Cloudera Runtime 7.1.4

Apache Hadoop Ozone Overview

Date published: 2020-08-11
Date modified: 2020-10-13

https://docs.cloudera.com/

Legal Notice

Contents

Introduction to Ozone

Apache Ozone is a scalable, redundant, and distributed object store optimized for big data workloads. Apart from scaling to billions of objects of varying sizes, applications that use frameworks like Apache Spark, Apache YARN and Apache Hive work natively on Ozone object store without any modifications.

HDFS works best when most of the files are large but HDFS suffers from small file limitations. Ozone is a distributed key-value object store that can manage both small and large files alike. Ozone natively supports the S3 API and provides a Hadoop-compatible file system interface. Ozone object store is available in a CDP Private Cloud Base deployment.

Ozone storage elements

Ozone consists of the following important storage elements:

- Volumes

  Volumes are similar to accounts. Volumes can be created or deleted only by administrators. An administrator creates a volume for an organization or a team.
- Buckets

  A volume can contain zero or more buckets. Ozone buckets are similar to Amazon S3 buckets. Depending on their requirements, regular users can create buckets in volumes.
- Keys

  Each key is part of a bucket. Keys are unique within a given bucket and are similar to S3 objects. Ozone stores data as keys inside buckets.

When a client writes a key, Ozone stores the associated data on DataNodes in chunks called blocks. Therefore, each key is associated with one or more blocks. Within a DataNode, multiple unrelated blocks can reside in a storage container.

Key features of Ozone

Key features of Ozone are:

- Consistency

  Strong consistency simplifies application design. Ozone object store is designed to provide strict serializability.
- Architectural simplicity

  A simple architecture is easy to use and easy to debug when things go wrong. The architecture of Ozone object store is simple and at the same time scalable. It is designed to store over 100 billion objects in a single cluster.
- Layered architecture

  The layered file system of Ozone object store helps to achieve the scale required for the modern storage systems, . It separates the namespace management from block and node management layer, which allows users to independently scale on both axes.
- Easy recovery

  A key strength of HDFS is that it can effectively recover from catastrophic events like cluster-wide power loss without losing data and without expensive recovery steps. Rack and node losses are relatively minor events. Ozone object store is similarly robust in the face of failures.
- Open source in Apache

  The Apache Open Source community is critical to the success of Ozone object store. All Ozone design and development are being done in the Apache Hadoop community.

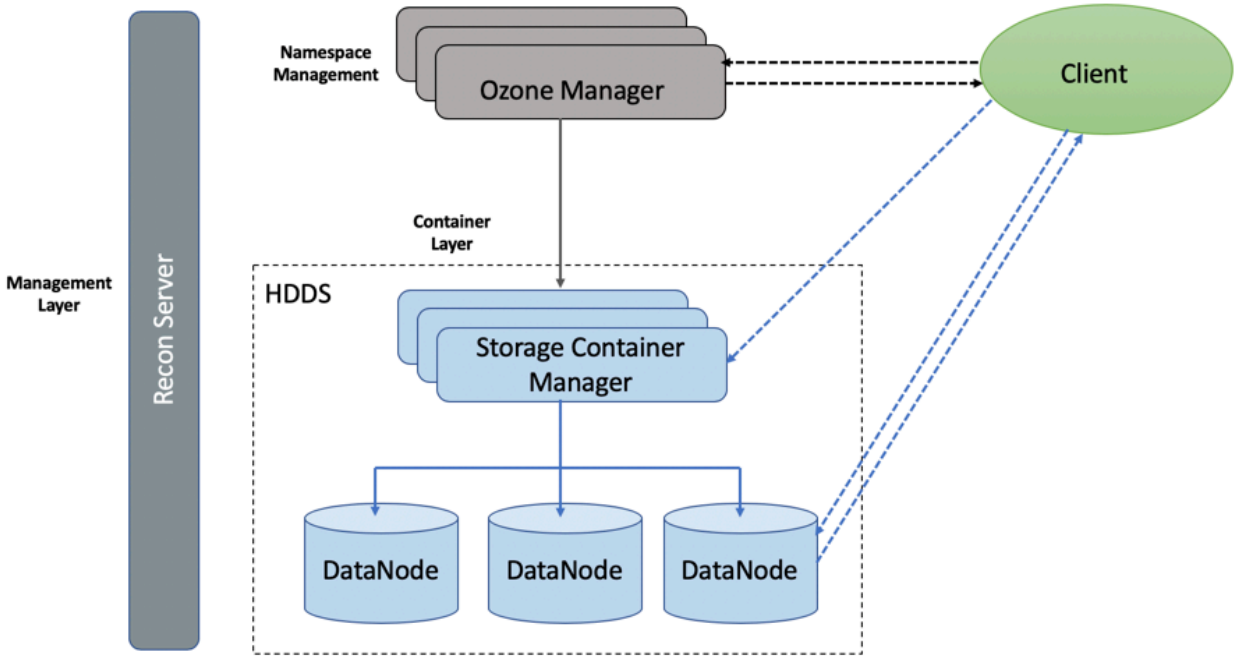
- Interoperability with Hadoop ecosystem

  Ozone object store is usable by the existing Apache Hadoop ecosystem and related applications like Apache Hive, Apache Spark and traditional MapReduce jobs.

Ozone architecture

Ozone can be co-located with HDFS with single security and governance policies for easy data exchange or migration and also offers seamless application portability. Ozone has a scale-out architecture with minimal operational overheads. Ozone separates management of namespaces and storage, helping it to scale effectively. The Ozone Manager (OM) manages the namespaces while the Storage Container Manager (SCM) handles the containers.

The following diagram shows the components that form the basic architecture of Ozone:



Hadoop Distributed Data Store

Ozone is built on a highly available, replicated block storage layer called Hadoop Distributed Data Store (HDDS).

Blocks

Blocks are the basic unit of storage. In Ozone, each block is of 256 MB in size. A collection of blocks forms a storage container. The SCM allocates blocks inside storage containers for the client to store data.

Storage Containers

A storage container is a group of unrelated blocks managed together as a single entity. A container exists in a DataNode and is the basic unit of replication, with a capacity of 2 GB to 16 GB.

DataNodes

DataNodes contain storage containers comprising of data blocks. The SCM monitors DataNodes through heartbeats.

Ozone Manager

The Ozone Manager (OM) is the metadata manager for Ozone. The OM manages the following storage elements:

- The list of volumes for each user

- The list of buckets for each volume
- The list of keys for each bucket

The OM maintains the mappings between keys and their corresponding Block IDs. When a client application requests for keys to perform read and write operations, the OM interacts with the SCM for information about blocks relevant to the read and write operations, and provides this information to the client. In addition, the OM also handles metadata operations from the clients.

Ozone Manager

The Ozone Manager (OM) is a highly available namespace manager for Ozone.

OM manages the metadata for volumes, buckets, and keys. OM maintains the mappings between keys and their corresponding Block IDs. When a client application requests for keys to perform read and write operations, OM interacts with SCM for information about blocks relevant to the read and write operations, and provides this information to the client. In addition, OM also handles metadata operations from the clients.

Storage Container Manager

The Storage Container Manager performs multiple critical functions for an Ozone cluster.

SCM manages the addition and removal of DataNodes, and allocates storage containers and blocks. SCM also manages block collections, ensuring that the blocks maintain the required level of replication. SCM allocates blocks to clients through OM for read and write operations. In addition, SCM executes recovery actions when faced with DataNode or disk failures.

Pipelines

Pipelines determine the replication strategy for the blocks associated with a write operation.

Recon Server

Recon is the management interface for Ozone. Recon provides a unified management API for Ozone.

How Ozone manages read operations

The client requests the block locations corresponding to the key it wants to read. The Ozone Manager (OM) returns the block locations if the client has the required read privileges.

The following steps explain how Ozone manages read operations:

1. The client requests OM for block locations corresponding to the key to read.
2. OM checks the ACLs to confirm whether the client has the required privileges, and returns the block locations and the block token that allows the client to read data from the DataNodes.
3. The client connects to the DataNode associated with the returned Block ID and reads the data blocks.

How Ozone manages write operations

The client requests blocks from the Ozone Manager (OM) to write a key. OM returns the Block ID and the corresponding DataNodes for the client to write data.

The following steps explain how Ozone manages write operations:

1. The client requests blocks from OM to write a key. The request includes the key, the pipeline type, and the replication count.
2. OM finds the blocks that match the request from SCM and returns them to the client.

Note: If security is enabled on the cluster, OM also provides a block token along with the block location to the client. The client uses the block token to connect to the DataNodes and send the command to write chunks of data.

3. The client connects to the DataNodes associated with the returned block information and writes the data.

4. After writing the data, the client updates the block information on OM by sending a commit request.
5. OM records the associated key information.

Note:

- Keys in Ozone are not visible until OM commits the block information associated with the keys. The client is responsible for sending the key-block information to OM after it has written the blocks on the DNs via a commit request.
- If OM fails to commit block information for keys after they have been written, for example, client was unable to send the commit request OM because the write job failed, the keys would not be visible but the data would remain on disk.

  HDDS-4123: OpenKeyCleanup service in Ozone cleans up data on disk whose block information has not been committed on OM. It does by expiring open keys that have not been committed for a configurable time period (7days by default). OM checks for expired keys every 300s by default and issues deletes for these keys. Any data associated with these keys are deleted when the deletes reach DNs. You can contact Cloudera support and request a Hotfix for HDDS-4123 to get this feature.