Oracle® NoSQL Database

Availability and Failover

Release 12.2.4.5

E88250-01

February 2018

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Introduction

Oracle NoSQL Database is a data storage product with enormous scalability and performance benefits. Additionally, Oracle NoSQL Database offers excellent *availability* mechanisms. These mechanisms are designed to provide your applications access to data contained in the store in the event of localized hardware and network failures.

This document describes the mechanisms Oracle NoSQL Database uses to ensure your data remains available, along with the various failover algorithms that Oracle NoSQL Database employs. In addition, this document describes application design patterns you can use to best make use of Oracle NoSQL Database's availability mechanisms. In some cases, tradeoffs exist between ensuring data is highly available, and achieving optimal performance. This document explores these tradeoffs.

The intended audience for this document includes system architects, engineers, and others who want to understand the concepts and issues surrounding data availability when using Oracle NoSQL Database. In addition, software engineers responsible for writing code that interacts with an Oracle NoSQL Database store should also read this document.

This document assumes you have read and are familiar with the contents of the *Oracle NoSQL Database Getting Started with the Table API* guide or the *Oracle NoSQL Database Getting Started with the Key/Value API* guide. If you have not read

one, or both, of these manuals, we recommend that you do so before continuing. In particular, you should understand these concepts:

- *Oracle NoSQL Database Concepts Manual*

  This document introduces terms and concepts you need to know before reading this document.

- Durability Guarantees

  This section includes concepts that lead to issues surrounding write availability.

- Consistency Guarantees

  This section includes concepts that lead to issues surrounding read availability.

Replication Overview

To ensure data durability and availability, Oracle NoSQL Database uses a single-master replication strategy. Using a single machine to perform write operations, Oracle NoSQL Database then broadcasts those operations to multiple read-only replicas.

The *Oracle NoSQL Database Concepts Manual*describes a shard as a collection of replication nodes, associated with a single master node and multiple replicas. Your store contains multiple shards, and your data is spread evenly across all of the shards that your store uses.

When you perform a write operation in your store, Oracle NoSQL Database completes the write operation on the master node in use by the shard containing your data. The master node performs this write according to whatever durability guarantees are in place at the time. If you set a strong durability guarantee, the master requires the participation of some or all of the replicas in the shard to complete the write operation.

If the master node of the shard becomes unavailable for any reason, the replica nodes in primary zones hold an election to determine which of the remaining replication nodes should take over as the master node. The replication node with the most up-to-date data wins the election.

The election is decided based on a simple majority vote. This means that a majority of the nodes in the shard in primary zones must be available to participate in the election to select a new master.

Loss of a Read-Only Replica Node

A common fail over case is losing a replica node due to a problem with the machine upon which it is running. This loss can be due to something as common as a hard drive failure.

In this case, the only shard that is affected is the one using the replica. By default, the effect on the shard is reduced read throughput capacity. The shard itself is capable of continuing normal operations. However, losing a single Replication Node reduces its capacity to service read requests by whatever read throughput a single host machine offers your store. Whether you detect this reduction in read throughput capacity depends on how heavy a read load your shard is experiencing. The shard could have

a low enough read load that losing the replica results in a minor performance reduction.

Such a small performance reduction assumes that a single host machine contains only one Replication Node. If you configure your store so that multiple Replication Nodes run on a single host, then the loss of throughput capacity increases accordingly. It is likely that the loss of a machine running multiple Replication Nodes will affect the throughput capacity of more than one shard, because it is unlikely that all the Replication Nodes on that machine will belong to the same shard. Again, whether you notice any performance reduction from the loss of the Storage Node depends on how heavy a read load the individual affected shards are experiencing.

In this scenario, with one exception, the shard will continue servicing write requests, and may be able to do so with no changes to its write throughput capacity. The master itself is not affected, so it can continue performing writes and replicating them to the remaining replicas in the shard. There can be reduced write throughput capacity if:

- there is such a heavy read load on the shard that the loss of one replica saturates the remaining replica(s); and
- the master requires an acknowledgement before finishing a write commit.

In this scenario, write performance capacity can be reduced either because the master is continually waiting for the replica to acknowledge commits, or because the master itself is expending resources responding to read requests. In either case, you may see degraded write throughput, but the level of degradation depends on how heavy the read/write load actually is on the shard. Again, it is possible that you will never detect any write throughput reduction, because the write load on the shard is low.

In addition, the loss of a single read-only replica can cause all write operations at that shard to fail with a `DurabilityException` exception. This happens if you are using a durability guarantee that requires acknowledgements from all replicas in the shard in primary zones. In this case, writes at that shard will fail until either that replica is brought back online, or you place a less strict durability guarantee into use.

Using durability guarantees that require acknowledgements from all replicas in primary zones offer you the strongest data durability possible (by making certain that your writes are replicated to every machine in a shard). At the same time, they have the potential to lose write capabilities for an entire shard from a single hardware failure. Consequently, be sure to balance your durability requirements against your availability requirements, and configure your store and related code accordingly.

Loss of a Read/Write Master

If you lose a host machine containing a shard's master, the shard will be incapable of responding to write requests, momentarily. The lack of write request response is so brief that it may not be detected by your client code. Only the shard containing the master is affected by this outage. All other shards continue to perform as normal.

In this case, the shard's replicas in primary zones will quickly notice the master is missing and call for an election. Typically this will occur within a few milliseconds after losing the master.

The replica nodes will conduct an election, and the replica in a primary zone with the most up-to-date set of data will be elected master. To be elected master requires a simple majority vote from the other machines in the shard hosting nodes in primary zones. Keep in mind that this simple majority requirement has implications if many machines are lost from your store.

Once a new master is elected, the shard will continue operations, reducing its read throughput capacity by one machine. As with the loss of a single replica (see the previous section), all write operations can continue as long as your durability guarantee does not require acknowledgements from all replicas in primary zones.

Your client code will not notice the missing master if the new master is elected and services the write request within the timeout value used for the write operation. However, we recommend that your production code include ways to guard against timeout problems. In the event of a timeout, your code should include a decision policy about what to do next. For example, your policy could:

- Retry the write operation immediately,
- Retry the write operation after a defined wait,
- Abandon the write operation entirely.

Unplanned Network Partitions

A shard can be split into two, non-communicating networks. Such an event can occur when a piece of network hardware, such as a router, fails in some way that divides the shard. The store's response to such an event depends on how the network partition divides the shard's Replication Nodes as in these three cases:

A single Replication Node is isolated from the rest of the shard. If the Replication Node is a read-only replica, the shard continues operating as normal, but without the read throughput capacity caused by the loss of a single machine. See Loss of a Read-Only Replica Node for more details.

A single Replication Node becomes isolated from the rest of the shard. If the Replication Node is a master, the shard handles the event in the same way as if it had lost a master. The shard holds an election to select a new master and then continues operating as normal. See Loss of a Read/Write Master for further information.

The new network partition divides the shard into two or more groups of machines. In this case, there will be at least one *minority node partition.* A minority node partition contains less than a majority of the Replication Nodes in the shard. There could also be a *majority node partition.* A majority node partition has the majority of nodes in the shard —. However, a majority node partition is not a given, especially if the new network partition creates more than two sets of Replication Nodes.

How failover is handled in this scenario depends on whether a majority node partition does exist, and if the master exists in that partition. There are also other issues to consider, such as the durability and consistency policies that were in use at the time the new network partition was created.

Master is in the Majority Node Partition

Suppose the shard is divided into two partitions. Partition A contains a simple majority of the Replication Nodes in primary zones, including the master. Partition B has the remaining nodes.

- Partition A continues to service read and write requests as normal, but with a reduced read throughput from the loss of however many Replication Nodes are in Partition B. A caveat in this situation is what durability policy is in use at the time. If Partition A does not have enough replicas from primary zones to meet the durability policy requirements, it could be prevented from servicing write requests. If the durability policy requires a simple majority, or less, of replicas, then the shard will be able to service write requests.
- Partition B continues to service read requests as normal, but with increasingly stale data. Depending on the consistency guarantee in place, Partition B might cease to service read requests. If a version-based consistency is in use, then Partition B will probably encounter `ConsistencyException` exceptions soon after the network partition occurs, due to its inability to obtain version tokens from the master. Similarly, if a time-based consistency policy is in use, then `ConsistencyException` exceptions will occur as soon as the replica lags too far behind the master, from which it is no longer receiving write updates. By default, a consistency guarantee is not required to service read requests. So unless you explicitly create and use a consistency policy, Partition B can continue to service read requests through the entire network outage.

  Partition B will attempt to elect a new master, but will be unable to do so because it does not contain the simple majority of Replication Nodes required to hold an election.

Further, if the partition is such that your client code can reach Partition A but not Partition B, then the shard will continue to service read and write requests as normal, but with a reduced read capacity.

However, if the partition is such that your client code can read Partition B but not Partition A, then the shard will be unable to service any write requests. This is because Partition A contains the master, and Partition B does not include enough Replication Nodes to elect a new master.

Master is in the Minority Node Partition

Suppose the shard is divided into two partitions. Partition A contains a simple majority of the Replication Nodes from primary zones, but NOT the master. Partition B has the remaining nodes, including the master.

Assuming both partitions are network accessible by your client code, then:

- Partition A will notice that it no longer has a master. Because Partition A has at least a simple majority of the Replication Nodes in primary zones, it will be able to elect a new master. It will do this quickly, and the shard will continue operations as normal.

  Whether Partition A can service write requests is determined by the durability policy in use. As long as the durability policy requires a simple majority, or less, of replicas, then the shard is able to service write requests.

- Partition B will continue to operate as normal, believing that it has a valid master. However, the only way Partition B can service write requests is if the durability policy in use requires no participation from the shard's replicas. If a majority of nodes in primary zones must acknowledge the write operation, or if all nodes in primary zones must acknowledge the write, then the partitions will be unable to service writes because not enough nodes are available to satisfy the durability policy.

  If durability NONE is in use, then for the period of time that it takes to resolve the network partition, the shard will operate with two masters. When the partition is resolved, the shard will recognize the problem and correct it. Because Partition A held a valid election, writes performed there will be kept. *Any writes performed in Partition B will be discarded.* The old master in Partition B will be demoted to a simple replica, and the replicas in Partition B will all be synced with the new master.

  > **Note:**
  >
  > Because of the potential for loss of data in this scenario, Oracle *strongly* recommends that you do NOT use durability NONE. The only time you should use that durability setting is if you want to absolutely maximize write throughput, and do not care if you lose the data.

Further, if the partition is such that your client code can reach Partition A but not Partition B, then the shard will continue to service read and write requests as normal, but only after an election is held, and then with a reduced read capacity.

However, if the partition is such that your client code can read Partition B but not Partition A, then the shard will be unable to service write requests at all, unless you use the weakest durability policy available. This is because Partition B does not include enough Replication Nodes to satisfy anything other than the weakest available durability policy.

No Majority Node Partition

Suppose the shard is divided into multiple partitions, and no partition contains a majority of the Replication Nodes in the shard. In this case, the shard's partitions can service read requests, so long as the consistency policy in use for the read supports it. If the read requires tight consistency with the master, and the master is not available to ensure the consistency can be met, then the read will fail.

The partition containing the master can service write requests only if you are using the weakest available durability policy, in which no acknowledgements from replicas are required. If acknowledgements are required, then there will not be enough replicas to satisfy the durability policy and no write operations can occur.

Once the network partition is resolved, the shard will elect a new master, synchronize all replicas with it, and continue operations as normal.

Zone Failover

Zones allow you to spread your data store across various physical installation locations. The different locations can be anything from different physical buildings near each other, to different racks in the same building. The basic goal of spreading your store across locations is to guard against large-scale infrastructure disruptions, such as power outages or major storm damage, by placing the nodes in your store physically as far apart as possible.

Oracle NoSQL Database provides support for two kinds of zones. *Primary* zones contain nodes which can serve as masters or replicas. Zones are created as primary zones by default. *Secondary* zones contain nodes which can serve only as replicas. Secondary zones can be used to make a copy of the data available at a distant location, or to maintain an extra copy of the data to increase redundancy or read capacity.

Both types of zones require high throughput network connections to transmit the replication data required to keep replicas up-to-date. Failing to provide sufficient network capacity will result in nodes in poorly connected zones falling farther and farther behind. Locations connected by low throughput network connections are not suitable for use with zones.

For primary zones, in addition to a high throughput network, the network connections with other primary zones should provide highly reliable and low latency communication. These capabilities make it possible to perform master elections for quick master failovers, and to provide acknowledgments to meet write request timeout requirements. Primary zones are not, therefore, suitable for use with an unreliable or slow wide area network.

For secondary zones, the nodes do not participate in master elections or acknowledgments. For this reason, the system can tolerate reduced reliability or increased latency for connections between secondary and primary zones. The network connections still need to provide sufficient throughput to support replication, and must provide sufficient reliability that temporary interruptions do not interfere with network throughput.

If you deploy your store across multiple zones, then Oracle NoSQL Database tries to physically place at least one Replication Node from each shard in each zone. Whether Oracle NoSQL Database can do this depends on the number of shards in use in your store, the number of zones, the number of Replication Nodes, and the number of physical machines available in each zone. Still, Oracle NoSQL Database makes a best-effort to spread Replication Nodes across available zones. Doing so guards against losing entire shards should the zone become unavailable for any reason.

All of the failover descriptions covered here apply to zones. Failover works across zones in the same way as it does if all nodes are contained within a single zone. Zones offer you the ability for your data to remain available in the event of a large outage. However, read and write capability for any given shard is still gated by whether the remaining zone(s) constitute a majority node partition, and the durability and consistency policies in use for your store activities.

Durability Summary

This document has described how durability guarantees affect a shard's write availability in the event of hardware or network failures. In summary:

- A durability guarantee that requires no acknowledgements from the shard's replicas gives you the best chance that the shard can continue servicing write requests in the event of an outage. However, this durability guarantee can also result in the shard operating with two masters, which leads to data loss once hardware problems are resolved. This is *not* a recommended configuration.
- A durability guarantee requiring a simple majority of primary zone replicas to acknowledge the write operation guards against two masters accidently operating at one time. However, it also means that the shard will be incapable of servicing write requests if more than a majority of the replicas are offline due to a hardware failure.
- A durability guarantee requiring all primary zone replicas to acknowledge the write operation guards against any possibility of data loss. However, it also means that the shard will be unable to service write requests if even one of the replicas is unavailable for any reason.

Consistency Summary

In most cases, replicas can continue to service read requests as long as the underlying hardware remains functional. In its default configuration, there is nothing that stops a replica from doing this, even if it is the only node running after some catastrophic failure.

However, is is possible for a replica to stop servicing read requests following a network failure, if the consistency policy requires either version information, or disallows stale data relative to the master. Whether this happens depends on how your Replication Nodes are exactly partitioned as a result of the failure, and how long it takes to establish a new master. The replica's ability to service read requests is also determined by the consistency policy in use for each request. If the read requires tight consistency with the master, and the master is not available to ensure the consistency can be met, then the read will fail.

Oracle® NoSQL Database Availability and Failover, Release 12.2.4.5
E88250-01