

Apache Ozone Overview

Date published: 2020-08-11

Date modified: 2021-03-16

CLOUDERA

Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

Contents

Introduction to Ozone.....	4
Ozone architecture.....	5
How Ozone manages read operations.....	7
How Ozone manages write operations.....	7

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.

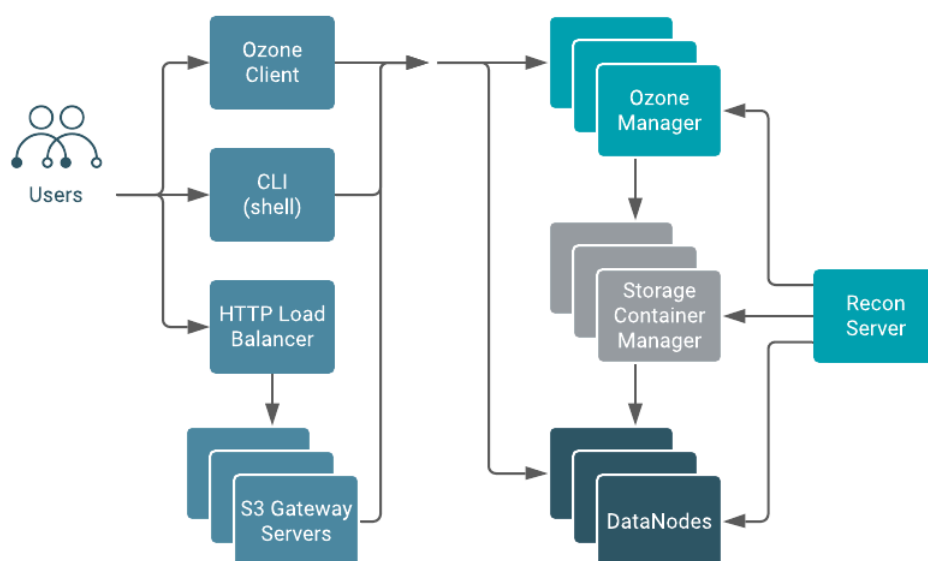
- Interoperability with Hadoop ecosystem

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:



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.

OM uses Apache Ratis (Raft protocol) to replicate Ozone manager state. While RocksDB (embedded storage engine) persists the metadata or key space, the flush of the Ratis transaction to the local disk ensures data durability in Ozone. Therefore, Cloudera recommends a low-latency local storage device like NVMe SSD on each OM node for maximum throughput. A typical Ozone deployment has three OM nodes for redundancy.

DataNodes

Ozone DataNodes, not the same as HDFS DataNodes, store the actual user data in the Ozone cluster. DataNodes store the blocks of data that clients write. A collection of these blocks is a storage container. The client streams data in the form of fixed-size chunk files (4MB) for a block. The chunk files constitute what gets actually written to the disk. The DataNode sends heartbeats to SCM at fixed time intervals. With every heartbeat, the DataNode sends container reports.

Every storage container in the DataNode has its own RocksDB instance which stores the metadata for the blocks and individual chunk files. Every DataNode can be a part of one or more active pipelines. A pipeline of DataNodes is actually a Ratis quorum with an elected leader and follower DataNodes accepting writes. Reads from the client go directly to the DataNode and not over Ratis.

Cloudera recommends an SSD on a DataNode to persist the Ratis logs for the active pipelines and significantly boosts the write throughput.

Storage Container Manager

The Storage Container Manager (SCM) is a master service in Ozone.

A storage container is the replication unit in Ozone. Unlike HDFS, which manages block-level replication, Ozone manages the replication of a collection of storage blocks called storage containers. The default size of a container is 5 GB. SCM manages DataNode pipelines and placement of containers on the pipelines. A pipeline is a collection of DataNodes based on the replication factor. For example, given the default replication factor of three, each pipeline contains three DataNodes. SCM is responsible for creating and managing active write pipelines of DataNodes on which the block allocation happens.

The client directly writes blocks to open containers on the DataNode, and the SCM is not directly on the data path. A container is immutable after it is closed. SCM uses RocksDB to persist the pipeline metadata and the container metadata. The size of this metadata is much smaller compared to the keyspaces managed by OM.

SCM is a highly available component which makes use of Apache Ratis. Cloudera recommends an SSD on SCM nodes for the Ratis write-ahead log and the RocksDB.

A typical Ozone deployment has three SCM nodes for redundancy. SCM service instances can be colocated with OM instances; therefore, you can use the same master nodes for both SCM and OM.

Recon Server

Recon is a centralized monitoring and management service within an Ozone cluster that provides information about the metadata maintained by different Ozone components such as OM and SCM.

Recon takes a snapshot of the OM and SCM metadata while also receiving heartbeats from the Ozone DataNode. Recon asynchronously builds an offline copy of the full state of the cluster in an incremental manner depending on how busy the cluster is. Recon usually trails the OM by a few transactions in terms of updating its snapshot of the OM metadata. Cloudera recommends using an SSD for maintaining the snapshot information because an SSD would help Recon in staying updated with the OM.

Ozone File System Interfaces

Ozone is a multi-protocol storage system with support for the following interfaces:

- ofs: Hadoop-compatible file system allows any application that expects an HDFS like interface to work against Ozone with no changes. Frameworks like Apache Spark, YARN, and Hive work against Ozone without the need for any change.
- s3: Amazon's Simple Storage Service (S3) protocol. You can use S3 clients and S3 SDK-based applications without any modifications to Ozone.
- o3fs: A bucket-rooted Hadoop Compatible file system interface.
- o3: An object store interface that can be used from the Ozone shell.

S3 Gateway

S3 gateway is a stateless component that provides REST access to Ozone over HTTP and supports the AWS-compatible s3 API. S3 gateway supports multipart uploads and encryption zones.

In addition, S3 gateway transforms the s3 API calls over HTTP to rpc calls to other Ozone components.

To scale your S3 access, Cloudera recommends deploying multiple gateways behind a load balancer like haproxy that support Direct Server Return (DSR) so that the load balancer is not on the data path.

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 DN's 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 DN's. You can contact Cloudera support and request a Hotfix for [HDDS-4123](#) to get this feature.