Cluster Design:

Exploiting Rack Aware in Strong Consistency Mode

Meher Tendjoukian
VP Operations
Aerospike

The Challenge

Maintain data consistency and availability in the face of catastrophic failure with minimal operator overhead

Exploiting Rack Aware in Strong Consistency Mode

- **Strong Consistency (SC) Overview**
- **Aerospike Rack Aware (RA)**
- **Failure modes when using RA in SC mode**
- **Demo**

Strong Consistency Overview

- The strong consistency guarantee states that all writes to a single record will be applied in a specific order (sequentially), and writes will not be re-ordered or skipped.
- In particular, writes that are acknowledged as committed have been applied, and exist in the transaction timeline in contrast to other writes to the same record. This guarantee applies even in the event of network failures, outages and partitions.
- AP vs. SC:
  - In SC mode, in case of a cluster splitting into multiple sub-clusters (aka split brain) at most one of those sub-clusters will be available and some partitions may not be available at all.
  - In AP mode, all partitions are always available in any sub-cluster. Sub-clusters can be in situations where fresh (empty) partitions are created.

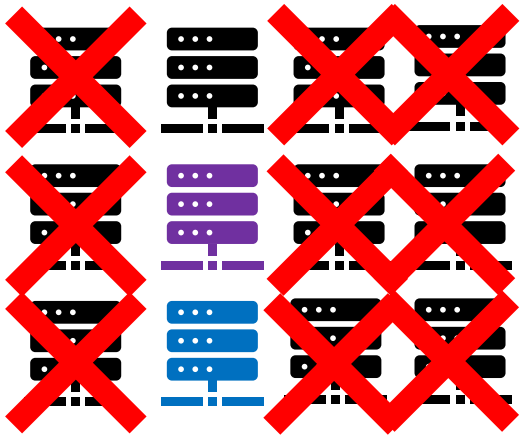
Strong Consistency Overview - Terminology

- *roster*
  - The *roster* defines the list of nodes that are part of the cluster in steady state. When all the *roster* nodes are present and all the partitions are current, the cluster is in its steady state and provides optimal performance.
- *roster-master*
  - For a specific partition, the *roster-master* refers to the node that would house the master of this partition if all nodes in the roster were part of the single cluster, i.e. the cluster was whole.
- *roster-replica*
  - For a specific partition, the *roster-replica* refers to the node that would house the replica of this partition if all nodes in the roster were part of the single cluster, i.e., the cluster was whole.

Quick review of Strong Consistency – Availability Rules

The following rules apply to the visibility (availability) of partitions:

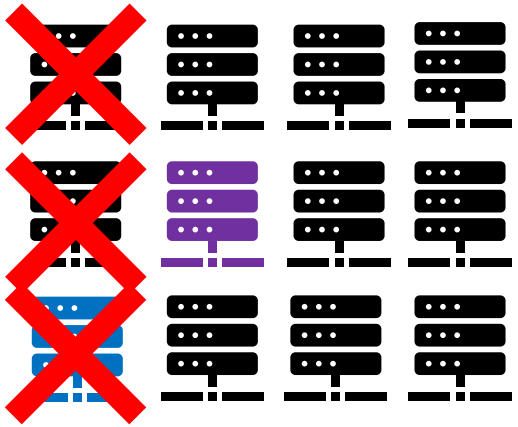
- If a sub-cluster has both the *roster-master* and *roster-replica* for a partition, then the partition is available for both reads and writes in that sub cluster.



Quick review of Strong Consistency – Availability Rules

The following rules apply to the visibility (availability) of partitions:

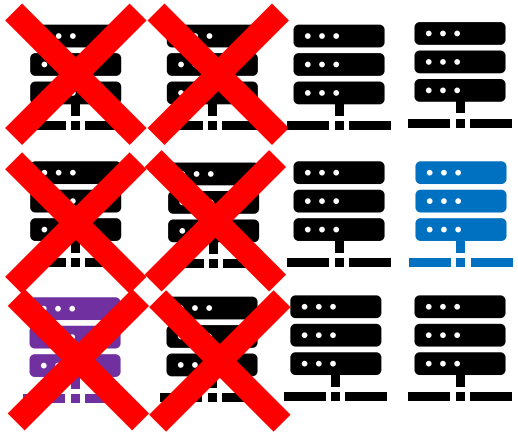
- If a sub cluster has a strict majority of nodes and has either the *roster-master* or *roster-replica* for the partition within its component nodes, the partition is available for both reads and writes in that sub cluster.



Quick review of Strong Consistency – Availability Rules

The following rules apply to the visibility (availability) of partitions:

- If a sub cluster has exactly half of the nodes in the full cluster (roster) and it has the roster-master within its component nodes, the partition is available for both reads and writes



Aerospike Rack Aware

- The Aerospike Rack Aware feature allows to store different replicas of partitions on different hardware failure groups.
  - For example, if *`replication-factor`* is 2, the master copy of a partition and its replica will be stored on different hardware failure groups.
  - These groups are defined by their *`rack-id`*.
  - In the cloud, these groups typically map to Availability Zones.

- If the *`replication-factor`* is greater than the number of racks, all racks will have a copy of the partition's data, but some racks will have additional copies of partitions, for each partition.

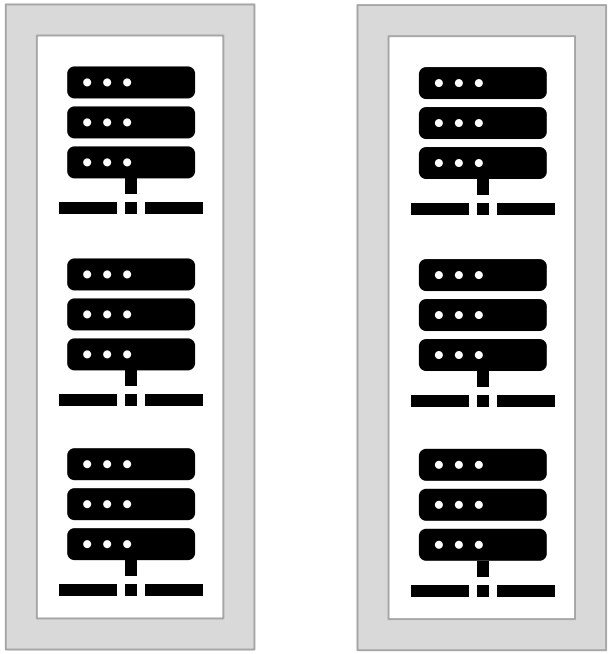
Aerospike Rack Aware

- Pros
  - All partitions have a full copy on each rack, therefore, if one rack fails (for example a power outage) all partitions are still present in the cluster.
  - Faster maintenance, bringing down a whole rack at a time, potentially leveraging `migrate-fill-delay` to avoid duplicating partitions within the same rack while a rack is down.
  - Client side rack aware allows for read transactions to stay in the same rack as the Client (saving on latency and money).

- Cons
  - Each write has to replicate on a different rack (potential extra latency and money).
  - Harder to size as the cluster should be able to handle all the load in the event a whole rack goes down.

Failure modes when using RA in SC mode

- **Cluster configuration:**
  - strong-consistency mode
  - 6 nodes
  - 2 racks with 3 nodes each: rack-id 1 and rack-id 2
  - replication-factor 2

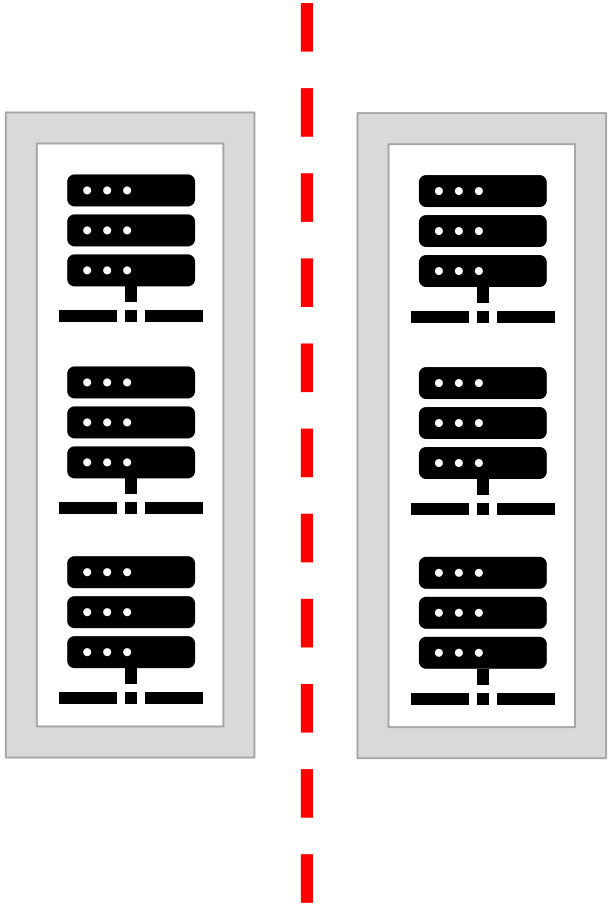


- **Some common failure situations:**
  - Network failure between the 2 racks
  - All nodes in rack with rack-id 1 go down (power outage)
  - All nodes in rack with rack-id 2 go down (power outage)

Failure modes when using RA in SC mode – Split brain

Network failure between the 2 racks

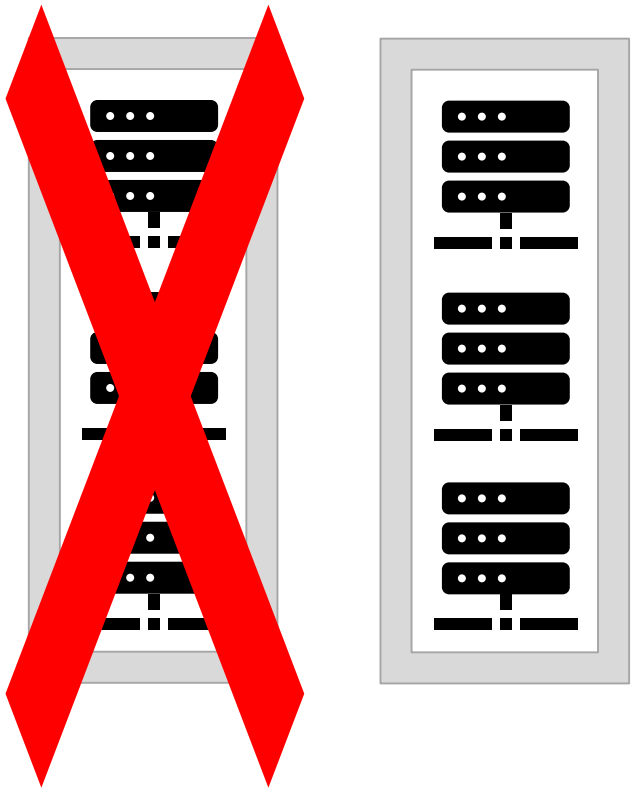
- Two 3 node sub-clusters
- All partitions are available across the cluster
- Each rack only has 50% of the partitions available
  - Rack with rack-id 1 has available the partitions it is *roster-master* for.
  - Rack with rack-id 2 has available the partitions it is *roster-master* for.
- It is assumed clients can still access both racks
  - If a client can see only 1 rack, it will only have access to 50% of the data.



Failure modes when using RA in SC mode

Rack with rack-id 1 goes down

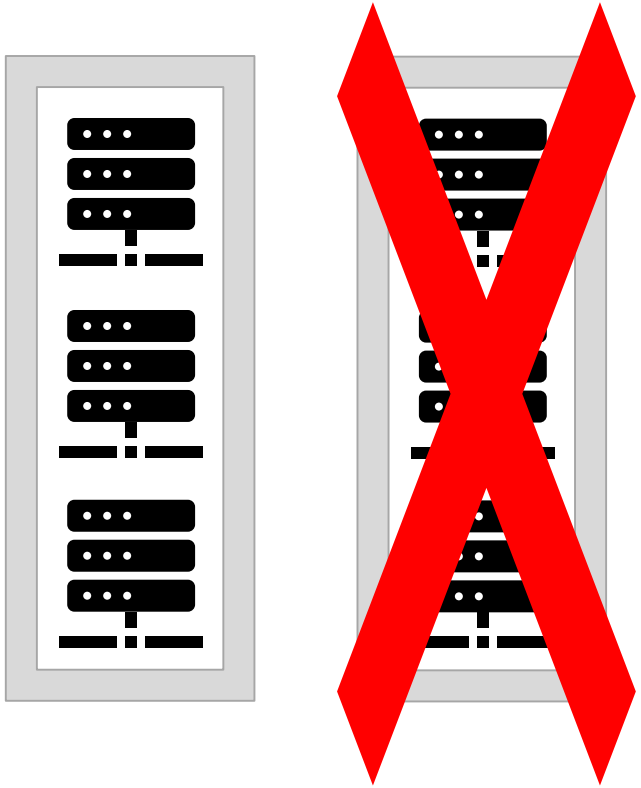
- 50% of the data available on rack with rack-id 2
- Rack with rack-id 2 has available the partitions it is *roster-master* for
- Rack with rack-id 2 has the remaining 50% of the partitions, but they cannot be made available
  - Rack with rack-id 2 does not know if the other rack is down or if it simply cannot communicate with it. It has to assume the other rack is taking writes.
- Can the *roster-replica* partitions be made available?
  - **Yes!** Re-roster with just the nodes from rack-id 2.



Failure modes when using RA in SC mode

Rack with rack-id 2 goes down

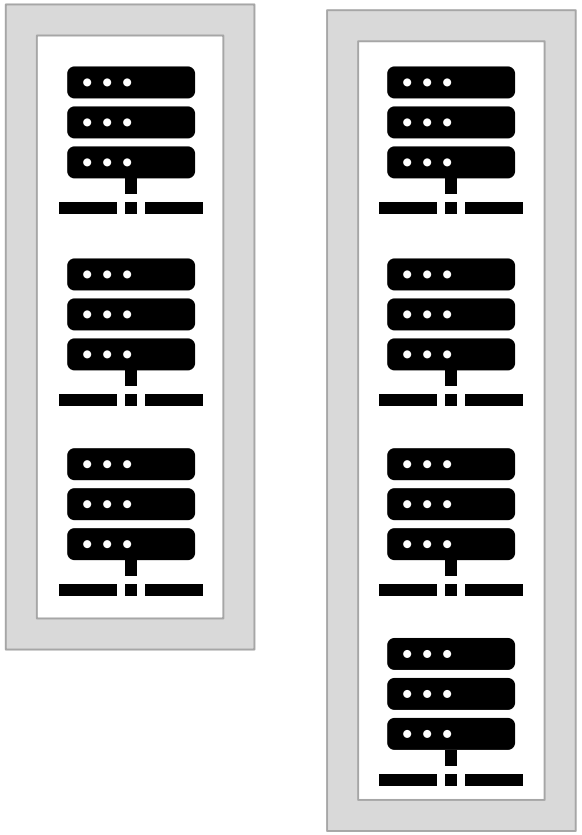
- 50% of the data available on rack with rack-id 1
- Rack with rack-id 1 has available the partitions it is roster-master for
- Same as previous case



Failure modes when using RA in SC mode – Alternate Design?

Can we do better?

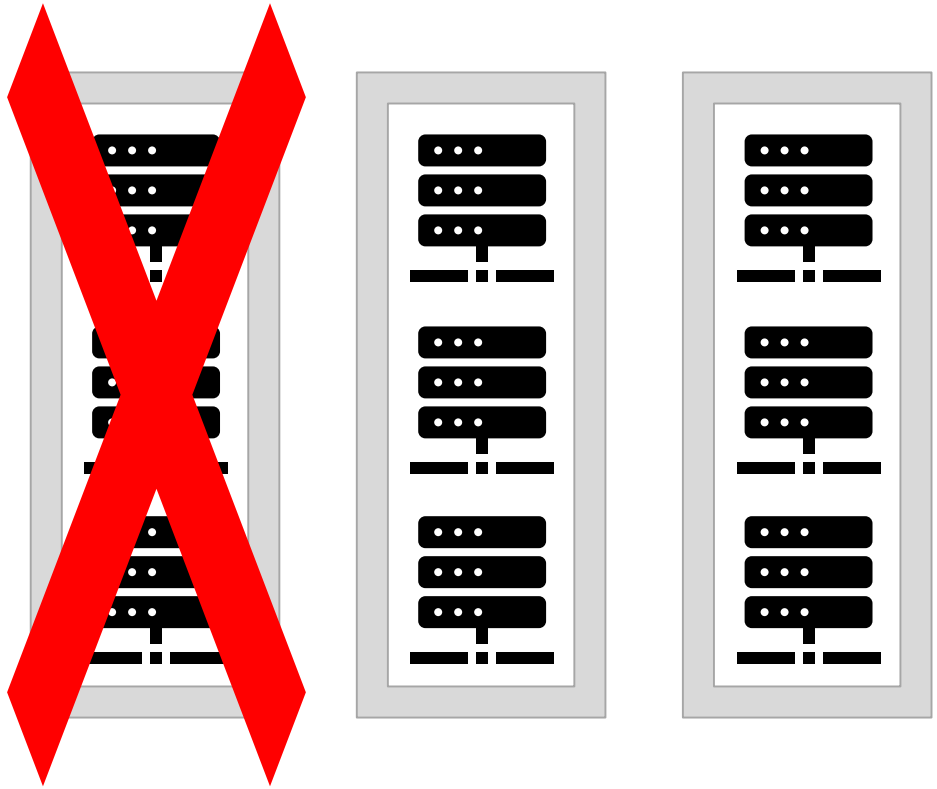
- Would be better to have situations where all the partitions are available in one rack
- What if the racks don't have the same number of nodes?
- Split brain -> larger rack is fully available, the smaller one is not available at all
- Small rack goes down -> large rack is fully available
- Large rack goes down -> small rack has all the data, need to re-roster for full availability
- Large rack gives us the strict majority in a cluster along with either *roster-master* or *roster-replica*



Failure modes when using RA in SC mode – Alternate Design?

3 Racks with RF 2

- Losing 1 rack always leaves a strict majority with full availability.



Demo

- **Setup fully running on laptop**
- **5 node cluster running version 4.5.2.1.**
- **Strong Consistency**
- **TLS for Client, Fabric and Heartbeat**
- **Uniform balance**
- **2 racks**
  - Rack with rack-id 1 has nodes 101, 102 and 103
  - Rack with rack-id 2 has nodes 204 and 205

- **Java Benchmark 4.4.0 for simulating read/write traffic**

The Challenge

Maintain data consistency and availability in the face of catastrophic failure with minimal operator overhead

In production since 2018 in European TIPS payment system where Aerospike powers real time payments

Q&A?

