Apache Ozone - Balancing and Deleting Data At Scale

Lokesh Jain
Software Engineer, Cloudera
About Me

• Senior Software Engineer, Cloudera
• PMC and committer for Apache Ozone, Apache Ratis and Apache Hadoop
• Contributing for past 4+ years
• Introduction
• Architecture
• Deletion
• Balancing
Ozone

- Distributed Object Store – Volumes, Buckets, Keys
- Object Store, Filesystem and S3 API
- Started as sub project in Hadoop, currently a top level project in Apache
• Introduction
• Architecture
• Deletion
• Balancing
Understanding the Hadoop FS Application API

Hadoop Compatible FS API

*FileSystem or FileContext*

BigData Apps – MR, Hive, Spark, ...

Non-Big data Apps

Note: S3 is accessed via Hadoop FS API using S3 connector

- HDFS Connector
- Ozone Connector
- S3 Connector
- GCS Connector
- ADLS Connector
- Ceph Connector

Ozone

HDFS

Scalable FS with Hierarchical Name space

S3

GCS

ADLS

Ceph

S3 API

S3 API

S3 API

S3 API

S3 API

S3 API

S3 API
HDFS & Ozone – Can Share Storage Servers and Physical Storage

Data Nodes: *Shared* Storage Servers for **HDFS-Blocks** and **Ozone/Quadra Blocks**

- **HDFS**: Scalable FS with Hierarchical Name space
- **Ozone**: Highly Scalable KV Object Store Flat++ Namespace

**Hadoop Compatible FS API**
*FileSystem or FileContext*

**S3 API**
How it all Fits Together

File (Object) = Bid[...]
Bid = Cid+ LocalId

Ozone Manager
K-V Flat
Namespace
Working Set in RAM

SCM
Container Management & Cluster Membership

ContainerMap
(CId -> IPAddress of DN)

Recon Server

HDDS
HDDS – Clean Separation of Block layer

HDDS Container Storage on DataNodes
(Bid -> Data, but blocks grouped in containers)

Container Reports

DataNodes and physical storage shared between Old HDFS and HDDS

Storage & IO

Namespace

DataNodes

Physical Storage - Shared

Old HDFS and HDDS
Ozone Write a Key

1. Client → PutKey → Ozone Manager → Allocate Block /Container/Pipeline → SCM

2. Client
   - Writes data as chunks
   - Update metadata of a block

3. Client → Commit Key → Ozone Manager

<table>
<thead>
<tr>
<th>Block</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 001</td>
<td>{ List of Chunks }</td>
</tr>
<tr>
<td>Block 002</td>
<td>{ List of Chunks }</td>
</tr>
</tbody>
</table>
Ozone Read a Key

1. Client \[\xrightarrow{\text{GetKey}}\] Ozone Manager
   - KeyLocationInfos

2. Client \[\xrightarrow{\text{Read data blocks as chunks}}\] Datanode

<table>
<thead>
<tr>
<th>Block</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block 001</td>
<td>{ List of Chunks }</td>
</tr>
<tr>
<td>Block 002</td>
<td>{ List of Chunks }</td>
</tr>
</tbody>
</table>
Details of the Namespace Layer
High Level Concepts & API

- **Name (Key):** `/Volume/bucket/dir1/dir2/`
- **Volumes** - Unit of management, admin
  - E.g. /home, /users, /tmp, /data-sales, /data-marketing
- **Ozone is Consistent**

- **Two APIs:**
  - Hadoop File system API
  - S3 API

---

- **BigData Apps**
  - MR, Hive, Spark

- **Non-Big data Apps**

- **Hadoop FileSystem & Hadoop FileContext Connectors**

- **S3 Gateway**

- **Ozone Object API (RPC)**
HDDS – The Storage layer
Key High-Level Concepts

**Container: set of blocks (5GB)**
- Replicated as a group (using Raft)
- Each Container has a unique ContainerId
  - Every block within a container has a local id
    - BlockId = ContainerId, LocalId

**Data Nodes – HDFS & HDSS can share DNs**
- DNs contain a set of containers
  - just like DNs used to contain blocks
- DNs send Container-reports to SCM
  - like block reports

**SCM – Storage Container manager**
- Cluster membership
- Receives container reports from DNs
- Manages container replication
- Maintained Container Map (Cid->IPAddr)
Structure of a Storage Container
• Introduction
• Architecture
• Deletion
• Balancing
Block deletion

Client → OM

OM

→ SCM

Periodic Batch deletion

Key Deletion Service

Heartbeat

Block Deletion Service

Block deletion request

DN
Block deletion

OM

Key Deletion Service

Periodic Batch deletion

List of blocks to delete

SCM

- Create delete transactions
- Transaction is list of blocks to be deleted for a specific container
Block deletion

Delete Transactions

- b11, b12, b13, b14, b21, b22, b23
- T1 - b11, b12, b13, b14
- T2 - b21, b22, b23
- TransactionId is monotonically increasing
Block deletion

SCM Block Deleting Service

- Sends delete transactions to datanodes with the respective container
- Deletes transaction once all datanodes ACK

Heartbeat

Delete transactions

Datanode

- Stores delete transaction and sends ACK to SCM

Background Block Deletion Service
## vs HDFS

<table>
<thead>
<tr>
<th>HDFS</th>
<th>Ozone</th>
</tr>
</thead>
<tbody>
<tr>
<td>• HB = 3 secs</td>
<td>• HB = 30 secs</td>
</tr>
<tr>
<td>• Default 20000 blocks deletion per min</td>
<td>• Default 20000 blocks deletion per min</td>
</tr>
<tr>
<td>• Synchronous deletion by datanode</td>
<td>• Asynchronous deletion</td>
</tr>
<tr>
<td></td>
<td>• Reduces to a flow problem</td>
</tr>
</tbody>
</table>
• Introduction
• Architecture
• Deletion
• Balancing
Container Balancer
Container Balancer

Iterate:
- Identify over and under utilised datanodes
- Identify source and target datanodes based on selection criteria
- Identify containers to be moved from source to target
- Issue move requests to Replication Manager

Replication Manager
- Replicates container to target
- On successful replication delete container in source datanode
Container Balancer

• Stateless Service
• Interface driven design
  • Interface to get DN reports with capacity usage of dns
  • Interface to get container and replica information for over-utilized datanodes
  • Selection criteria for containers to balance
  • Selection criteria for target dns which should receive the selected containers
  • Interface to move the selected containers
• Can be extended to balance hot/cold data in cluster
Container Balancer

Limits/Throttling

• Maximum size moved from/to datanode
• Maximum size moved by balancer per iteration
• Percentage of total datanodes involved in balancing
• Limit bandwidth used for container move
Container Balancer

Selection Criteria for target datanodes
• Container obeys placement policy after replication
• Should not already contain the container
• User provided datanode list
• Priority of replication > balancing
Selection Criteria for containers

• Containers should not be undergoing replication
• Better to move containers not following placement policy
• Move larger containers if possible
• User provided exclude and include list
Email -

ljain@apache.org
Thank You
Please take a moment to rate this session.

Your feedback is important to us.