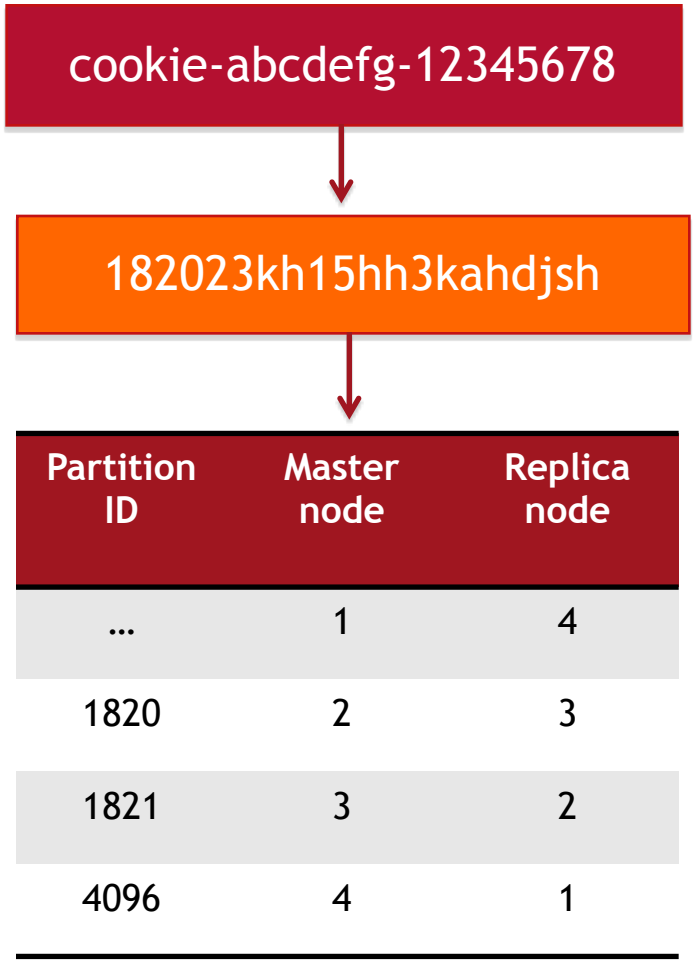
- Smart Client simply calculates Partition ID to determine Node ID
- No Load Balancers required



Even record distribution



Data is Distributed Randomly



Partition ID	Master node	Replica node
…	1	4
1820	2	3
1821	3	2
4096	4	1

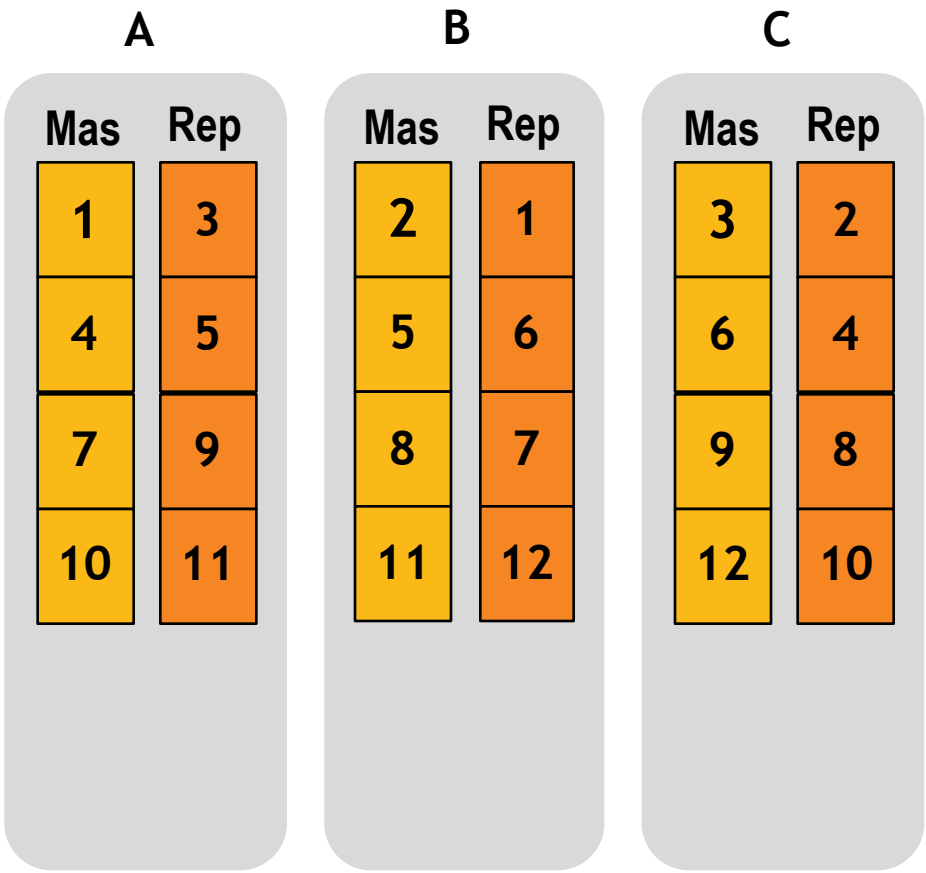
- Every key is hashed into a 20 byte (fixed length) string using the RIPEMD160 hash function
- This hash + additional data (fixed 64 bytes) are stored in RAM in the index
- 12 bits of this hash are used to compute the partition id
- There are 4096 partitions
- Partition id maps to node id based on cluster membership

Losing a Node

Let's take a 3 node cluster with 12 partitions and a replication factor of 2.
When everything is stable, every thing will be evenly distributed.

Partition Map

Partition	Master	Replica
1	A	B
2	B	C
3	C	A
4	A	C
5	B	A
6	C	B
7	A	B
8	B	C
9	C	A
10	A	C
11	B	A
12	C	B

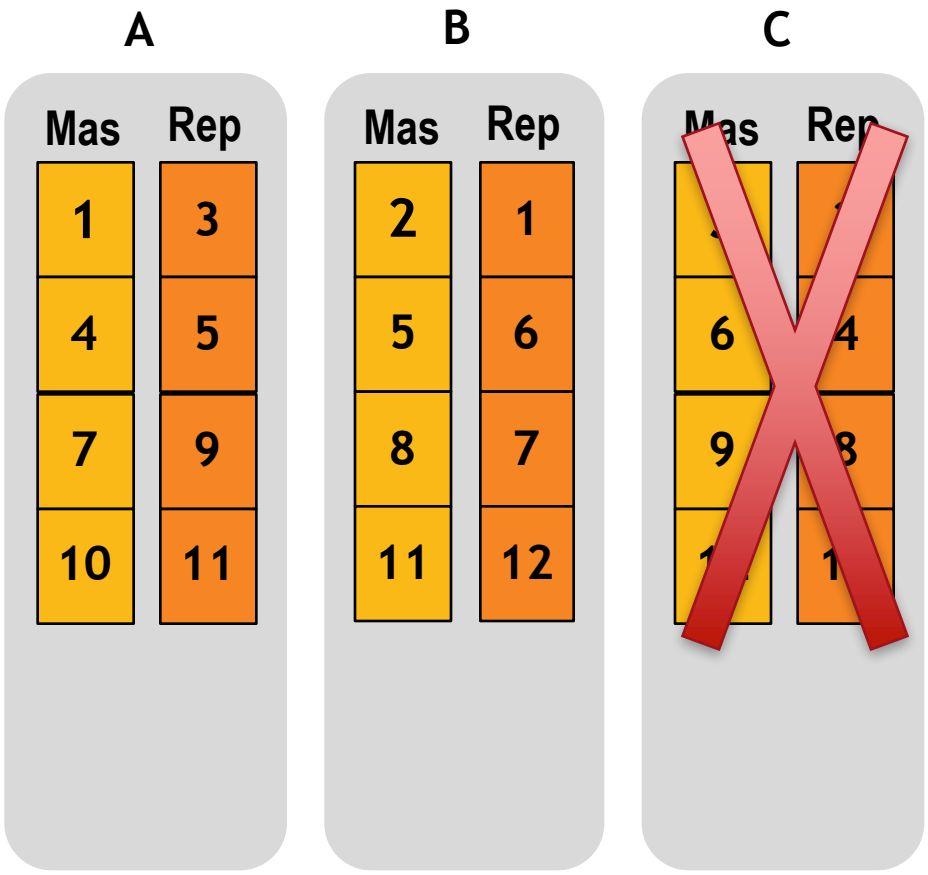


Losing a Node

So what happens if a node dies?

Partition Map

Partition	Master	Replica
1	A	B
2	B	C
3	C	A
4	A	C
5	B	A
6	C	B
7	A	B
8	B	C
9	C	A
10	A	C
11	B	A
12	C	B

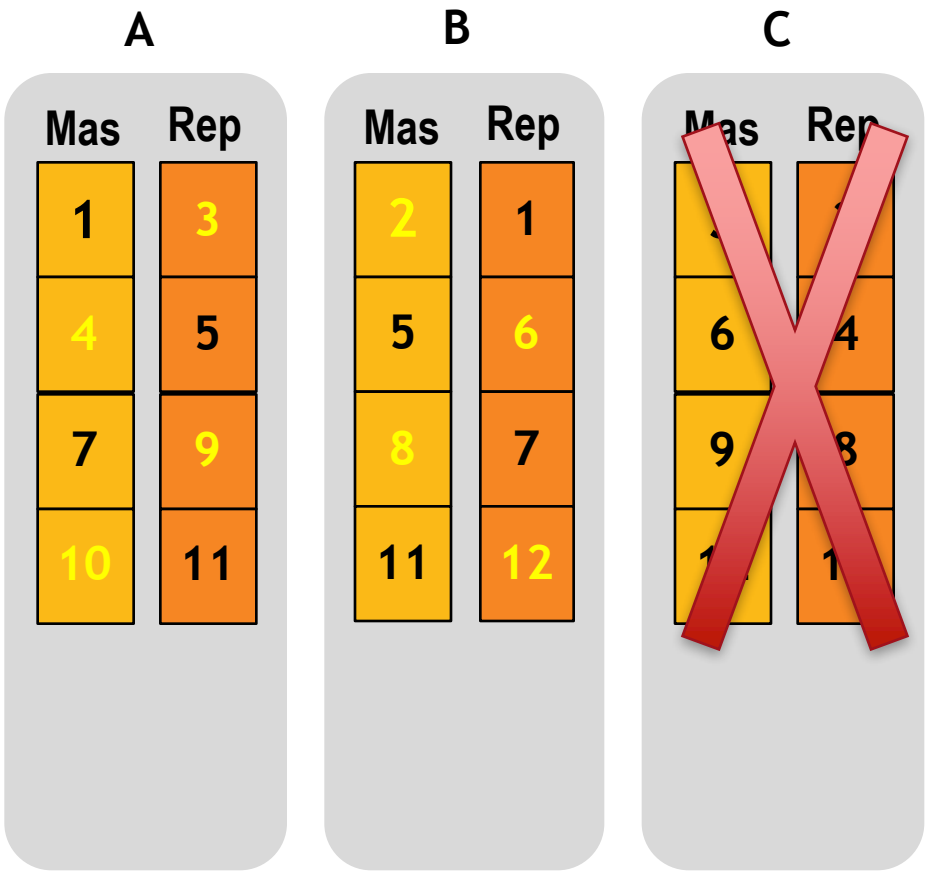


Losing a Node

Some of the partitions will only have a single copy.

Partition Map

Partition	Master	Replica
1	A	B
2	B	C
3	C	A
4	A	C
5	B	A
6	C	B
7	A	B
8	B	C
9	C	A
10	A	C
11	B	A
12	C	B

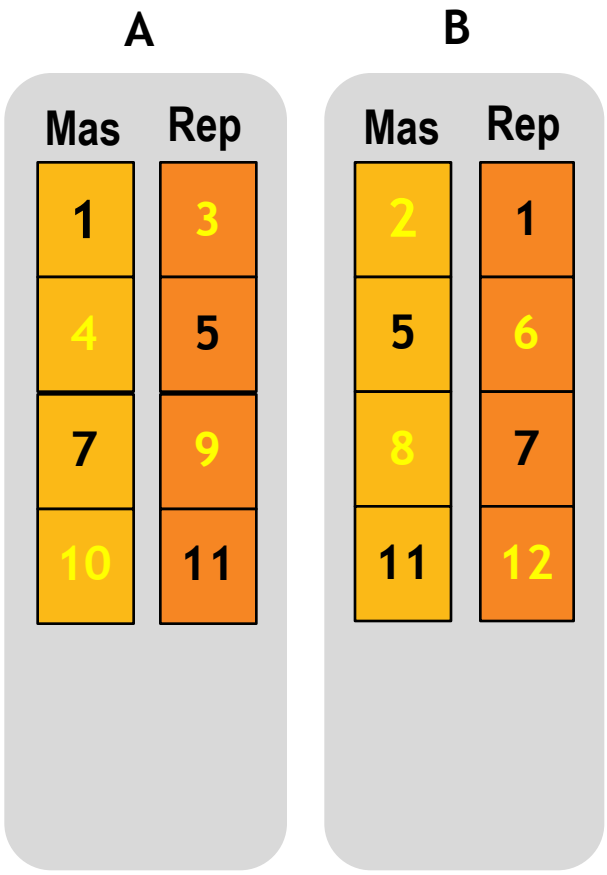


Losing a Node

So the cluster will exclude the missing node and create a new partition map.

New Partition Map

Partition	Master	Replica
1	A	B
2	B	A
3	B	A
4	A	B
5	B	A
6	A	B
7	A	B
8	B	A
9	B	A
10	A	B
11	B	A
12	A	B

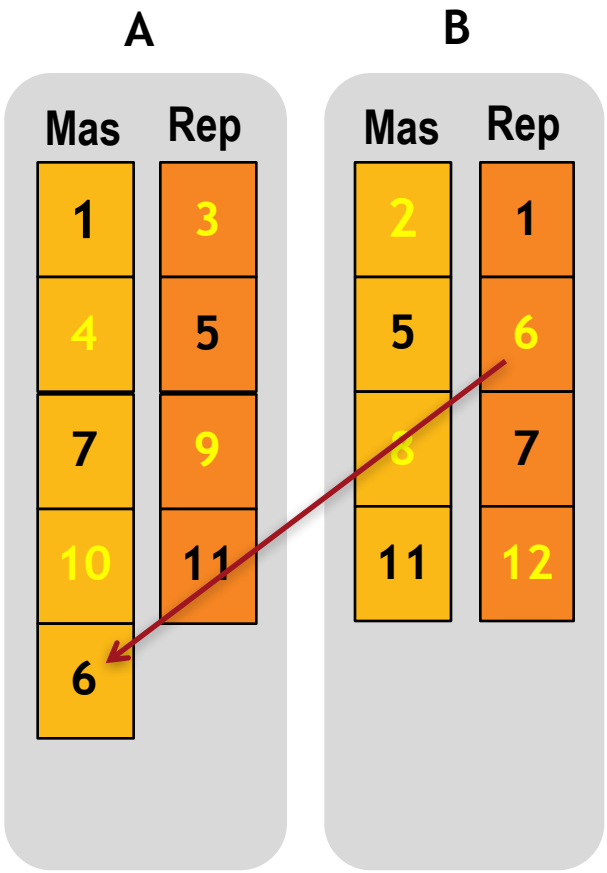


Losing a Node

It will then begin to make copies of all the data

New Partition Map

Partition	Master	Replica
1	A	B
2	B	A
3	B	A
4	A	B
5	B	A
6	A	B
7	A	B
8	B	A
9	B	A
10	A	B
11	B	A
12	A	B



Losing a Node

Once it has completed all the partitions, the cluster will be in a stable state again. With 2 full copies of all data.

New Partition Map

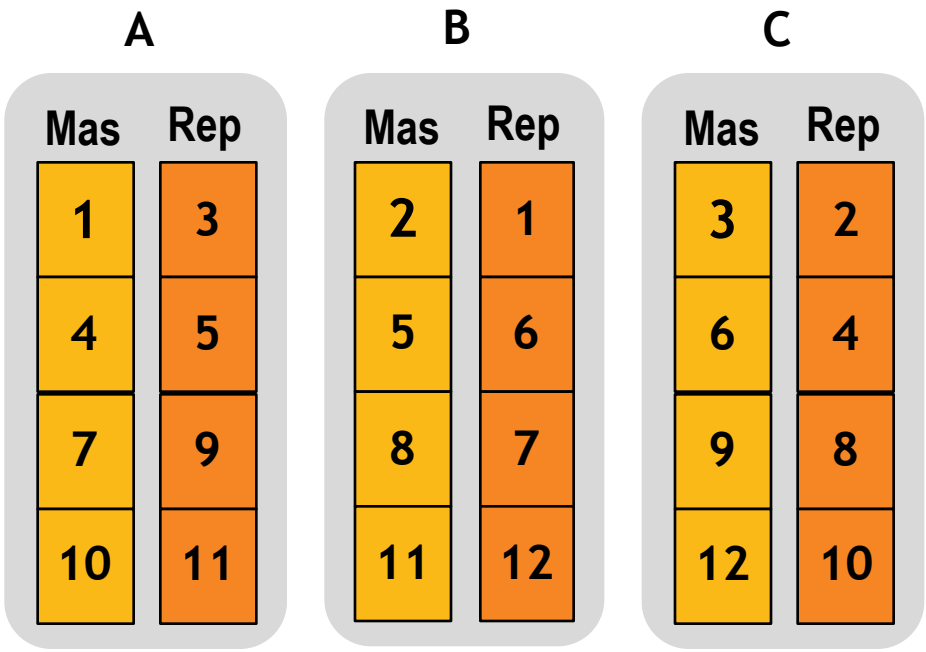
Partition	Master	Replica
1	A	B
2	B	A
3	B	A
4	A	B
5	B	A
6	A	B
7	A	B
8	B	A
9	B	A
10	A	B
11	B	A
12	A	B

Adding a Node

Now let's start with the same situation, but add a node this time. The same starting state: 12 partitions, 3 nodes, replication factor of 2.

Partition Map

Partition	Master	Replica
1	A	B
2	B	C
3	C	A
4	A	C
5	B	A
6	C	B
7	A	B
8	B	C
9	C	A
10	A	C
11	B	A
12	C	B



Adding a Node

When the new node is added, it starts empty.

Partition Map

Partition	Master	Replica
1	A	B
2	B	C
3	C	A
4	A	C
5	B	A
6	C	B
7	A	B
8	B	C
9	C	A
10	A	C
11	B	A
12	C	B

Adding a Node

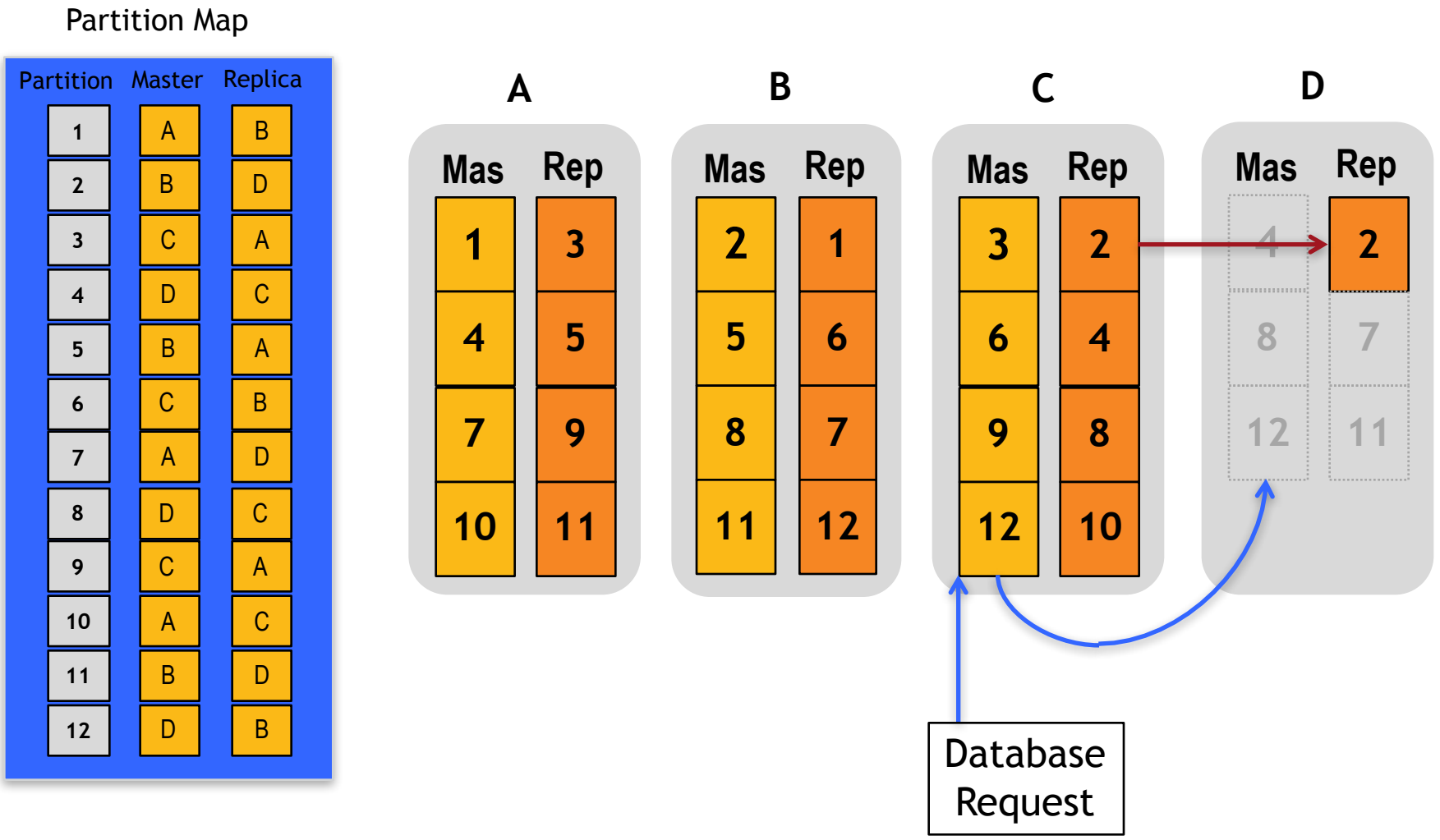
The cluster creates a new partition map, with the new node included.

Partition Map

Partition	Master	Replica
1	A	B
2	B	D
3	C	A
4	D	C
5	B	A
6	C	B
7	A	D
8	D	C
9	C	A
10	A	C
11	B	D
12	D	B

Adding a Node

The cluster will then migrate (rebalance) the partitions, one at a time to the new node. During this time it is possible for the partition map to be out of sync with the actual data distribution. Aerospike nodes will proxy the request.

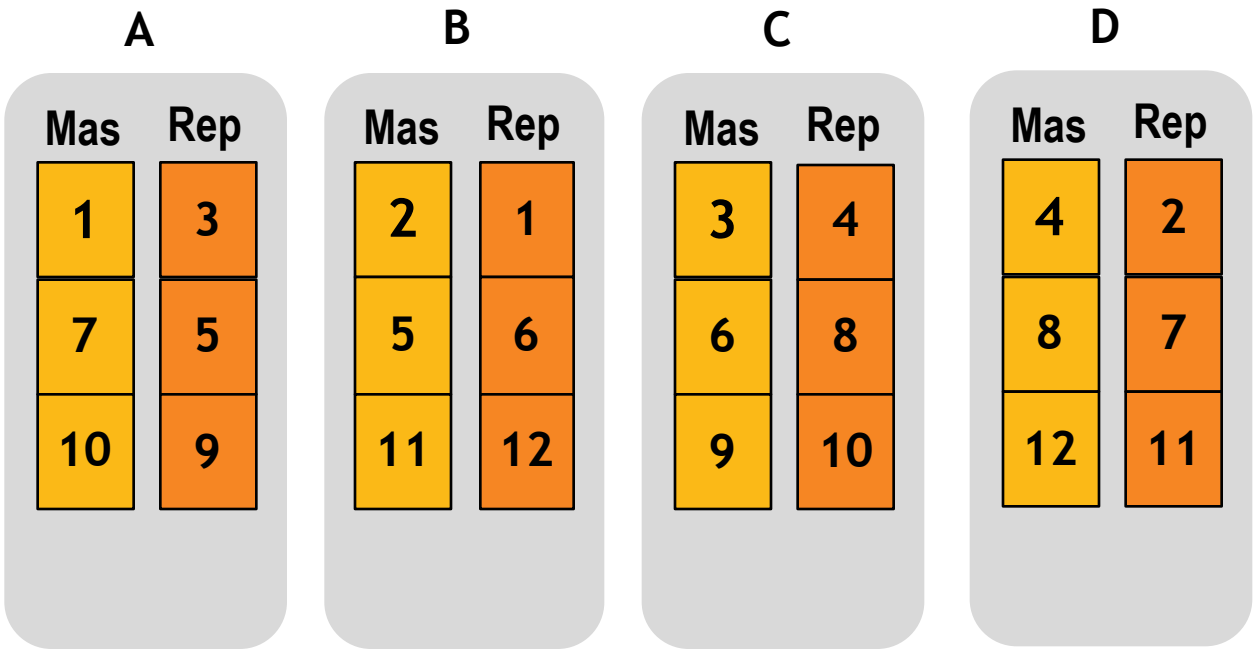


Adding a Node

Once all the partitions have migrated, the database will be in a new stable state, with replicated copies of all data again.

Partition Map

Partition	Master	Replica
1	A	B
2	B	D
3	C	A
4	D	C
5	B	A
6	C	B
7	A	D
8	D	C
9	C	A
10	A	C
11	B	D
12	D	B



Summary

In this module, we covered:

- Data hierarchy.
- High level architecture.
- Data partitioning.