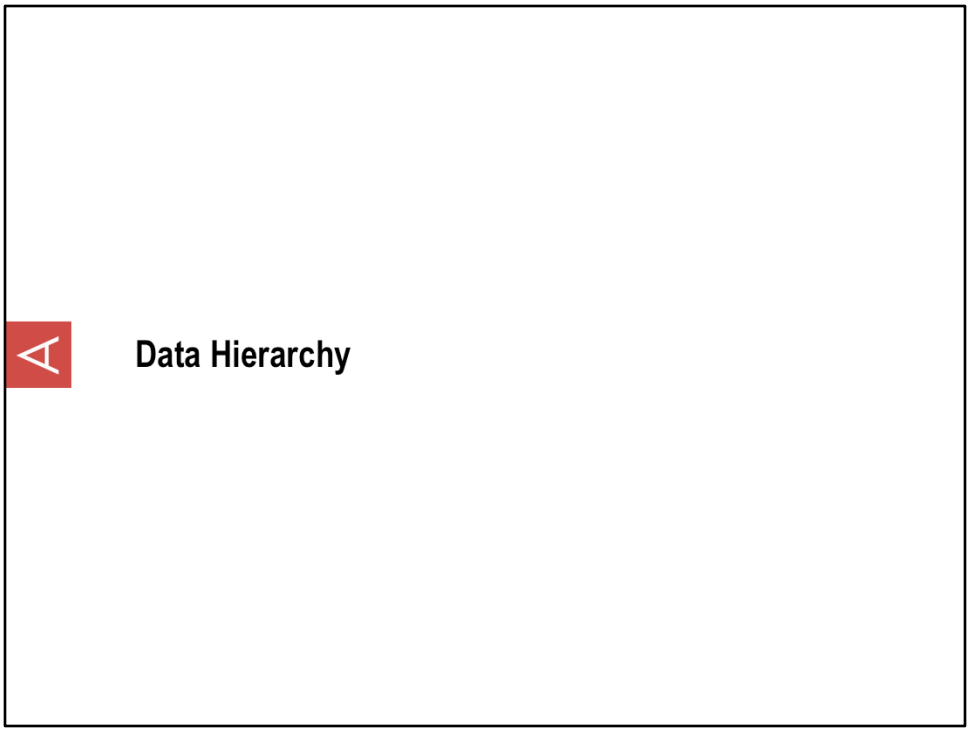
Architecture

Objectives

This module covers the following:

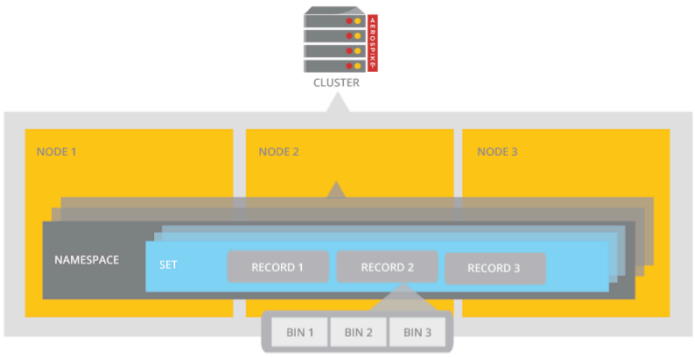
- Data hierarchy.
- High level architecture.
- Data partitioning.


Data Hierarchy

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.

Data Hierarchy

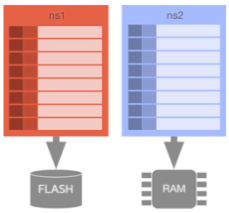


Nodes

- 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 imply 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.

Tracking what happens in the event of node loss is critical. You should understand what will happen under different failure modes, such a single node loss, single rack loss, etc.

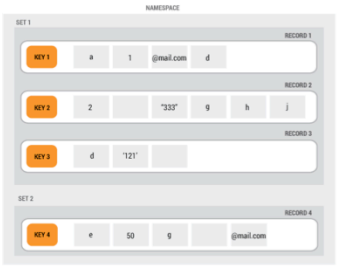
Namespaces



- Similar to a database in a relational database.
- Are associated with the storage media:
  - Hybrid (RAM for index and flash for data)
  - RAM + disk for persistence only (RAM for index + data)
  - RAM only (RAM for index + data)
- Each can be configured with their own:
  - replication factor (change requires a cluster-wide restart which means downtime)
  - RAM and disk configuration
  - settings for high-watermark
  - default TTL (if you have data that must never be automatically deleted, you must set this to "0"). The client can override the default TTL.
  - Adding/Removing of namespaces require a cluster-wide restart (downtime), but configuration and storage can be changed through a rolling restart of the nodes (no cluster downtime).
- Some companies will choose to add an empty namespace on first deployment to ensure they can add a namespace without a cluster-wide restart.

Sets

- Similar to “tables” in relational databases.
- Sets are optional.
- Schema does not have to be pre-defined. Records in the same set may have different bins, or even different types to the same bin.
- In order to request a record, you must know its set.
- Scans can be done across a set
- Set names take up space per record, so they should be kept small



This term is the most often confused. While the logical association with relational tables is roughly correct, the differences are easy to trip up people familiar with relational tables.

Records

- Similar to a row in a relational database.
- Any change to a record will result in a complete write of the entire record.
- Sometimes referred to as “objects”.
- Writes to records are atomic.
- All data for a record will be stored on the same node.
- Data will be in the same block, except for Large Data Types (LDTs), so will be limited by the write-block-size (128KB default, 1 MB max).
- Although LDTs store data for a record in a different block, it will still be on the same node.

Bins

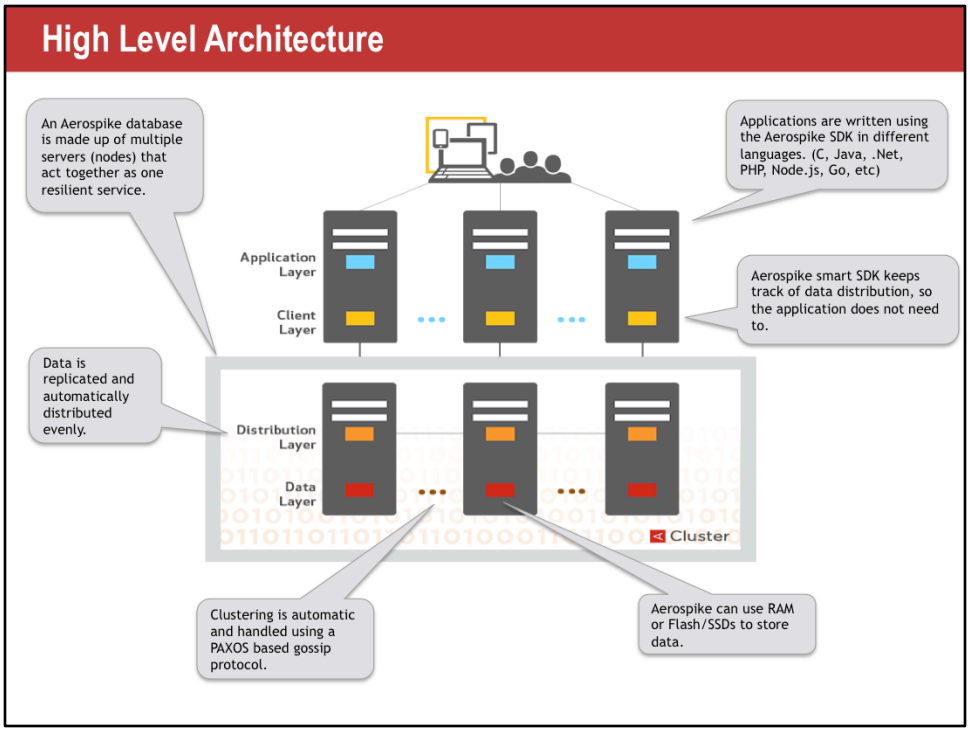
- Bin values Are typed. Current types are:
  - integer
  - string
  - blob [language specific]
  - list
  - map

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

- Different records may have the same bin name with different types.
- A single bin may be updated by the client, without requiring multiple transactions (read + update)
  - Increment (add)
  - User Defined Function (UDF)
- There is a limit of 32K bin names in a namespace, even if they were deleted.

The 32K limit is due to Aerospike storing the bin names but not deleting them. This is of particular interest to admins familiar with columnar databases such as Cassandra, where in many cases the database may have many thousands of name, or even column names with timestamps.


High Level Architecture

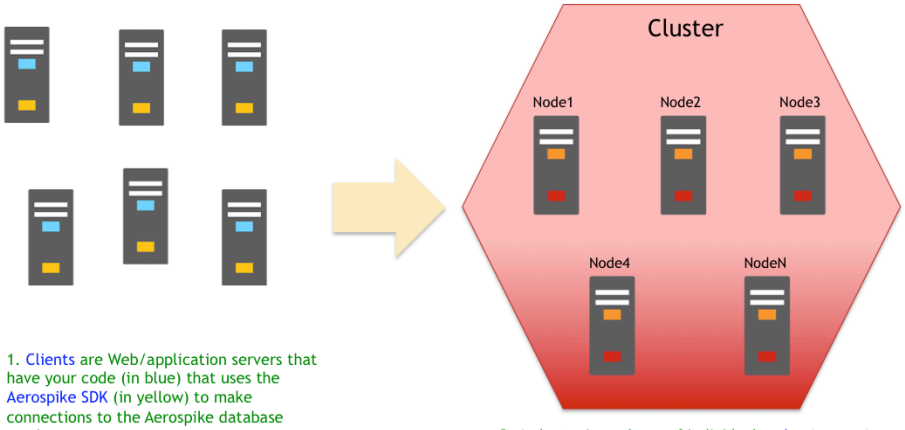


Aerospike was designed from the beginning as a distributed database.

The initial intent was to make a high throughput, low latency database that was capable of staying up 24 x 7 with no need to go down due to hardware failure or maintenance.

The client SDKs have been written in a number of different languages. Please go to http://www.aerospike.com/develop/ to see a full list of supported languages.

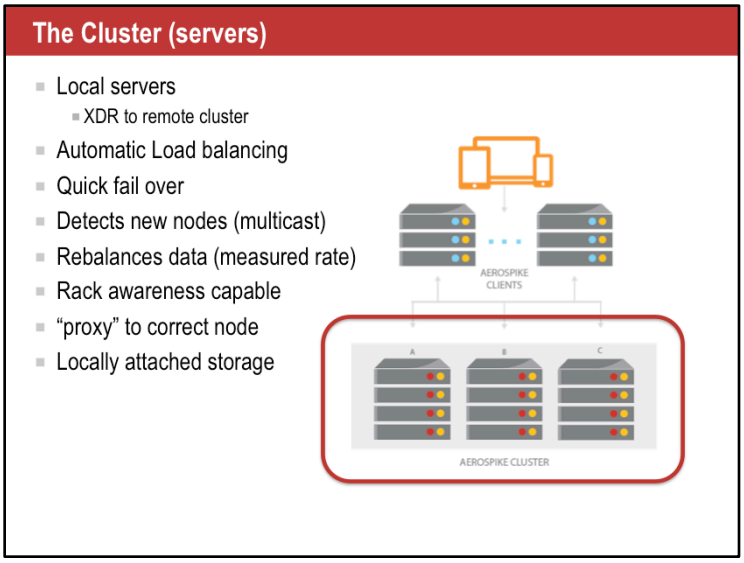
High Level Terminology



2. The Aerospike database service is provided by a cluster. Your code does not need to be aware of the internal structure.

1. Clients are Web/application servers that have your code (in blue) that uses the Aerospike SDK (in yellow) to make connections to the Aerospike database service.

3. A cluster is made up of individual nodes (servers) that store data in a distributed manner. These nodes can store multiple copies of the data, which is the replication factor. For example, Primary + 1 Secondary copy means a replication factor of 2.



Cluster Layer

The Aerospike “shared nothing” architecture is designed to reliably store TB of data with automatic fail-over, replication and cross data-center synchronization. This layer scales linearly and implements many of the ACID guarantees. This layer is also designed to eliminate manual operations with the systematic automation of all cluster management functions. It includes 3 modules:

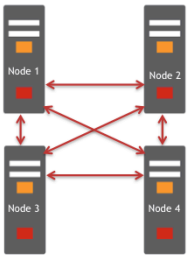
- The Cluster Management Module tracks nodes in the cluster. The key algorithm is a Paxos-like consensus voting process which determines which nodes are considered part of the cluster. Aerospike implement special heartbeat (active and passive) to monitor inter-node connectivity.
- When a node is added or removed and cluster membership is ascertained, each node uses distributed hash algorithm to divide the primary index space into data 'slices' and assign owners. Data Migration Module then intelligently balances the distribution of data across nodes in the cluster, and ensures that each piece of data is duplicated across nodes and across data centers, as specified by the system’s configured replication factor.
- Division is purely algorithmic, the system scales without a master and eliminates the need for additional configuration that is required in a sharded environment.
- The Transaction Processing Module reads and writes data as requested and provides many of the consistency and isolation guarantees. This module is responsible for
    1. Sync/Async Replication : For writes with immediate consistency, it propagates changes to all replicas before committing the data and returning the result to the client.
    2. Proxy : In rare cases during cluster re-configurations when the Client Layer may be briefly out of date, it transparently proxies the request to another node.
    3. Duplicate Resolution : when a cluster is recovering from being partitioned, it resolves any conflicts that may have occurred between different copies of data.

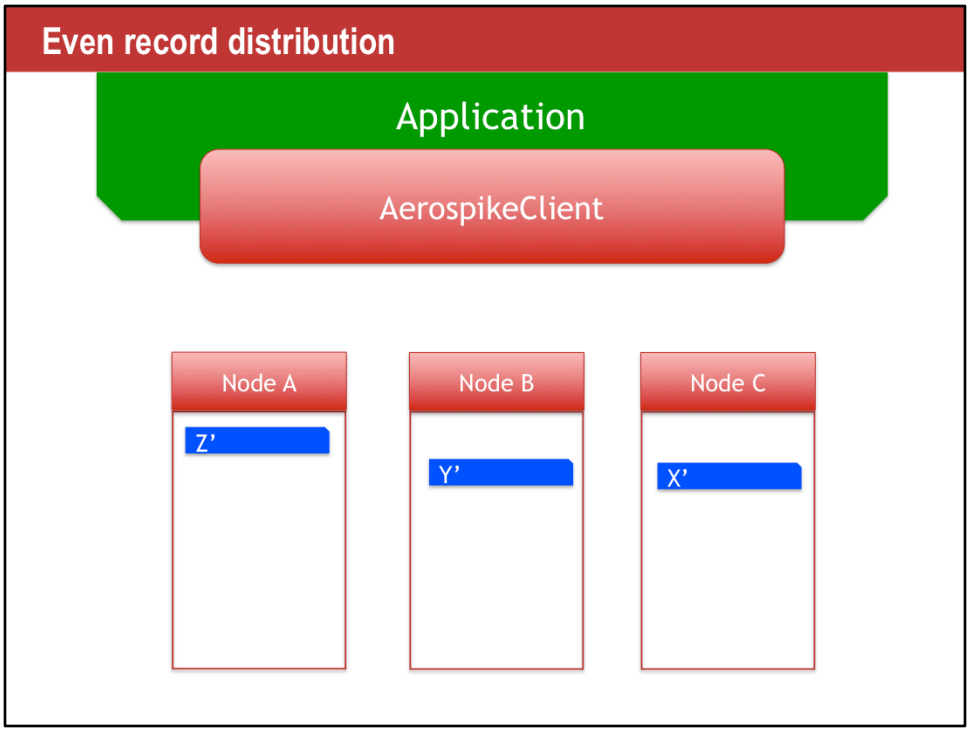
Once you have the first cluster up, you can optionally install additional clusters in other data centers and setup cross data-center replication – this ensures that if your data center goes down, the remote cluster can take over the workload with minimal or no interruption to users.

Cluster Formation

The basic way this operates is that each node must send heartbeats that can be heard by other nodes. When enough of the heartbeats from one server have been missed by the others, it will be removed from the cluster.

We will look into this in more detail later.

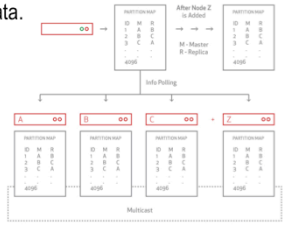


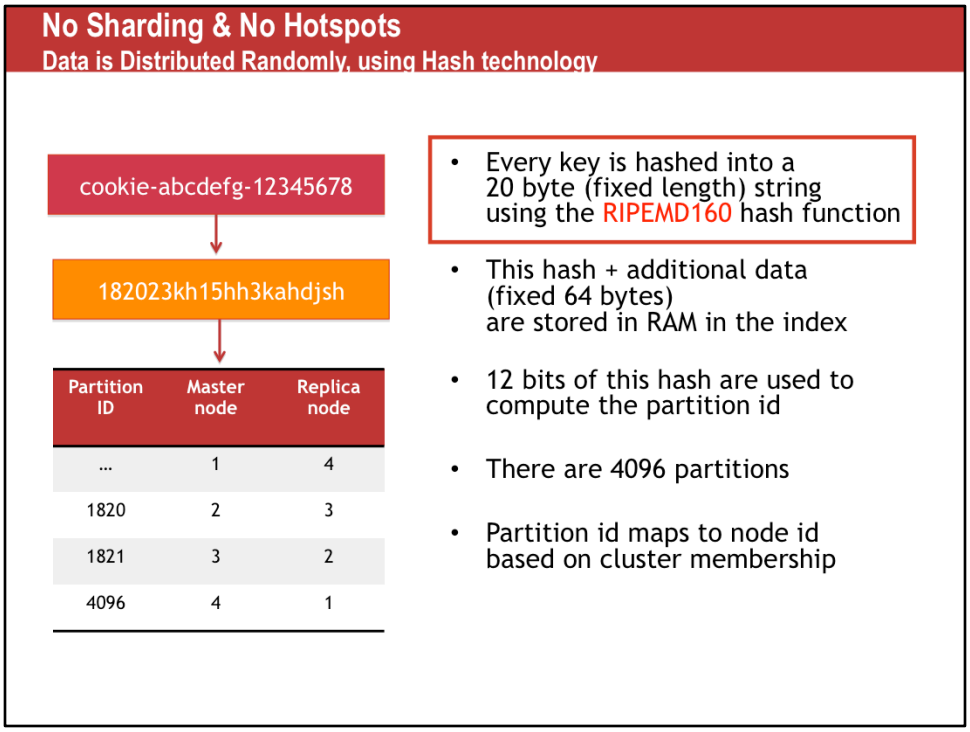

Even record distribution
Application
AerospikeClient
Node A
Z'
Node B
Y'
Node C
X'

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. Allows for a single network connection from any client to the cluster.
4. Does not require the developer to understand how the data is distributed.
5. Takes into account replica copies of the data.





When a cluster forms, it will generate a partition map.

The partition map is distributed to all the nodes in the cluster as well as all the clients. It is essential to the central operation of the database and allows for it to run without any true “master”.

For the curious: https://online.tugraz.at/tug_online/voe_main2.getvolltext?pCurrPk=17675. And: https://discuss.aerospike.com/t/what-will-aerospike-do-if-there-is-a-hash-collision-two-records-have-the-same-key/779.

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.

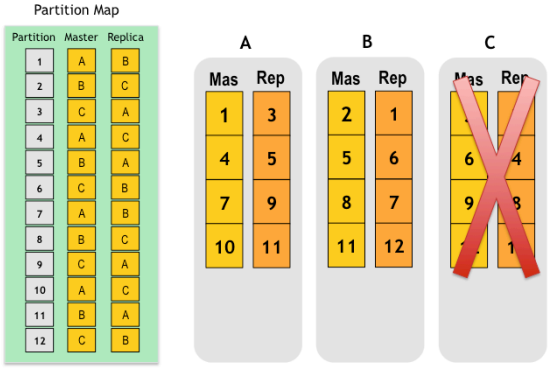
Partition Map

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

A		B		C	
Mas	Rep	Mas	Rep	Mas	Rep
1	3	2	1	3	2
4	5	5	6	6	4
7	9	8	7	9	8
10	11	11	12	12	10

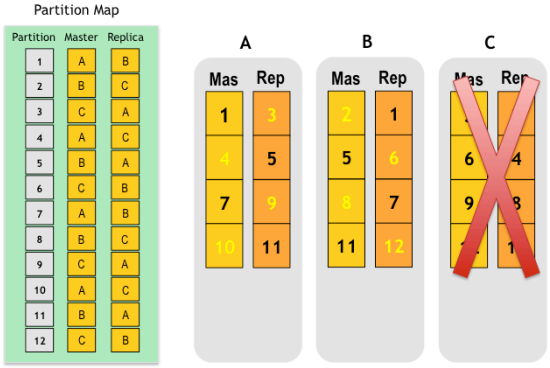
Losing a Node

So what happens if a node dies?



Losing a Node

Some of the partitions will only have a single copy.

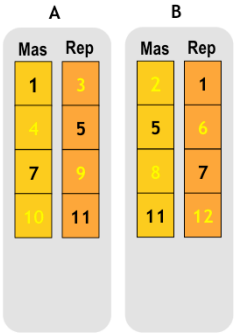


Losing a Node

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

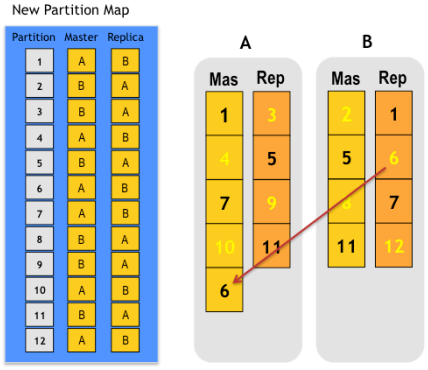
New Partition Map

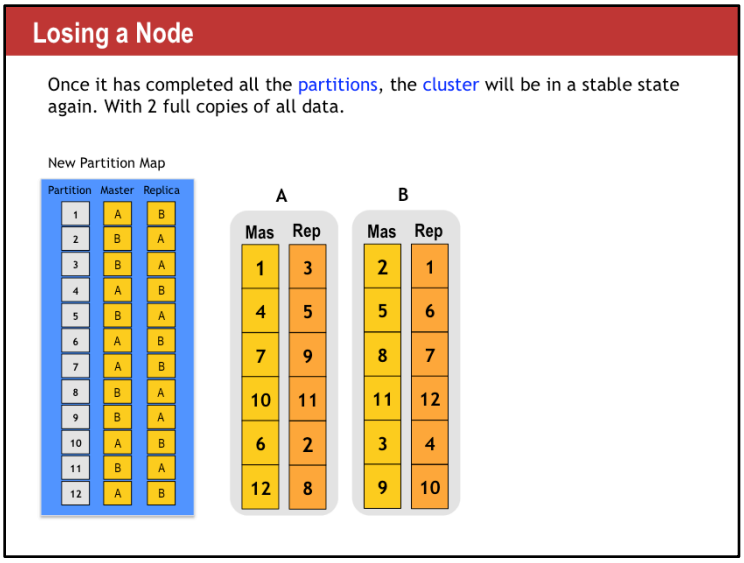
Partition	Master	Replica
1	A	B
2	B	A
3	B	A
4	A	B
5	B	A
6	A	B
7	A	B
8	B	A
9	B	A
10	A	B
11	B	A
12	A	B



Losing a Node

It will then begin to make copies of all the data, one partition at a time.





Migrations can be tuned. They will impact performance in some situations, especially for writes, due to default write duplicate resolution.
Aerospike is working on improving and trying to minimize overhead of migrations.
If necessary, migrations can be sped up or slowed down. See: https://discuss.aerospike.com/t/speeding-up-migrations/683 for details.

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

A

Mas	Rep
1	3
4	5
7	9
10	11

B

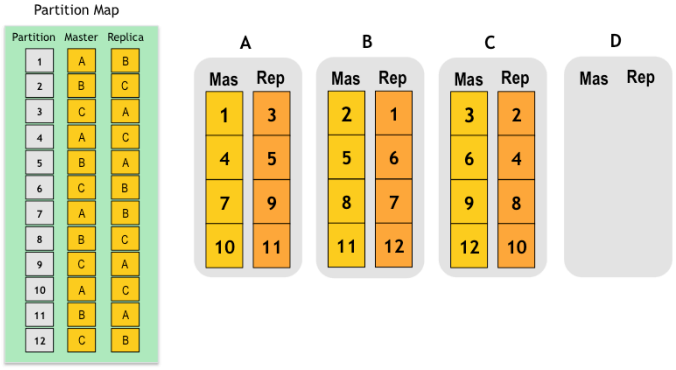
Mas	Rep
2	1
5	6
8	7
11	12

C

Mas	Rep
3	2
6	4
9	8
12	10

Adding a Node

When the new node is added, it starts empty.



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

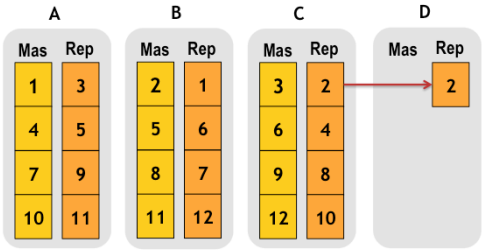
A		B		C		D	
Mas	Rep	Mas	Rep	Mas	Rep	Mas	Rep
1	3	2	1	3	2		
4	5	5	6	6	4		
7	9	8	7	9	8		
10	11	11	12	12	10		

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.

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

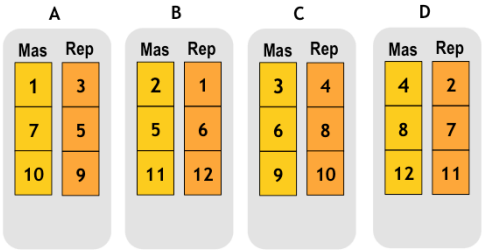


Adding a Node

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.

These tasks are ones that you can use in your environments. Doing proper benchmarking will not only help to determine what your performance is, but also useful for finding basic connectivity issues.