
AEROSPIKE
IN-MEMORY + NOSQL + ACID
AEROSPIKE DEPLOYMENT
CONFIGURATION
Aerospike aer . o . spike [air-oh- spahyk]
noun, 1. tip of a rocket that enhances speed and stability

Objectives

In this section we will be covering:

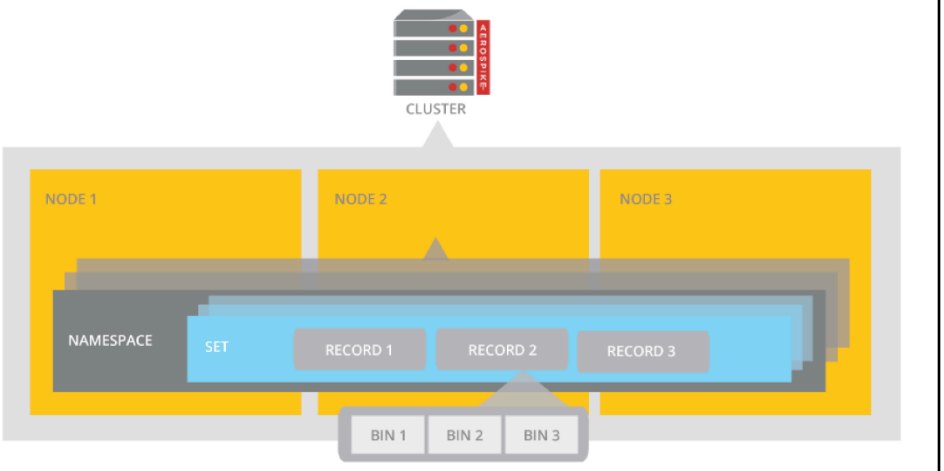
- Data Hierarchy
- How Aerospike Reads/Writes/Updates Data
- Defragmentation
- Data Hygiene
- Configuring Storage

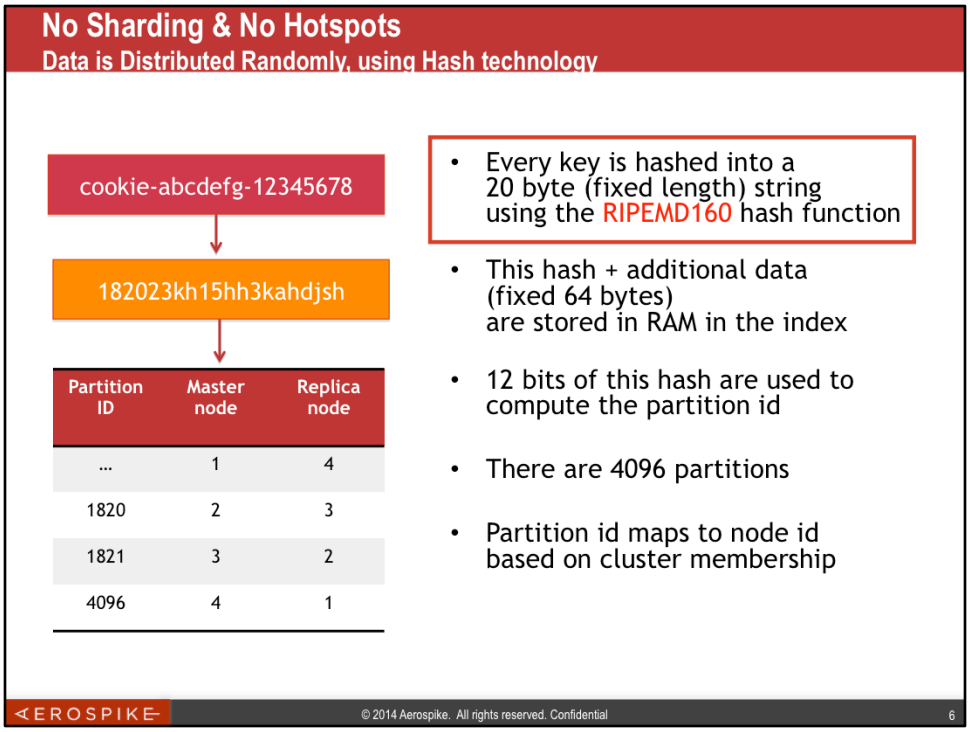

Data Hierarchy
AEROSPIKE
© 2014 Aerospike. All rights reserved. Confidential
3

Database Hierarchy

Term	RDBMs	Definition	Notes
Cluster	Database	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).	
Set	Table	An unstructured grouping of data that have 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.	
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



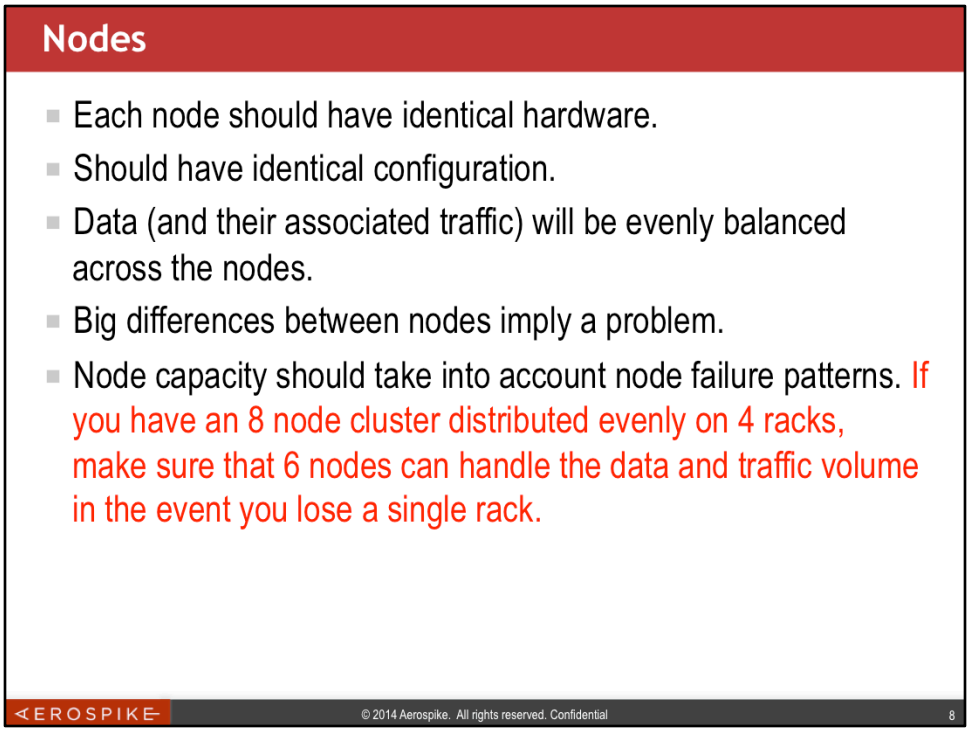


When a cluster forms, it will generate a partition map.

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”.

Cluster

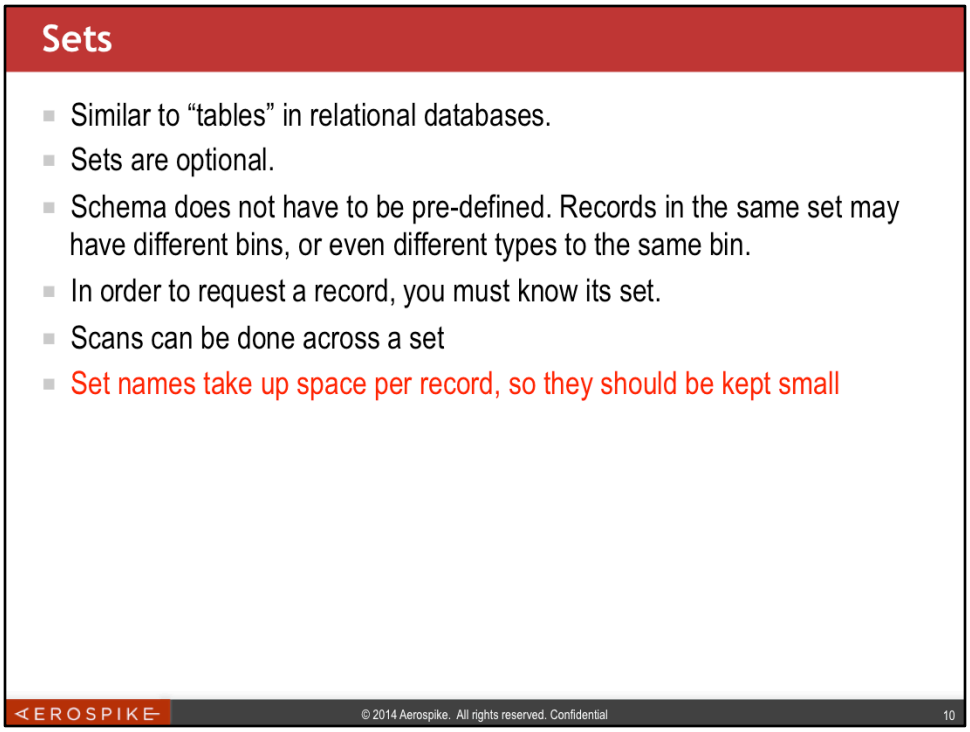
- Will be distributed on different nodes.
- Management of cluster is automated, so no manual rebalancing or reconfiguration is necessary.
- Will contain one or more namespaces. Adding/removing namespaces requires a cluster-wide restart.



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

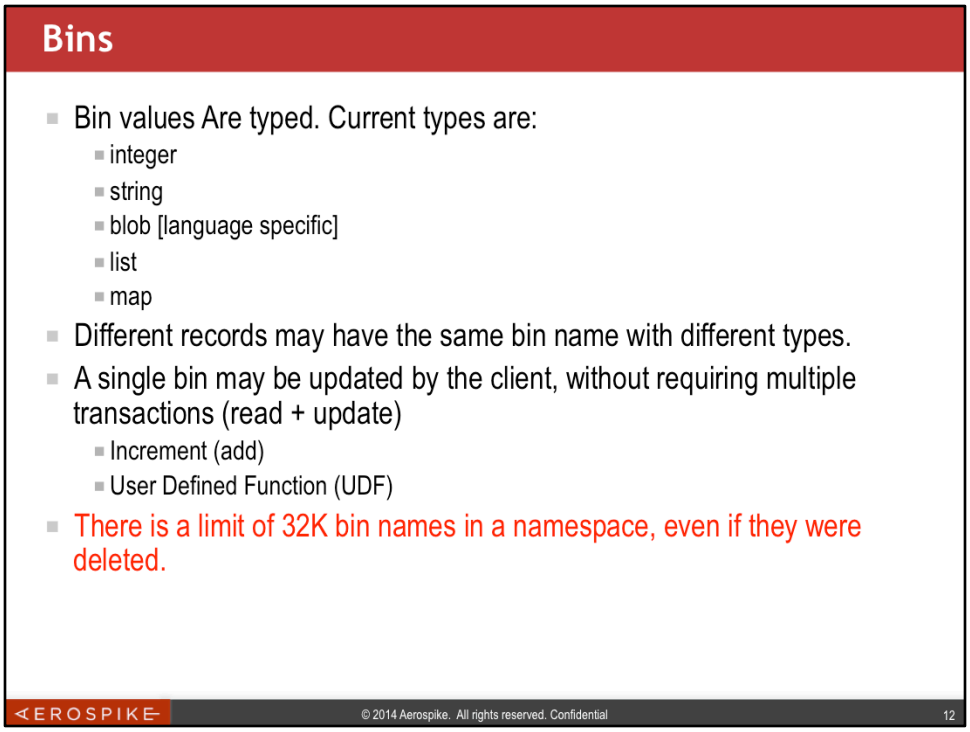
- Similar to a database in a relational database.
- Are associated with the storage media:
  - Hybrid (RAM for index and flash for data)
  - RAM + disk for persistence only (RAM for index + data)
  - RAM only (RAM for index + data)
- Each can be configured with their own:
  - replication factor (change 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.
  - Changes to namespaces require a cluster-wide restart (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.



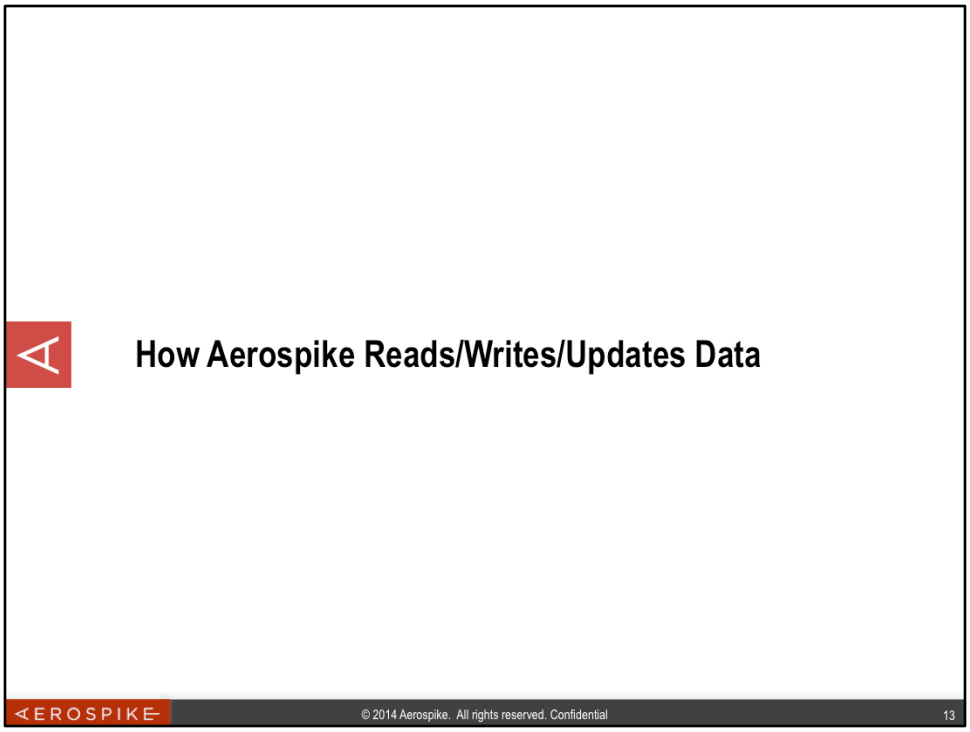
This term is the most often confused. While the logical association with relational tables is roughly correct, the differences are easy to trip up people familiar with relational tables.

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.



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.


How Aerospike Reads/Writes/Updates Data
AEROSPIKE
© 2014 Aerospike. All rights reserved. Confidential
13

Data Access Patterns

How does Aerospike handle the following operations

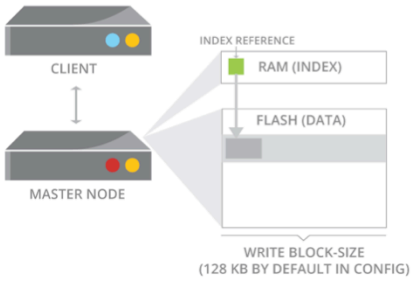
- Read
- Write*
- Update*
- Replace*
- Delete*

* The following slides do not illustrate how replica are written.

Accessing An Object In Aerospike

Reading A Standard Data Type With Flash

1) Client finds Master Node from partition map.
2) Client makes read request to Master Node.
3) Master node finds data location from index in RAM.
4) Master node reads entire object from flash. This is true even if only reading bin.
5) Master node returns value.

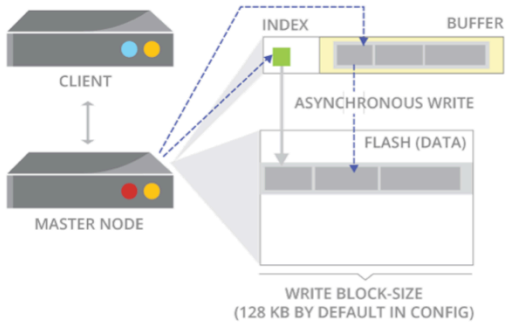


In this case, if the master node is down, the client may connect to a replica.

Accessing An Object In Aerospike

Writing A New Standard Data Type Record With Flash

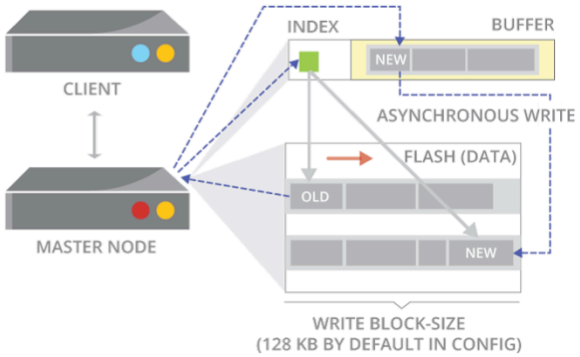
1) Client finds Master Node from partition map.
2) Client makes write request to Master Node.
3) Master Node make an entry into index (in RAM) and queues write in temporary write buffer.
4) Master Node coordinates write with replica nodes (not shown).
5) Master Node returns success to client.
6) Master Node asynchronously writes data in blocks.
7) Index in RAM points to location on flash.



Accessing An Object In Aerospike

Updating A Standard Data Type Record With Flash

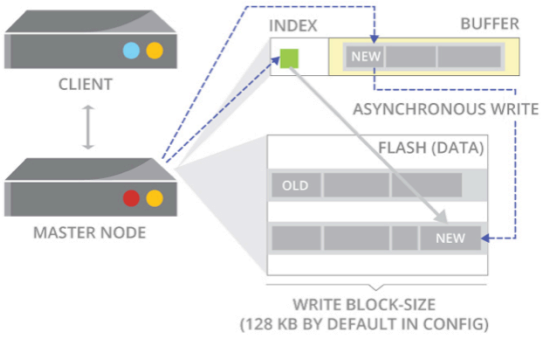
1) Client finds Master Node from partition map.
2) Client makes update request to Master Node.
3) Master Node reads the existing record
4) Master Node queues write of updated record in a temporary write buffer
5) Master Node coordinates write with replica nodes (not shown).
6) Master Node returns success to client.
7) Master Node asynchronously writes data in blocks.
8) Index in RAM points to new location on flash.



Accessing An Object In Aerospike

Replacing A Standard Data Type Record With Flash

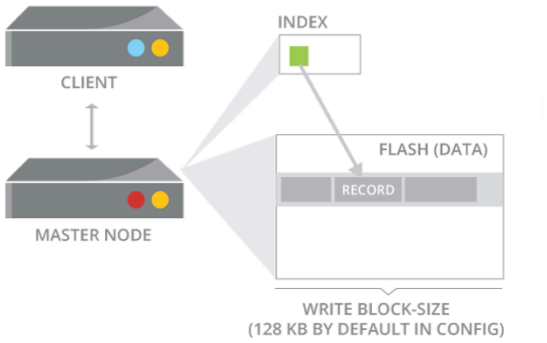
1) Client finds Master Node from partition map.
2) Client makes replace request to Master Node.
3) Master Node queues write of replaced record in a temporary write buffer
4) Master Node coordinates replace with replica nodes (not shown).
5) Master Node returns success to client.
6) Master Node asynchronously writes data in blocks.
7) Index in RAM points to new location on flash.



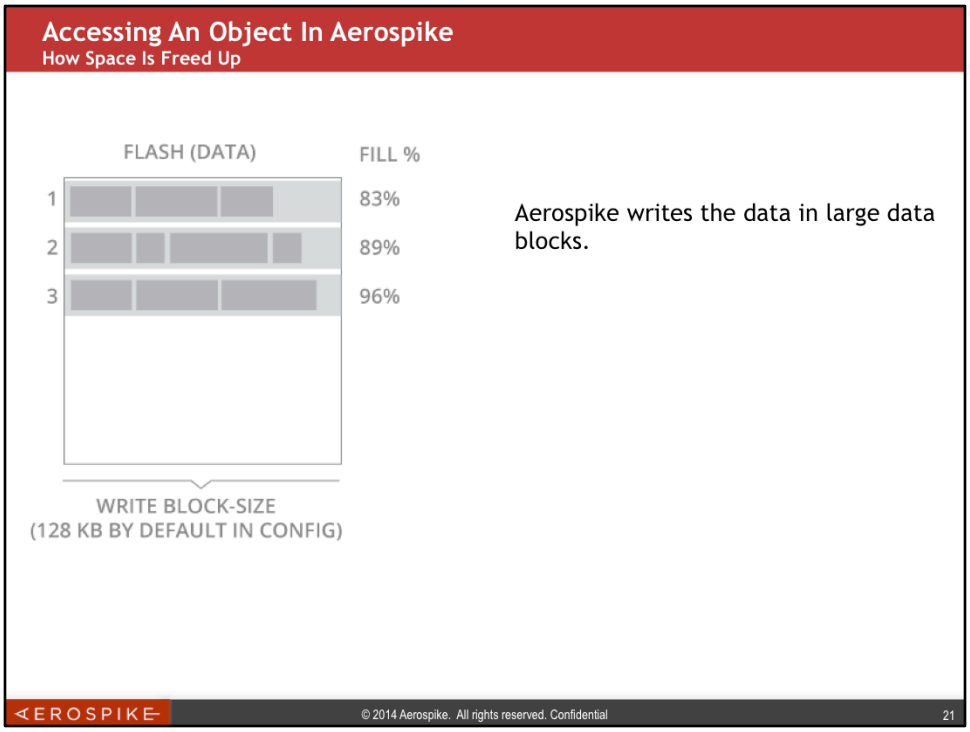
Accessing An Object In Aerospike

Deleting A Standard Data Type Record With Flash

1) Client finds Master Node from partition map.
2) Client makes delete request to Master Node.
3) Master Node coordinates deletion with replica nodes (not shown).
4) Master Node returns success to client.
5) Data is eventually deleted from flash by defragmentation.



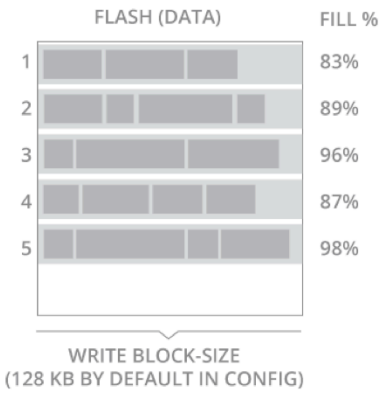

Defragmentation
AEROSPIKE
© 2014 Aerospike. All rights reserved. Confidential
20



As each buffer gets filled, it will write the block onto Flash.

Accessing An Object In Aerospike

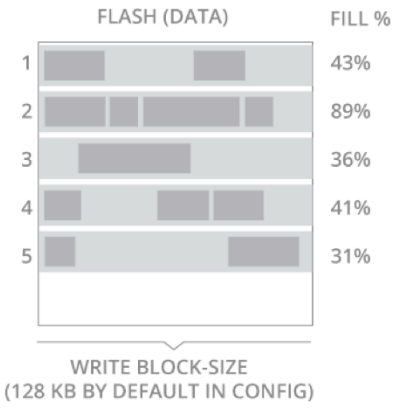
How Space Is Freed Up



As new data is added to the disk, new blocks will be continually written to the flash device.

Accessing An Object In Aerospike

How Space Is Freed Up



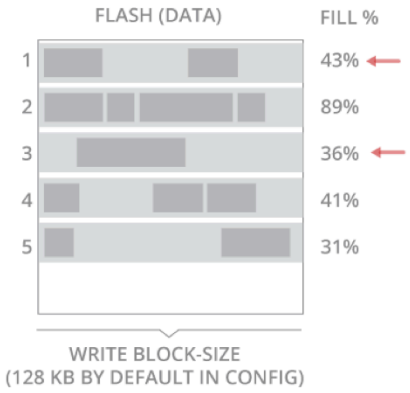
Over time, some records will be deleted or updated, resulting in fragmented usage on the flash/SSD disk. This unused space must be freed up.

The amount of space actually used is called the disk used percent. However the percent available on disk is actually represented by the percentage of free blocks (those at 0% filled).

This means it is possible for a flash device to be 50% utilized, but only have 20% available. The rest being taken by unused space in each block.

Accessing An Object In Aerospike

How Space Is Freed Up

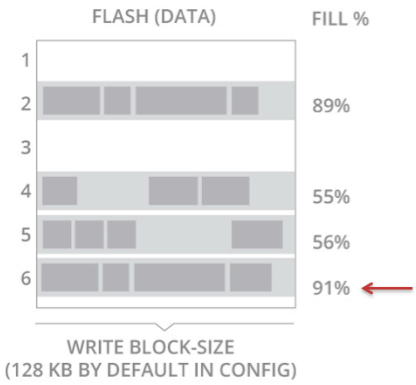


Some databases use a nightly process called “compaction,” which is an intensive process. Aerospike runs a regular process (every few minutes) that looks for blocks below some level of use (called the defrag-lwm-pct).

In this example, if the low watermark is 50%, blocks 1 and 3 to the left are below 50% occupied.

Accessing An Object In Aerospike

How Space Is Freed Up

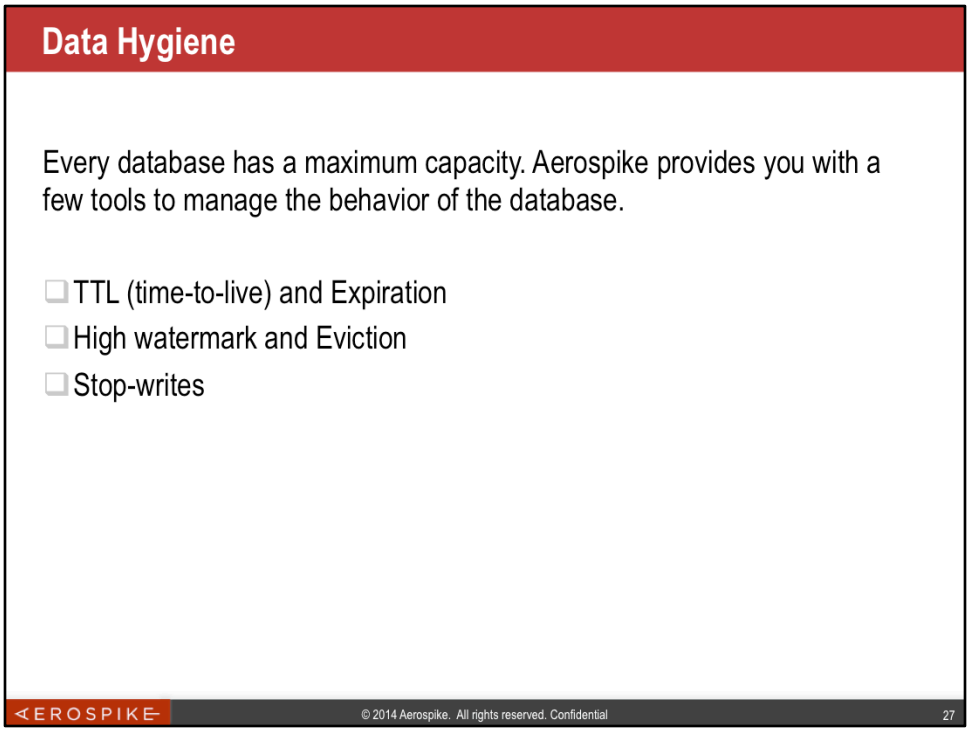


The defragmenter will take the data in these blocks and put the used data back into the write queue, where each record in the blocks will be treated like any new data.

Because this runs constantly, there is no special time where the performance of the database is bad.

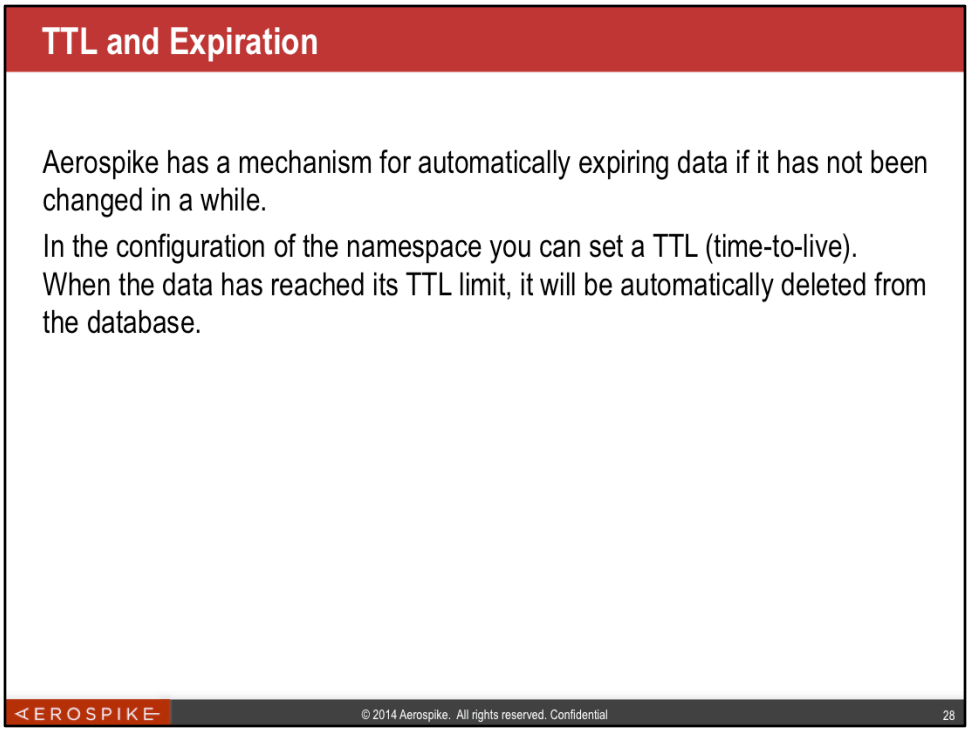
This algorithm operates best when the flash device is less than 50% occupied. As disk use grows above this, the performance of the defragmenter will decrease.


Data Hygiene
AEROSPIKE
© 2014 Aerospike. All rights reserved. Confidential
26



In most cases the DBA is responsible for tracking the policies on what happens when space gets tight.

Each of these are policies that will help you control the use of space within your database.



Correct use of the TTL is imperative for the health of the database.

TTL and Expiration

This works using the following logic:

- When a record is first written, the server will take the current time and add the TTL. The server will use the default TTL as configured on the server or, if supplied, an overriding TTL from the client.
- If the record is updated the TTL will be extended as if it were a new write. You can override this from the client.
- A “touch” from the client has the same impact, but does not change the data.
- Reads do not affect the expiration time.
- When the expiration time has been reached a background job will delete the data.

TTL and Expiration

Special Note:

If you do not want data to expire, but want to keep all data, you can set the TTL value to 0.

However, if you have set the TTL to any non-zero value and want it to not expire, you can set it to a negative value. This will set the record to not expire automatically. The only way to delete the record is manually.

High Watermarks and Eviction

One of the most dangerous things that can happen to a database is to reach maximum capacity.

Aerospike has a set of safety measures that will allow you to minimize the chances of hitting that limit and taking corrective action before then.

High Watermarks and Eviction

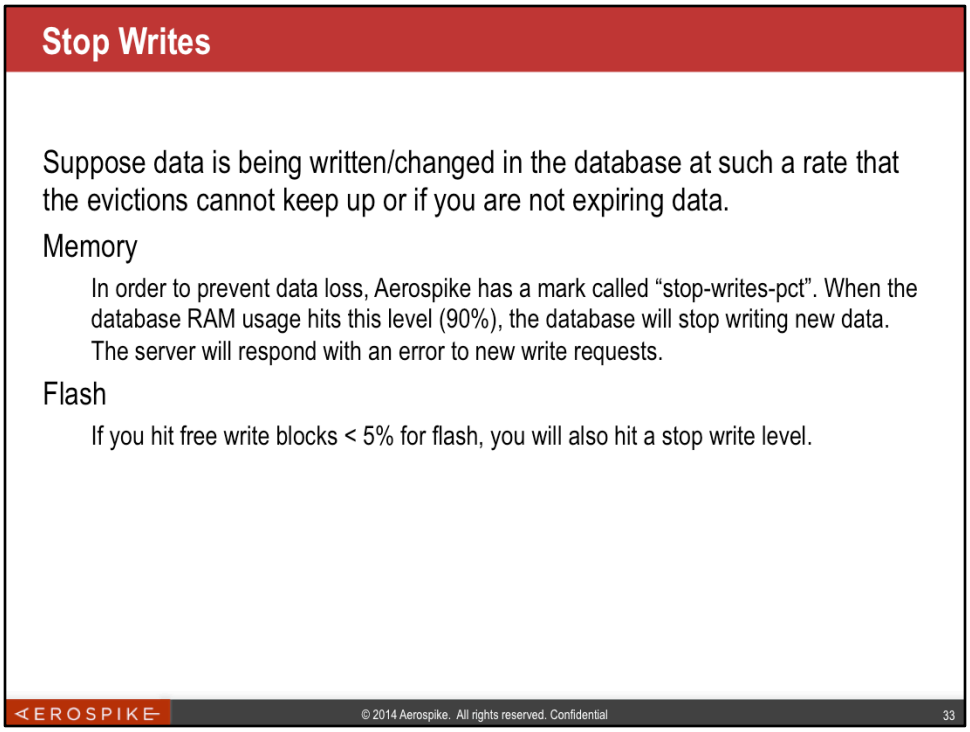
A high watermark is a threshold for RAM or disk use. When the watermark has been breached, the server will begin to evict data. It will start by evicting data closest to its TTL. This can be thought of as an advanced expiration.

The server put all data into 100 bins of time. Any data in the bin nearest to its void time will be evicted until enough room has been cleared or the server will move to the next bin.

For example:

A namespace has a 100 day TTL. It will automatically expire data that is closest to expiration, the 99 day bin.

Important note: If you have data with wildly different TTLs, such as 100 days and 5 minutes, the data with the 5 minute TTL will all be in the first bin to expire. The server does not discriminate between the data in the same bin. So be careful with your management of data.



Obviously, it is bad if the database is in read-only mode, but it is better than running out of space and being uncertain if data was written.


Configuring Storage
AEROSPIKE
© 2014 Aerospike. All rights reserved. Confidential
34

Configuration File

There are 7 major contexts in an Aerospike configuration file.

- service (required, covered in Part 1)
- logging (required, covered in Part 1)
- network (required, covered in Part 1)
- mod-lua (required for 3.x, covered in Part 1)
- cluster (optional, not covered)
- namespace (at least 1 required)
- xdr (optional, not covered)

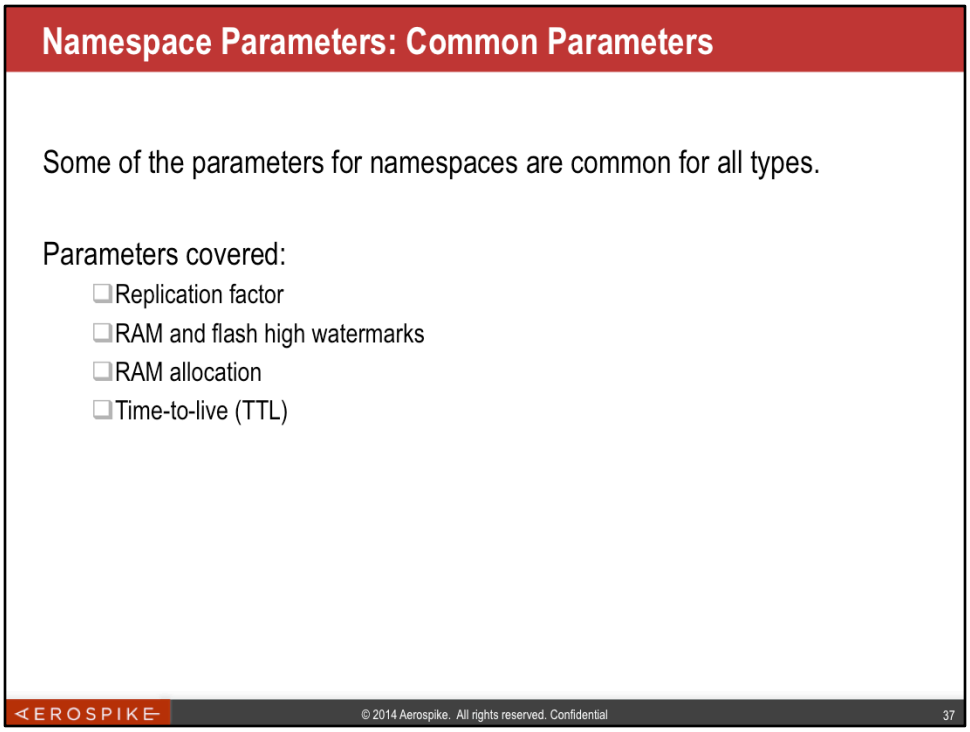
These will look like this:

```
namespace <NAMESPACE NAME> {
        ...
}
```

Namespace Configuration Parameters

Each namespace must be configured with 2 sets of variables:

- Common namespace variables.
- Storage variables
  - RAM (with or without HDD for persistence)
  - Flash/SSD



Each of these parameters will exist for all namespaces. While the values may not exist in the configuration file, there are default values.

Replication Factor

Description	The total (master + replica) copies of data in the database cluster.
Context location	namespace
Config parameters (defaults)	replication-factor
Notes	Some databases refer to the replication factor as being only the number of copies, without the master. Aerospike refers to it as the total number of copies.
Change dynamically	No
Best practices	For almost all use cases, the best replication factor is “2”. “1” would not give any replication and means that the loss of a node means a loss of data. “3” or more would mean you would need additional storage to accommodate the additional copies.

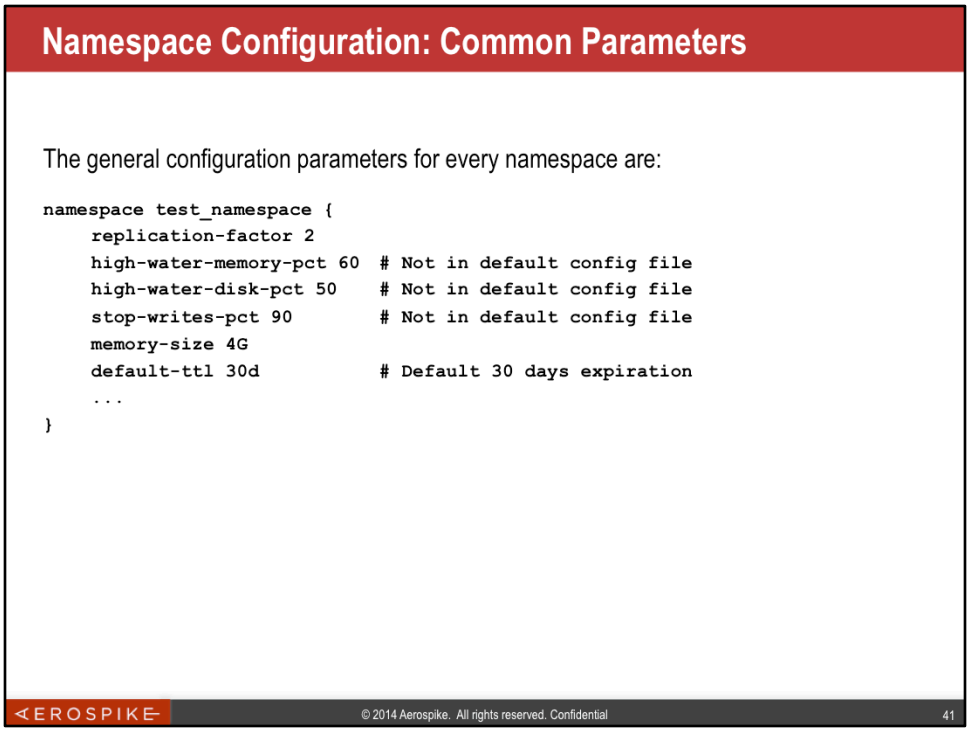
Storage Watermarks

Description	Aerospike has built in a set of watermarks that trigger actions meant to act as safety valves.
Context location	namespace
Config parameters (defaults)	high-water-memory-pct high-water-disk-pct stop-writes-pct
Notes	These features do not exist in most other databases, so some care should be taken to understand how these work. If either the high-water-memory-pct or high-water-disk-pct, the server will begin to evict data closest to its time-to-live (TTL). If the either RAM or disk should exceed the stop-writes-pct, the server will no longer accept write requests for new records (note that it will accept updates to existing records).
Change dynamically	Yes

Best practices	• Recommended settings are:
	  • high-water-memory-pct 60
	  • high-water-disk-pct 50
	  • stop-writes-pct 90
	• You should always account for changes in node could within the cluster. So consider what happens if you were to lose a node.
	• These settings are dynamically changeable, if you do want to increase these values temporarily, makes sure to take the appropriate corrective action.
	• In order to properly defragment Flash/SSD namespaces, keep the high-water-disk at 50.

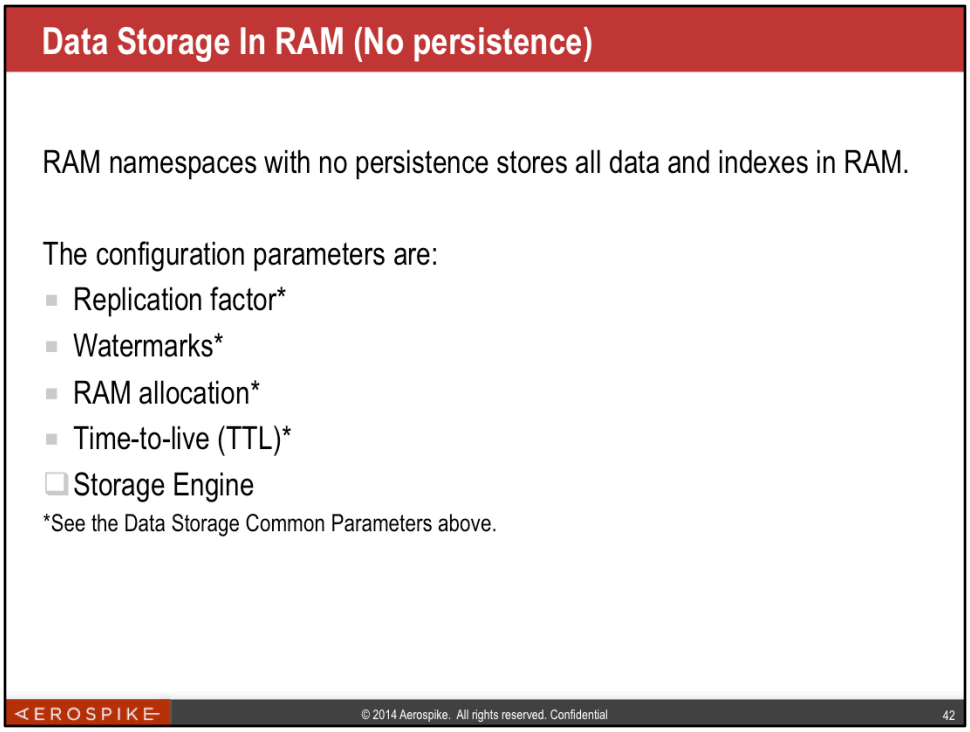
RAM Allocation

Description	All namespaces (whether RAM or Flash/SSD) require RAM for at least the index. This controls how much RAM is allocated.
Context location	namespace
Config parameters (defaults)	memory-size (4G)
Notes	Aerospike does not grab all of this memory when the node starts up. The minimum size for this is 128 MB. Aerospike uses exactly 64 bytes of memory for each record. This will be multiplied by the replication factor. If you are using a RAM namespace, all data will also be stored in RAM. Aerospike does not cache data.
Change dynamically	Yes
Best practices	Always be aware that the amount of RAM in use will increase if the cluster loses a node. The amount of memory set here should also take into account the high water marks for memory above.



There are some variables we cover here that will not be in the default config file. What is shown above are the values that the database will use as defaults if they are not in the file.

While you may want to adjust the settings for the memory high water mark, it is highly recommended to not adjust the settings for disk.



The only additional parameter you need to set for this is the storage engine.

Storage Engine for RAM (No Persistence)

Description	Sets how data will be stored.
Context location	namespace
Config parameters (defaults)	storage-engine memory
Notes	Aerospike does not cache. So all data (index + data) will be stored in RAM.
Change dynamically	No
Best practices	Be sure to take into consideration what will happen if you lose one or more nodes. See the Storage Watermark section above.

Namespace Configuration: Common Parameters

The general configuration parameters for every namespace are:

```
namespace test_namespace {
    replication-factor 2
    high-water-memory-pct 60  # Not in default config file
    high-water-disk-pct 50    # Not in default config file
    stop-writes-pct 90        # Not in default config file
    memory-size 4G
    default-ttl 30d           # Default 30 days expiration
    storage-engine memory
}
```


Some of these are not in the default config file, but the shown values are the default if they are not specified.

Data Storage In RAM + HDD

RAM namespaces with persistence (RAM +HDD) stores all data and indexes in RAM, but stores the data on disk. Note that although we say "HDD," you may also use Flash (SSDs).

The configuration parameters are:

- All the values from Data Storage in RAM above
- ❑ Storage Engine
- ❑ Persistence File
- ❑ Whether or not to store data in memory

Storage Engine for RAM + HDD (With Persistence)

Description	Sets how data will be stored.
Context location	namespace
Config parameters (defaults)	storage-engine device
Notes	Aerospike does not cache. So all data (index + data) will be stored in RAM. The device will only be used for persistence and will only be used when starting up the node. The server will use this to rebuild the index in memory.
Change dynamically	No
Best practices	Be sure to take into consideration what will happen if you lose one or more nodes. See the Storage Watermark section above.

RAM + HDD Persistence File

Description	These are the settings for the persistence file
Context location	namespace:storage-engine
Config parameters (defaults)	**file** or **device** filesize
Notes	When using a file, you specify the path of the file. When using the device, you specify the device (such as "/dev/sdb1") of the disk or partition. You must also specify the filesize. When using device, the server assumes it will use the entire disk, so please make sure there is no data on this partition.
Change dynamically	No
Best practices	• Aerospike recommends setting the size of the persistence file at 4x the amount of RAM allocated in the "memory-size" of the namespace. • You may use more than one file or device, but the server will balance between them.

Store Data In Memory

Description	Tells the server to store data in memory, rather than on storage.
Context location	namespace:storage-engine
Config parameters (defaults)	data-in-memory
Notes	When set to “true”, the server will store data in RAM and use disk only for persistence. If set to “false”, the server will only store the index in RAM and use the disk for data storage.
Change dynamically	No
Best practices	• In principle it is possible to use rotational disk to store the data and use RAM only for the index. Aerospike has found that performance with this configuration is very poor. Aerospike does not recommend or support this configuration.

Namespace Configuration: RAM + HDD

The configuration parameters for RAM + HDD are:

```
namespace RAM_persist_namespace {
    replication-factor 2
    high-water-memory-pct 60      # Not in default config file
    high-water-disk-pct 50        # Not in default config file
    stop-writes-pct 90            # Not in default config file
    memory-size 4G
    default-ttl 30d               # Default 30 days expiration
    storage-engine device {
        file /opt/aerospike/data/test.data
        filesize 16G
        data-in-memory true
    }
}
```

Data Storage In Flash/SSD

Flash/SSD namespaces store primary indexes in RAM and all data on Flash/SSD.

The configuration parameters are:

- Common Data Storage Parameters
- Storage Engine
- Flash/SSD Devices
- Whether or not to store data in memory

*See the Data Storage Common Parameters above.

Storage Engine for Flash/SSD

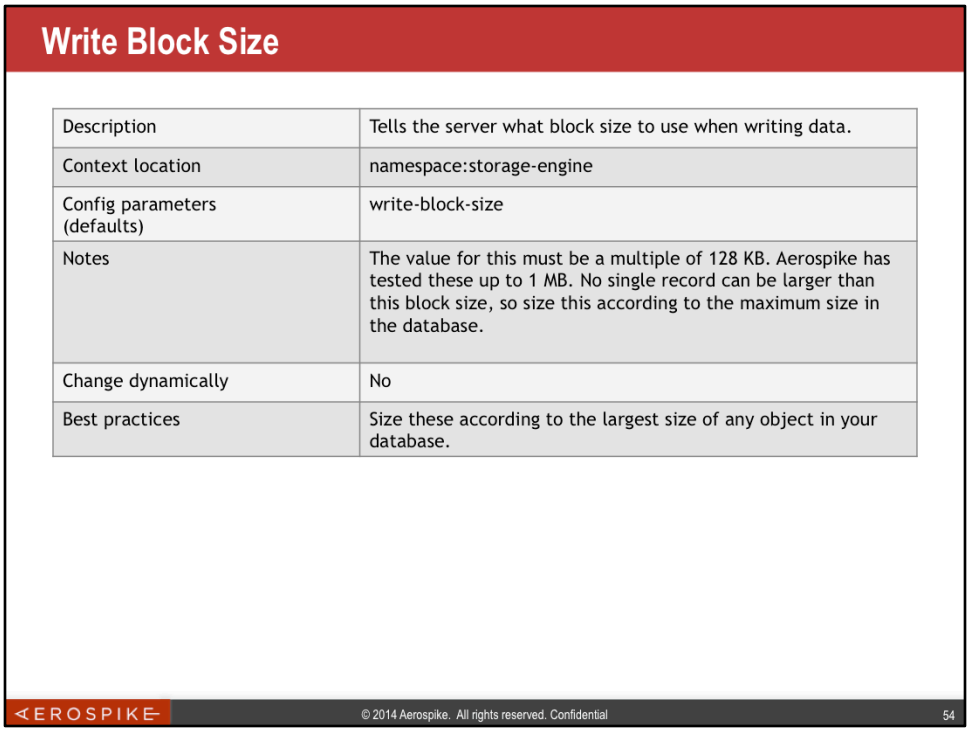
Description	Sets how data will be stored.
Context location	namespace
Config parameters (defaults)	storage-engine device
Notes	The server will use RAM for index only and Flash/SSD for data.
Change dynamically	No
Best practices	• Be sure to take into consideration what will happen if you lose one or more nodes. See the Storage Watermark section above. • Flash/SSDs should be no more than 50% full to allow for efficient defragmentation. Usage at above this rate is fine for short periods of time.

Flash/SSD Devices

Description	These are the settings for the Flash/SSD devices
Context location	namespace:storage-engine
Config parameters (defaults)	device
Notes	You must specify the device (such as /dev/sdb) of the Flash/ SSD.
Change dynamically	No
Best practices	• You may use more than one file or device, but the server will balance between them. Do not use heterogeneously sized devices. • You may use SATA, SAS, or PCIe devices.

Store Data In Memory

Description	Tells the server to store data in memory, rather than on storage.
Context location	namespace:storage-engine
Config parameters (defaults)	data-in-memory
Notes	When set to “true”, the server will store data in RAM and use disk only for persistence. If set to “false”, the server will only store the index in RAM and use the disk for data storage.
Change dynamically	No
Best practices	• In principle it is possible to use Flash/SSD as persistence only. However, most modern rotational hard drives are fast enough so that using Flash/SSD is not necessary. • The drives must be cleared or “dd”ed prior to use in the database. • Be sure to test the drives in the servers using the Aerospike ACT test tool (http://github.com/aerospike/act/).



You can increase the block size, but cannot decrease the block size.

Namespace Configuration: Flash/SSD

The configuration parameters for Flash/SSD

```
namespace ssd_namespace {
    replication-factor 2
    high-water-memory-pct 60      # Not in default config file
    high-water-disk-pct 50        # Not in default config file
    stop-writes-pct 90            # Not in default config file
    memory-size 4G
    default-ttl 2592000           # Default 30 days expiration
    storage-engine device {
        device /dev/sdb
        device /dev/sdc
        data-in-memory false
        write-block-size 128K
    }
}
```

Namespace Configuration:

Comparing the configurations of RAM + HDD and Flash/SSD:

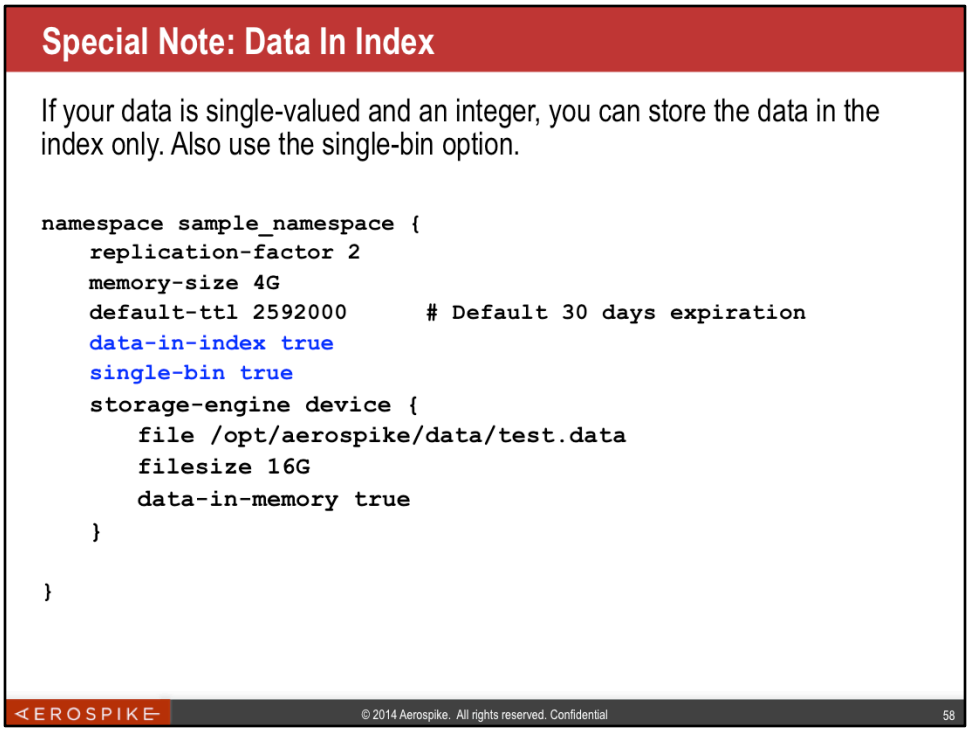
```
namespace RAM_persist_namespace {
    replication-factor 2
    high-water-memory-pct 60
    high-water-disk-pct 50
    stop-writes-pct 90
    memory-size 4G
    default-ttl 30d
    storage-engine device {
        file /opt/aerospike/data/test.data
        filesize 16G
        data-in-memory true
    }
}
```

```
Namespace SSD_persist_namespace {
    replication-factor 2
    high-water-memory-pct 60
    high-water-disk-pct 50
    stop-writes-pct 90
    memory-size 4G
    default-ttl 30d
    storage-engine device {
        device /dev/sdb
        device /dev/sdc
        data-in-memory false
        write-block-size 128K
    }
}
```

Special Note: Single Bin Optimization

If you have certain kinds of data that belongs in a single bin, you can turn on the single bin optimizations. This will store the data more compactly. To turn it on, simply add the parameter “single-bin” as below.

```
namespace sample_namespace {
    replication-factor 2
    memory-size 4G
    default-ttl 2592000      # Default 30 days expiration
    single-bin true
}
```



Using data in index means you will only need 64 bytes of RAM for every record (multiplied by the replication factor) for every record in the cluster.

If you use the single-bin option and persistence and are using the Enterprise Edition, you can also use Faststart. This means that the index and the included data will be stored in shared memory. If you do not power down, you can restart the database, which will reconnect to the shared memory and be able to get going without having to read data from persistence.

Summary

What we have covered:

- Data Hierarchy
- How Aerospike Reads/Writes/Updates Data
- Defragmentation
- Data Hygiene
- Configuring Storage