We believe that data can make what is impossible today, possible tomorrow.

We empower people to transform complex data anywhere into actionable insights faster and easier.

We deliver a hybrid data platform with secure data management and portable cloud-native data analytics.
“The future data ecosystem should leverage distributed data management components — which may run on multiple clouds and/or on-premises — but are treated as a cohesive whole with a high degree of automation. Integration, metadata and governance capabilities glue the individual components together.”
CLOUDERA DATA PLATFORM

The only hybrid data platform for modern data architectures with data anywhere

Open data fabrics, lakehouses and data meshes with data anywhere at scale

Multi-cloud & on-premises data management and analytics

“Write once, run anywhere” data analytics portability

Unified security & governance with open cloud-native storage formats
BIG DATA STORAGE REQUIRES ...

Performance
Can it handle **large** workloads

Scale
Can it **Scale** To 100’s PB, 1000’s of nodes and billions of objects

API Compatibility
Does it support **S3 API** and Modern Architecture?
... AND NATIVE INTEGRATION WITH BIG DATA WORKLOADS

Application

Support HDFS and S3 API based applications

Security

Support access control policy, lineage and governance

Encryption

Is the data protected at rest and in-transit?
Apache Ozone

Scalable, redundant distributed object store

Designed for data applications to store structured, unstructured binary data at scale with the capability to read, write and run enterprise applications and workloads at scale as often as possible.
MEETUP AGENDA

- History and Overview *(Sid Wagle)*
- Ozone bucket types *(Ethan Rose)*
  - Legacy, Object Store (OBS), and Filesystem Optimized (FSO)
  - Diverse workloads and when/how to use?
  - Demo
- Ozone Snapshots *(Prashant Pogde, Siyao Meng)*
  - Key differentiator
  - Use cases in a hybrid world
  - Demo
- Ozone Performance *(Ritesh Shukla)*
  - Benchmarks and certifications
  - Optimized for the new DC outlook
A BRIEF HISTORY OF OZONE

Siddharth (Sid) Wagle
PMC, committer (Ozone, Ratis, Hadoop)
HISTORICAL MILESTONES

2015
Jun
HDFS-7240
Hadoop feature branch
First hadoop-ozone commit lands

2018
Apr
HDDS-1
Hadoop sub-project
Hadoop Distributed Data Store

2020
Oct
Ozone-1.0.0
Apache Ozone now a TLP
Apache board establishes Apache Ozone as a top-level project

Initial Goal
Scale HDFS - 2x

New Goal
Storage Containers
Native S3, HCFS
Scale HDFS - 10x

Cloudera GA
THRIVING COMMUNITY
Open source Partner Logos

Google
TARGET
Cisco
Preferred Networks
Microsoft
Tencent
CLOUDERA
Bloomberg
Oracle
HP
Intel
Shopee
APACHE OZONE COMMUNITY
Snapshot - 2022

- Ozone PMC Chair: Sammi Chen
- 28 PMC members, 61 Committers
  - Committers / PMC members located in US, Hungary, India, China, Germany, ...
  - from Cloudera, Tencent, G-Research, Infinstor, Oracle, Microsoft, Intel, Target
- 199 contributors (at least one PR merged), 127 active contributors in the past two years.
- 5000+ commits in total on the main branch, 2100+ merged in the past two years.
APACHE OZONE RELEASES

- Generally Available since 1.0.0 in Sep 2020
- 1.2.1, released in Dec 2021
- Version 1.3.0 is in-progress
  - Tons of new features and improvements
    - Erasure Coding
    - Container Balancer
    - S3 Multi-Tenancy
    - S3 gRPC improvements
  - 1000+ new commits since 1.2.1 release and counting
    - 2,265 changed files with 150,474 additions and 36,212 deletions
**APACHE OZONE - What is it?**

Ozone is a distributed **Key Value Object Store** with native S3 and FS interfaces. Ozone is designed and **optimized** for Big Data workloads. Ozone can **scale** up to **billions of objects** and work effectively in containerized environments like Yarn or Kubernetes. (30x of HDFS)

Ozone is **strongly consistent** and provides the benefits of traditional HDFS and S3 Object Store

Scale to **1000’s of nodes** with dense storage configurations

**Reduce cost** per TB using commodity hardware
KEY CONCEPTS

- Ozone consists of volumes, buckets, and keys.
  - **Volumes** are similar to user accounts or tenants. Only administrators can create or delete volumes.
  - **Buckets** are similar to Amazon S3 buckets. A bucket can contain any number of keys, but buckets cannot contain other buckets.
  - **Keys** are similar to files.
- The hierarchical file system builds on top of the flat key-value store.
BUILDING BLOCKS
Use proven technologies - don’t reinvent the wheel

- **RAFT** replication – [http://raft.github.io](http://raft.github.io)
  - Open source Java implementation of RAFT - Apache Ratis Library.
- **Storage Containers** – Unit of replication (collection of blocks)
  - RocksDB - container metadata
    - Supported by and battle-tested at Facebook.
- **OM** – a namespace manager (also uses RocksDB to store the namespace)
- **HDDS** – a distributed container management layer
- Hadoop security model and Hadoop RPC
OZONE BUCKET TYPES

Ethan Rose
Ozone PMC, committer
OZONE NAMESPACE LAYOUT
/volume/bucket/key

- **Volume**
  - Top level namespace grouping
  - Must have unique names
  - Can only contain buckets
- **Bucket**
  - Must have unique names within the same volume
  - Can only contain keys
- **Key**
  - A file stored in the system
S3 GATEWAY

Allows S3 clients to talk to Ozone

- Stateless server
- Translates S3 REST API calls to Ozone client RPC calls
BEFORE BUCKET LAYOUT TYPES

History

- Ozone was originally built as an object store
- Directories were simulated with prefixes
BEFORE BUCKET LAYOUT TYPES

Problems

• Directory **rename** and **delete** were slow and non-atomic
  – Could cause problems for services expecting Ozone to be a drop-in replacement for HDFS
• Strict S3 compatibility was opt in/out for the whole cluster
  – Is `/volume/bucket/dir1/../../../dir2/key1`
    • The literal name of a key?
    • A path to resolve to `/volume/bucket/dir2/key1`?
EXISTING: LEGACY BUCKETS

- Path normalization handled by config key: `ozone.om.enable.filesystem.paths`
- No directories, only prefixes
- Existing buckets automatically inherit this layout

<table>
<thead>
<tr>
<th>Interface</th>
<th>Legacy buckets accessible?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ozone sh</td>
<td>✓</td>
</tr>
<tr>
<td>ofs (Hadoop compatible)</td>
<td>✓</td>
</tr>
<tr>
<td>S3</td>
<td>✓</td>
</tr>
</tbody>
</table>
NEW: OBJECT STORE BUCKETS (OBS)

- Strict S3 compatibility
  - `/volume/bucket/dir1/..///dir2/key1` is the name of a key
- No directories, only prefixes

<table>
<thead>
<tr>
<th>Interface</th>
<th>OBS buckets accessible?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ozone sh</td>
<td>✓</td>
</tr>
<tr>
<td>ofs (Hadoop compatible)</td>
<td>×</td>
</tr>
<tr>
<td>S3</td>
<td>✓</td>
</tr>
</tbody>
</table>
NEW: FILE SYSTEM OPTIMIZED BUCKETS (FSO)

- Path normalization
  - `/volume/bucket/dir1/../dir2/key` -> `/volume/bucket/dir2/key1`
- Quick, atomic directory operations
  - Rename
  - Delete

<table>
<thead>
<tr>
<th>Interface</th>
<th>FSO buckets accessible?</th>
</tr>
</thead>
<tbody>
<tr>
<td>ozone sh</td>
<td>✓</td>
</tr>
<tr>
<td>ofs (Hadoop compatible)</td>
<td>✓</td>
</tr>
<tr>
<td>S3</td>
<td>✓ (Paths normalized)</td>
</tr>
</tbody>
</table>
FSO BUCKET IMPLEMENTATION

- Implement a hierarchical layout with a key value store (RocksDB)
  - Key: `<Parent object ID>`/`<Object name>`
  - Value: Object metadata, including ID
- Resolve `/engineering/sre/servers/datacenter1/server1.log`
WHICH BUCKET TYPE SHOULD YOU USE?

• OBS:
  - Services built for S3
  - Object store workloads

• FSO:
  - Services built for HDFS
  - Analytic workloads
IMPALA + OZONE

Featuring FSO Buckets
IMPALA + OZONE

- **Impala**: SQL engine built to run in Hadoop clusters
  - Metadata stored in Hive Metastore
  - Data stored in Hadoop compatible storage
- We will store Impala’s data in Ozone instead of HDFS
## IMPALA-9400: IMPALA OZONE SUPPORT

<table>
<thead>
<tr>
<th>Jira</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPALA-10212</td>
<td>ofs support in Impala</td>
</tr>
<tr>
<td>IMPALA-9448</td>
<td>Test coverage for Ozone transparent data encryption</td>
</tr>
<tr>
<td>IMPALA-10213</td>
<td>Support data locality of Impala daemons on Ozone</td>
</tr>
<tr>
<td>IMPALA-10214</td>
<td>Support file handle cache for Ozone</td>
</tr>
</tbody>
</table>
CHOOSING BUCKET TYPE

- Impala has native support for HDFS
- Some Impala operations will have poor performance without fast directory renames and deletes:
  - DROP TABLE/PARTITION
  - INSERT OVERWRITE
  - LOAD IN PATH
- Therefore, we will put Impala’s data in an FSO bucket
DEMO
Bucket types in a live Ozone cluster
Contribute! [https://github.com/apache/ozone](https://github.com/apache/ozone)
OZONE SNAPSHOTS

Prashant Pogde, Siyao Meng
Ozone PMC, committer
SNAPSHOT USE CASES

- Data protection and backup
  - Bucket Granularity
- Replication and DR
  - Stable source image for replication
  - Efficient way to find changes since last replication
  - Near continuous replication
- Compliance
- Easy Rollbacks for Application state
- Malware protection
- Time Travel for data sets
- Incremental analytics
OBJECT VERSIONING VS SNAPSHOTS

- A single object - multiple versions
- A group of objects as a unit
  - Consistent with each other
  - Point in time, together as a group
- Examples:
  - Application updating multiple rows of a table to get to a consistent state
  - Application updating a group of tables to move from one state to another
- Easy rollback for a DB/table to last App consistent view
APP CONSISTENT VIEW EXAMPLE

App Consistent State 1

Table Object 1 - v1
- Row- Object-1 - v5
- Row- Object-2 - v10
- Row- Object-3 - v15

App Inconsistent State

Table Object 1 - v2
- Row- Object-1 - v5
- Row- Object-2 - v8
- Row- Object-3 - v17

App Consistent State 2

Table Object 1 - v3
- Row- Object-1 - v10
- Row- Object-2 - v15
- Row- Object-3 - v20
VALUE DIFFERENTIATORS

- Instantaneous Snapshot creation,
- Immediately available for list/read/restore
- Bucket Snapshots,
  - Extensible to volumes.
  - No nested snapshots
- Basic operations: create/list/delete/restore
- Support out of order deletion
- Snapshot diff between two arbitrary snapshots of the same bucket
- Efficient Snapshot diff mechanism
- Insights: support stats e.g.
  - space locked by snapshots
  - Space freed if a snapshot is deleted
SNAPSHOT DESIGN

Ozone Client

Ozone Manager
{Manage Object Namespace}

Storage Container Manager
{Manage Containers, allocate blocks, certificates, datanodes}

DataNodes
{Store Data Blocks In Containers}

Recon
{observe, metrics, stats, aggregation, repair}

Immutable Data blocks

HA Raft Ring

Snapshotting the object NameSpace is enough to get a point in time image of the object store.
SNAPSHOT DESIGN : KEY IDEAS

● Leverage RocksDB Checkpointing mechanism
  ○ Flush the WAL
  ○ Create Checkpoint

● Instant Snapshots
  ○ Hard links for the SST files
  ○ All Snapshots are self contained with hard links
  ○ In future, it can also be used to support writable Snapshots

● RATIS driven for consistency across all OM HA nodes
  ○ OM nodes issue the checkpoint creation at exactly the same point
ACCESSING SNAPSHOTs

- Snapshots can be accessed through ".snapshot" hidden directory in the namespace
- Only bucket level snapshots are supported currently
- ".snapshot" hidden directory will be available under the bucket path
- For example, key "/volume1/bucket1/k1" from snapshot "snap1" can be read through path "/volume1/bucket1/.snapshot/snap1/k1"
- Snapshots & OM HA
SNAPSHOT DESIGN (Continued ...)

Ozone Manager

Active Object Store Metadata Manager

Is Snapshot Path?

Ozone Metadata Manager

Active Object Store Metadata

Create Snapshot

Active OS RocksDB

Snapshot 0 RocksDB

Snapshot 1 RocksDB

Snapshot n RocksDB

Snapshot Metadata Manager

Snapshot 0 Metadata

Snapshot-1 Metadata

Snapshot-n Metadata

Snapshot Aware Key Deletion Service

Create Snapshot

Snapshot 0 RocksDB

Snapshot 1 RocksDB

Snapshot n RocksDB
SNAPSHOT DIFF

● Given two snapshots of a bucket: Identify changes
● Some of the Use cases
  ○ It can be used for incremental backup
  ○ Incremental Replication for DR
  ○ Efficient virus scans
  ○ Incremental analytics
SNAPSHOT DIFF: MECHANISM

- Namespace walk to identify changes
- Simple but it doesn’t scale
  - Ozone is designed to hold billions of objects
- SnapDiff needs to be an efficient mechanism
- Proportionate to amount of churn in the Object Store
- Other alternatives
  - Maintain Change Log: grows too quickly, latency impact
  - Leverage LSM architecture
SNAPSHOT DIFF : LEVERAGING LSM ARCHITECTURE

- LSM doesn’t do in-place updates
- RocksDB SST files are immutable
- New updates always go to new set of SST files
- In a simple world:
  - Just compare the SST files between two snapshots

Only if it were that simple
SNAPSHOT DIFF: ROCKSDB COMPACTIONS

- RocksDB compaction keeps compacting existing SST files into new files.
SNAPSHOT DIFF : OVERALL PICTURE

Active ObjectStore

007.sst
008.sst
009.sst

Snapshot 1

Active ObjectStore

000.sst
008.sst
002.sst

Snapshot 2

Active ObjectStore

010.sst
011.sst
012.sst
013.sst
SNAPSHOT DIFF: COMPACTIATION DAG

SnapDiff (Snapshot1, Snapshot2) = \{12.sst\}
Note that \{007.sst, 008.sst, 009.sst\} is same as \{010.sst, 011.sst\}
SNAPSHOT DIFF (Continued…)

- Rename Handling?
  - Use unique Object IDs to track renames
- Store the Computed Snapdiffs
- Compute Snapdiffs using existing Snapdiffs
  - Diff(snap1, snap3) = Merge{Diff(snap1, snap2), Diff(snap2, snap3)}
- SnapDiffs can be served from any OM node including the follower nodes
- Snapshot reads can also be directed to OM follower nodes

“Only use published RocksDB APIs”
SNAPSHOT DIFF : Overall Picture

Ozone Manager : Generate Compaction DAG

- SnapDiff Handler
  - Store & Manage SnapDiffs
  - Store SnapDiffs

- Find SST files that are different
- KeyInfo based diff for the target key set
- Filter SST that are not in bucket scope

In-Memory Compaction DAG

Persisted Compaction DAG
SNAPSHOTS : GARBAGE COLLECTION

- Updated Key Deletion Service
- Key Deletion from Active Object store
  - Just check previous snapshot before reclamation
- Snapshot Deletion
  - Check both previous and next snapshot for key claimation
  - No need to check the entire snapshot chain
SNAPSHOTS : DEMO

● Creating Snapshots
● Listing Snapshots
● Read from Snapshots
● Read from Snapshots after updates to active ObjectStore
● Snapdiff
OZONE PERFORMANCE

Ritesh Shukla
Ozone committer
SEPARATION OF CONCERNS

- Foreground
  - Ozone Client

- Background
  - Recon

- Scale Out
  - Datanode
  - Storage Container Manager

- No foreground load
- No background load

- hadoop-ozone
- hadoop-hdds
AGGREGATION VIA CONTAINERS

User

Chunks

Blocks

Containers

Volumes

Datanodes

Storage Container Manager

Unit of Recovery

Unit of Failure
OZONE SCALES!

Number of Objects

Key Creation Rate
OZONE SCALES!
WHY DOES FOREGROUND SCALE

Ozone Manager is designed to optimize application load and object count

- **No heap limitations**: Working set can be cached in memory and unused data can be destaged to disk via RocksDB
- **No block report load**: Background processing is separated from foreground.
- **NVME**: OM uses NVME to store RocksDBs
- **Simplicity**: Written with simplicity in mind
  - Example: Snapshots leverage RocksDB to preserve simplicity of IO path.
Ozone is designed to optimize node density and node count

- **Container abstraction**: Container count scales only with capacity and not object count.
  - Node count scales better than HDFS.
- **Storage Container Manager**: Dedicated service to track, recover and rebalance containers.
- **Higher density**:
  - Datanodes certified against ~0.5 Petabyte capacity nodes
    - Cisco UCS M6: 256 TB per datanode
    - Cisco UCS S3260: 384 TB per datanode
  - Datanodes simulated against 1 Petabyte capacity nodes (200k containers)
## OZONE VS. HDFS

<table>
<thead>
<tr>
<th>Capability</th>
<th>Ozone</th>
<th>HDFS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage Density</td>
<td>1000’s of nodes at 600TB per node</td>
<td>1000’s of nodes at 100TB per node</td>
</tr>
<tr>
<td>Scalability</td>
<td>10B Objects</td>
<td>400M Objects</td>
</tr>
<tr>
<td>Recovery</td>
<td>Fast recovery</td>
<td>Slow startup based on size</td>
</tr>
<tr>
<td>High Availability</td>
<td>Active - Active</td>
<td>Active - Standby</td>
</tr>
<tr>
<td>Protocol Support</td>
<td>Hadoop / S3 API</td>
<td>Hadoop API</td>
</tr>
</tbody>
</table>
SMALL OBJECTS... WELCOME!

PUT/GET Throughput 8 Datanodes 8 Clients 20 Threads

PUT
GET
HARDWARE TRENDS
All NVME clusters are increasingly common

- Ozone’s metadata is stored on SATA SSD or NVME
- Increasing number of customers using all NVME clusters (metadata and data)
  - High density nodes with Ozone
  - High performance workloads
  - Effectively lower TCO for all NVME clusters.
IMPALA + OZONE

- Data warehouse is popular use case for Ozone customers
- Cloudera is investing in optimizing Impala + Ozone stack
Ozone is close to HDFS in performance

**SCR:** Short circuit read

**Remote:** replica preference = remote
Ozone is close to HDFS in performance

SCR: Short circuit read
Remote: replica preference = remote
INVESTING INTO PERFORMANCE
Upcoming releases are performance focused

• Datanode - saturating the network
  – RATIS streaming
    • Efficient data path with rack awareness
    • Zero copy buffers
  – Simplified IO path for erasure coding
• OM - operations per second
  – Concurrency improvements
  – Caching background updates
  – Reducing latency per operation
• Impala improvements against Ozone
RATIS STREAMING - MULTIPLE CLIENTS

Multiple clients can achieve 3x performance over current implementation
SUMMARY

- Ozone architecture solves big data scale issues
- Ozone is cost effective and meets performance
- Cloudera is continuing to push performance across the stack for Ozone
- Hardware trends well suited to leverage Ozone’s capabilities.
- Performance tests validate architecture and direction for Ozone.
CONTRIBUTIONS WELCOME!

Questions?
Flamegraph (Ozone DataNode)
Tests conducted

• Freon read load post hard restart (minimal caching)

• Warp test to measure network saturation when using S3

• Impala TPCDS benchmark

• Ratis streaming performance tests
Software under test

CDP Private Cloud Base 7.1.8 +

- IMPALA-11457 Fix regression with unknown disk id
- HDDS-4970 Significant overhead when DataNode is over-subscribed
- HDDS-7135 ofs file input stream should support StreamCapabilities interface
- HDDS-7136 Memory leak due to ChunkInputStream.close() not releasing buffer
- HDDS-7161 Make Checksum.int2ByteString() zero-copy

All fixes are upstreamed in Apache Ozone 1.3.0 + Apache Impala 4.1.1
Software under test

CDP Private Cloud Base 7.1.8 +

- IMPALA-11457 Fix regression with unknown disk id
- HDDS-4970 Significant overhead when DataNode is over-subscribed
- HDDS-7135 ofs file input stream should support StreamCapabilities interface
- HDDS-7136 Memory leak due to ChunkInputStream.close() not releasing buffer
- HDDS-7161 Make Checksum.int2ByteString() zero-copy

All fixes are upstreamed in Apache Ozone 1.3.0 + Apache Impala 4.1.1
# Testbed

3 x master nodes, 16 x DataNodes

## Master nodes

<table>
<thead>
<tr>
<th>CPU</th>
<th>2 x Intel(R) Xeon(R) Gold 6230 CPU @ 2.10GHz/20 cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory</td>
<td>384GB ( 12 x 32GB DDR4 @ 2933MHz)</td>
</tr>
<tr>
<td>OS Boot</td>
<td>Cisco Boot optimized M.2 Raid controller with 2 x 240GB SATA SSD</td>
</tr>
<tr>
<td>SSD</td>
<td>3.8TB SATA SSD Enterprise Value</td>
</tr>
<tr>
<td>Storage Controller</td>
<td>Cisco 12G Modular Raid Controller with 2GB cache</td>
</tr>
<tr>
<td>Network Adapter</td>
<td>Cisco UCS VIC 1387 2 x 40Gbps ports x8 PCIe Gen3</td>
</tr>
</tbody>
</table>

## Data Nodes

<table>
<thead>
<tr>
<th>CPU</th>
<th>2 x Intel(R) Xeon(R) Gold 6262V CPU @ 1.90GHz/24 cores</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory</td>
<td>384GB ( 12 x 32GB DDR4 @ 2933MHz)</td>
</tr>
<tr>
<td>OS Boot</td>
<td>Cisco Boot optimized M.2 Raid controller with 2 x 240GB SATA SSD</td>
</tr>
<tr>
<td>NVMe</td>
<td>10 x 8TB Intel P4510 U.2 High Performance Value</td>
</tr>
<tr>
<td>Storage Controller</td>
<td>NA</td>
</tr>
<tr>
<td>Network Adapter</td>
<td>Cisco UCS VIC 1387 2 x 40Gbps ports x8 PCIe Gen3</td>
</tr>
</tbody>
</table>