STORAGE DEVELOPER CONFERENCE

SDC 21

BY Developers FOR Developers

Virtual Conference
September 28-29, 2021

A SNIA Event

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

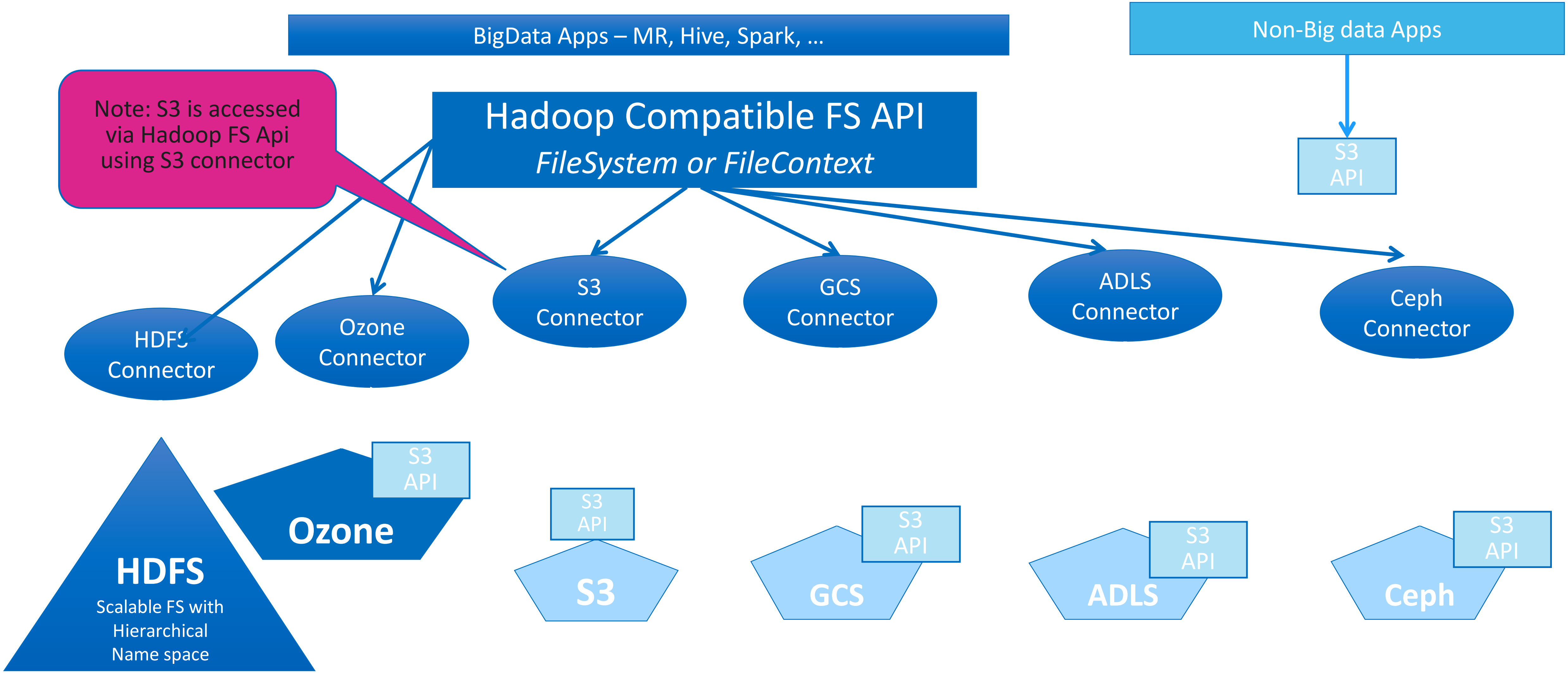
- Introduction
- Architecture
- Deletion
- Balancing

Ozone

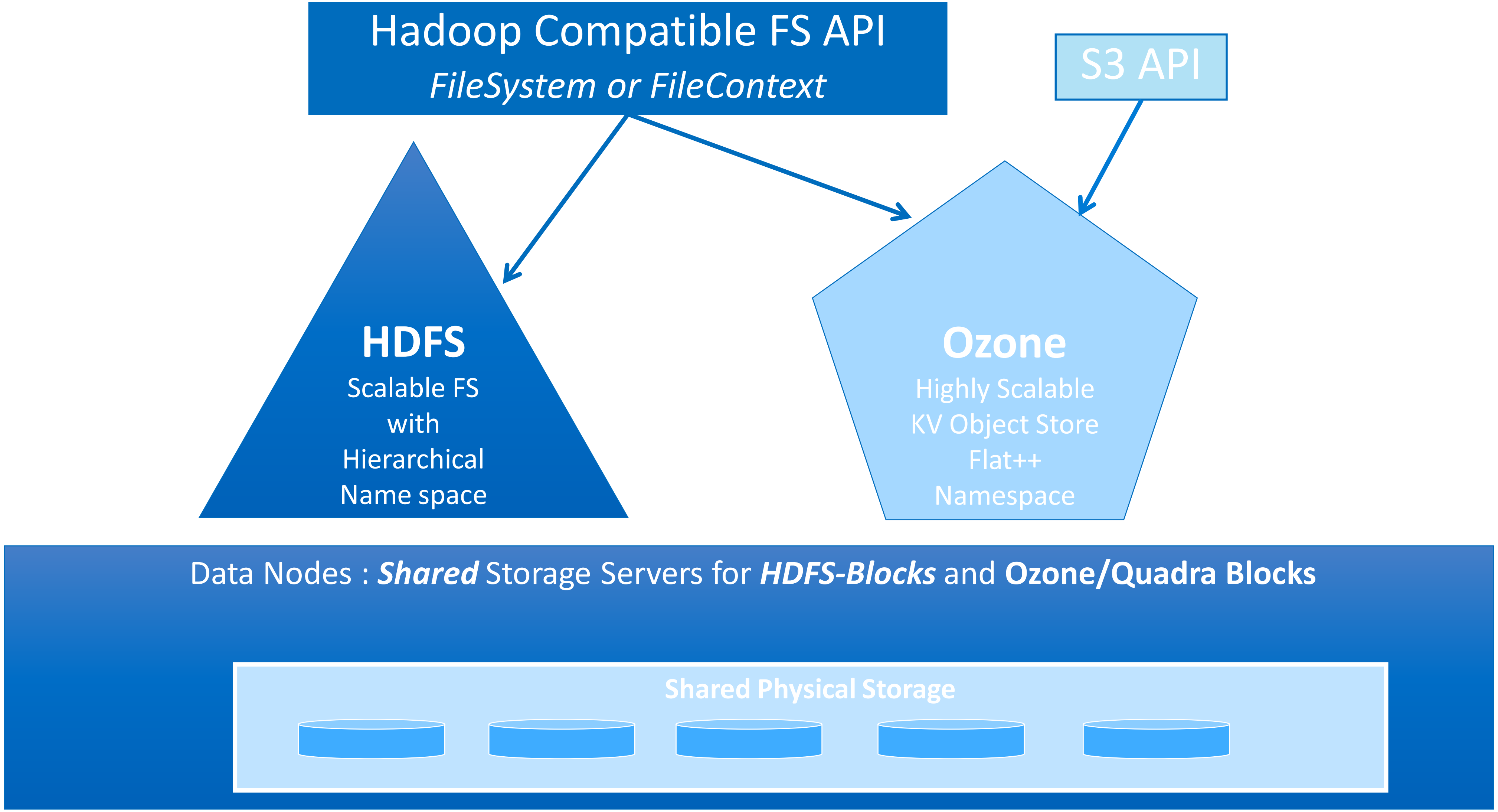
- Distributed Object Store – Volumes, Buckets, Keys
- Object Store, Filesystem and S3 API
- Started as sub project in Hadoop, currently a top level project in Apache

- Introduction
- Architecture
- Deletion
- Balancing

Understanding the Hadoop FS Application API



HDFS & Ozone – Can Share Storage Servers and Physical Storage


Hadoop Compatible FS API

FileSystem or FileContext

S3 API

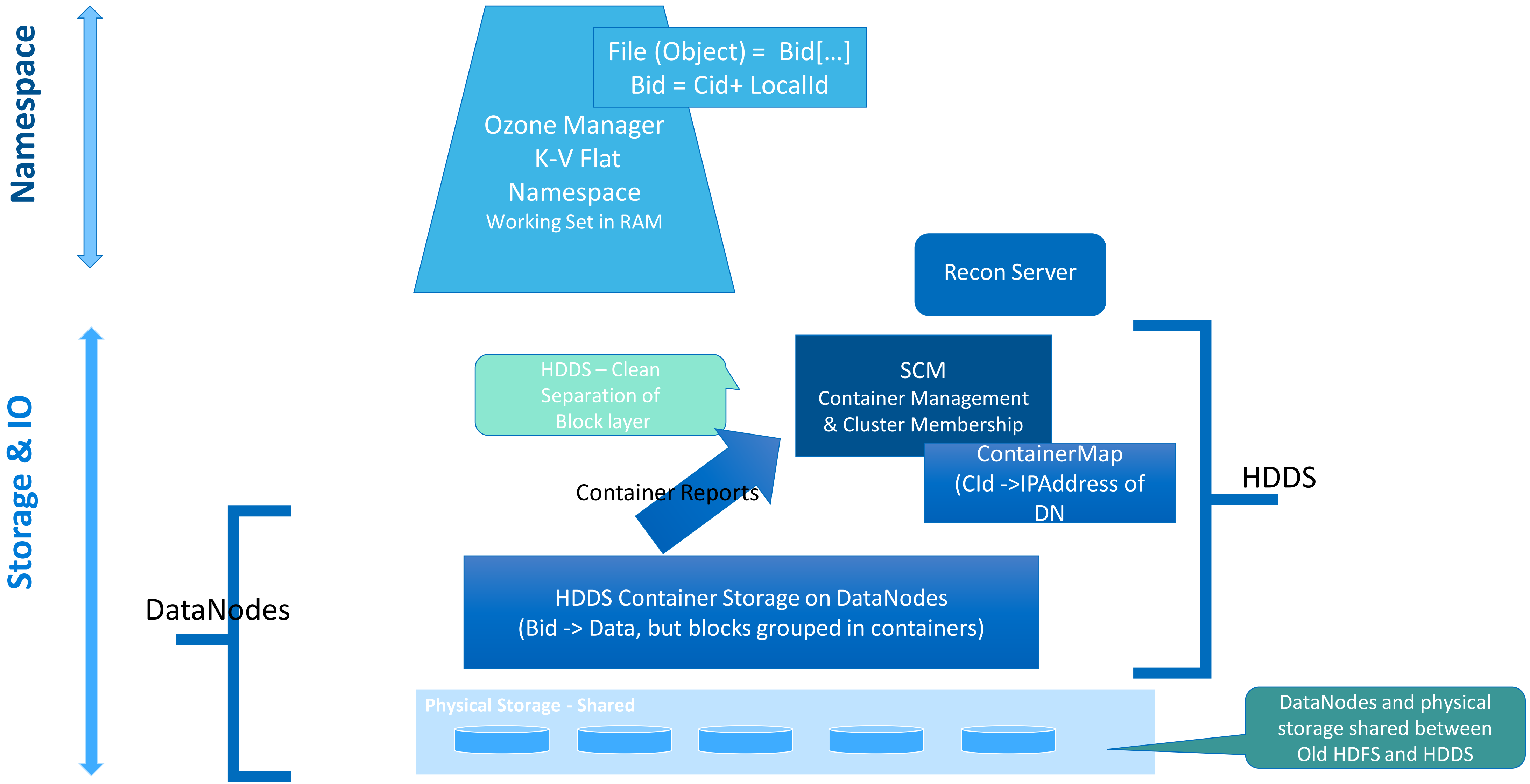
HDFS
Scalable FS
with
Hierarchical
Name space

Ozone
Highly Scalable
KV Object Store
Flat++
Namespace

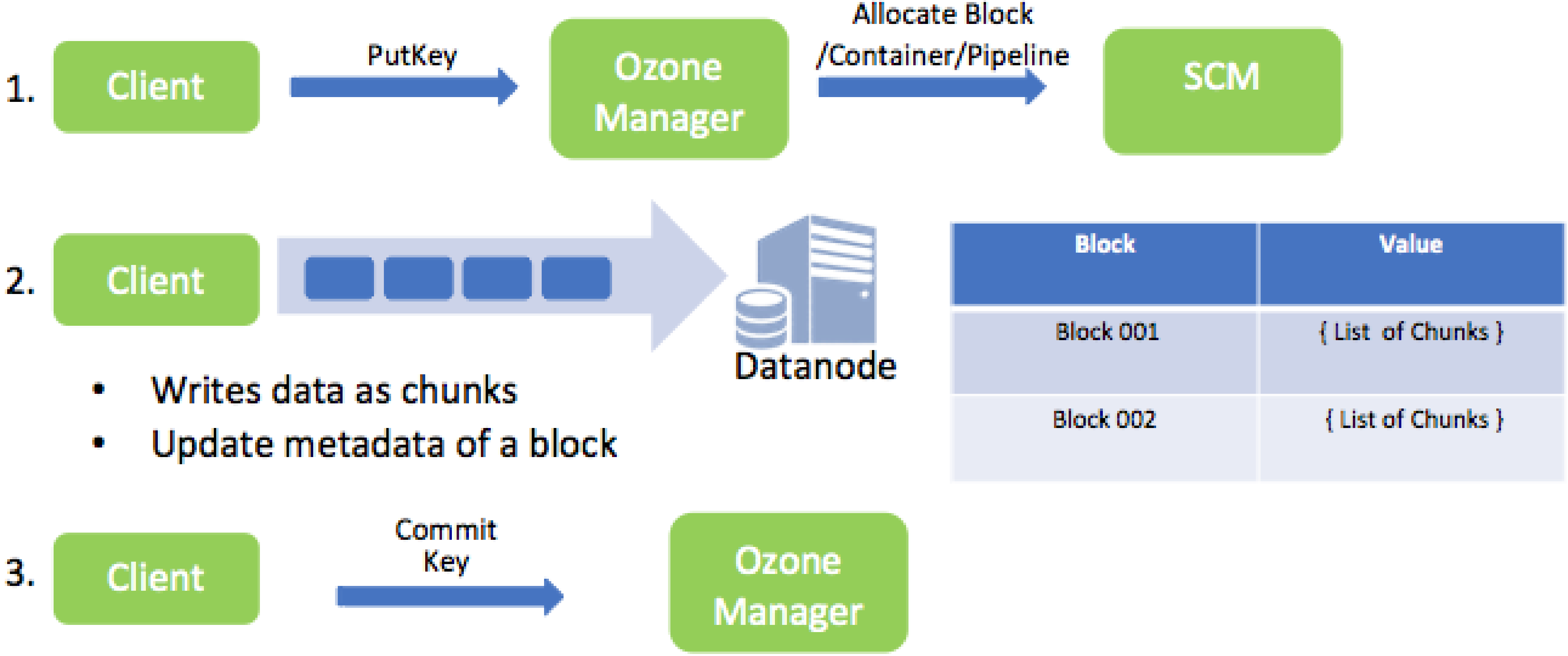
Data Nodes : ***Shared*** Storage Servers for ***HDFS-Blocks*** and **Ozone/Quadra Blocks**

Shared Physical Storage

How it all Fits Together



Ozone Write a Key



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

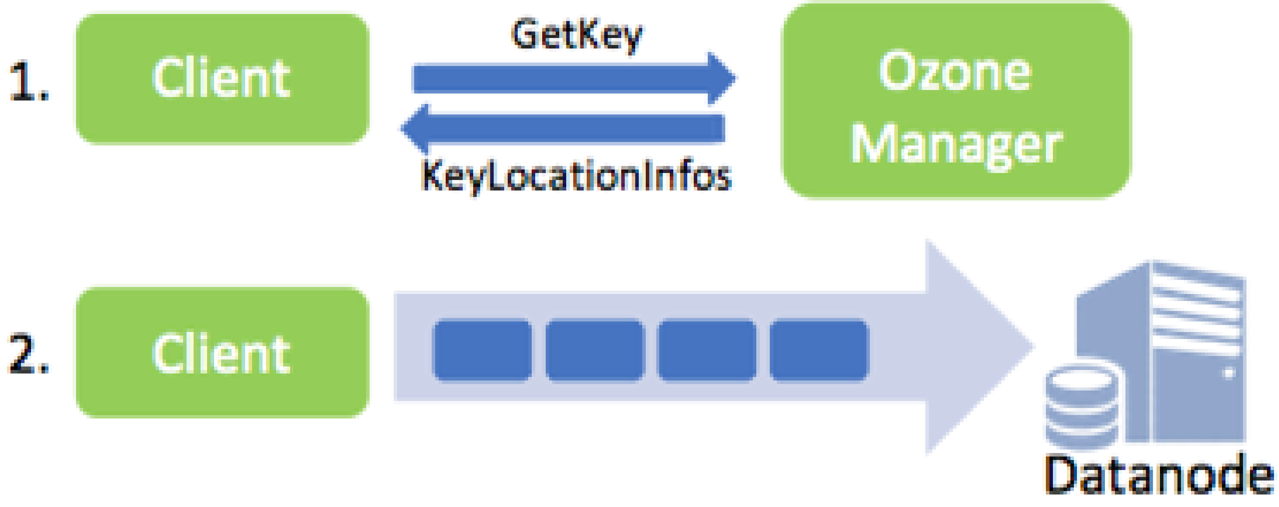
2. Client → Datanode

- Writes data as chunks
- Update metadata of a block

Block	Value
Block 001	{ List of Chunks }
Block 002	{ List of Chunks }

3. Client → Commit Key → Ozone Manager

Ozone Read a Key



- Read data blocks as chunks

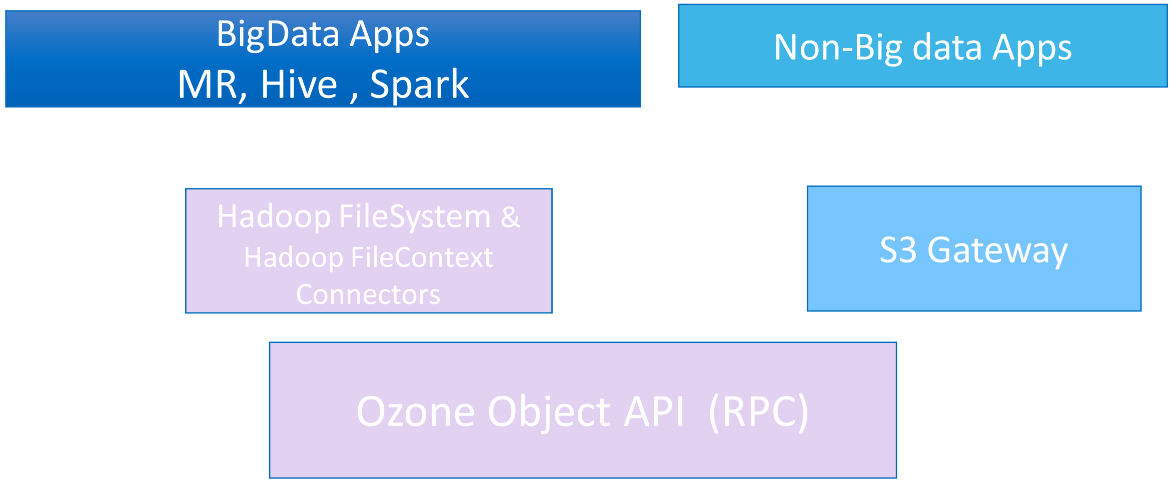
Block	Value
Block 001	{ List of Chunks }
Block 002	{ List of Chunks }

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



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

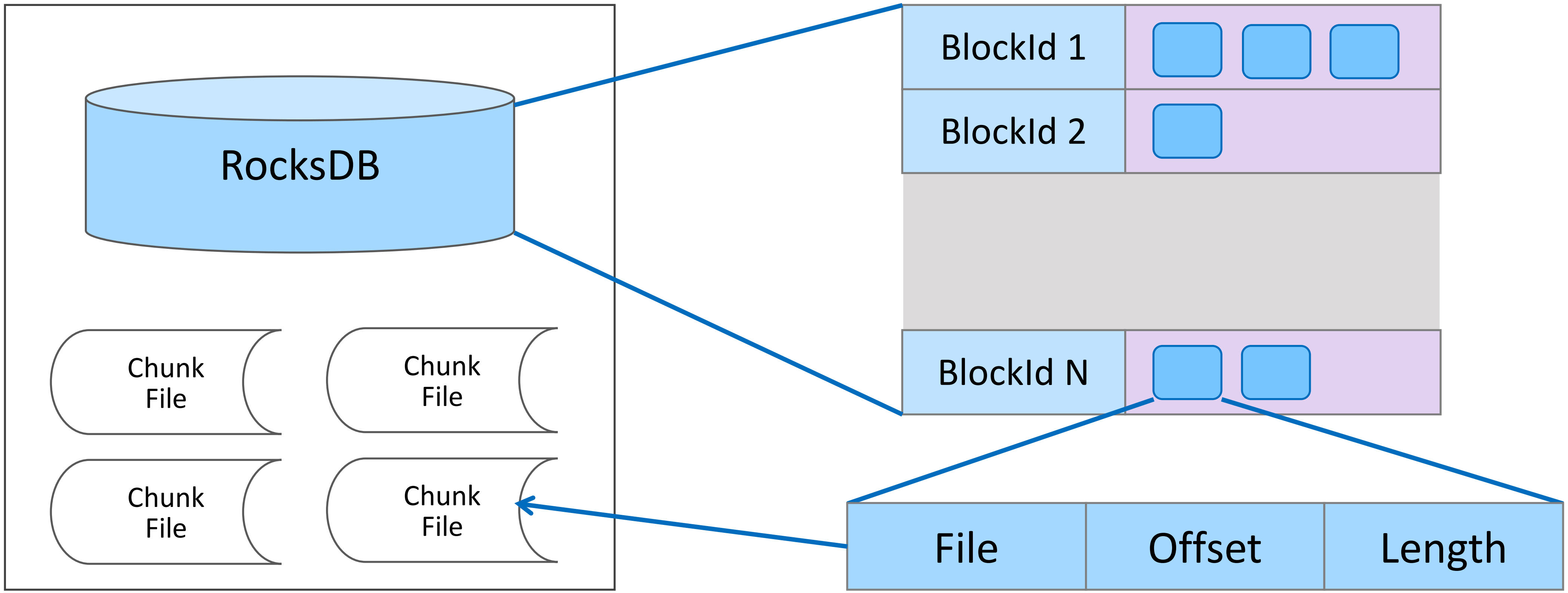
SCM – Storage Container manager

- Cluster membership
- Receives container reports from DNs
- Manages container replication
- Maintained Container Map (Cid->IPAddr)

Data Nodes – HDFS & HDDS can share DNs

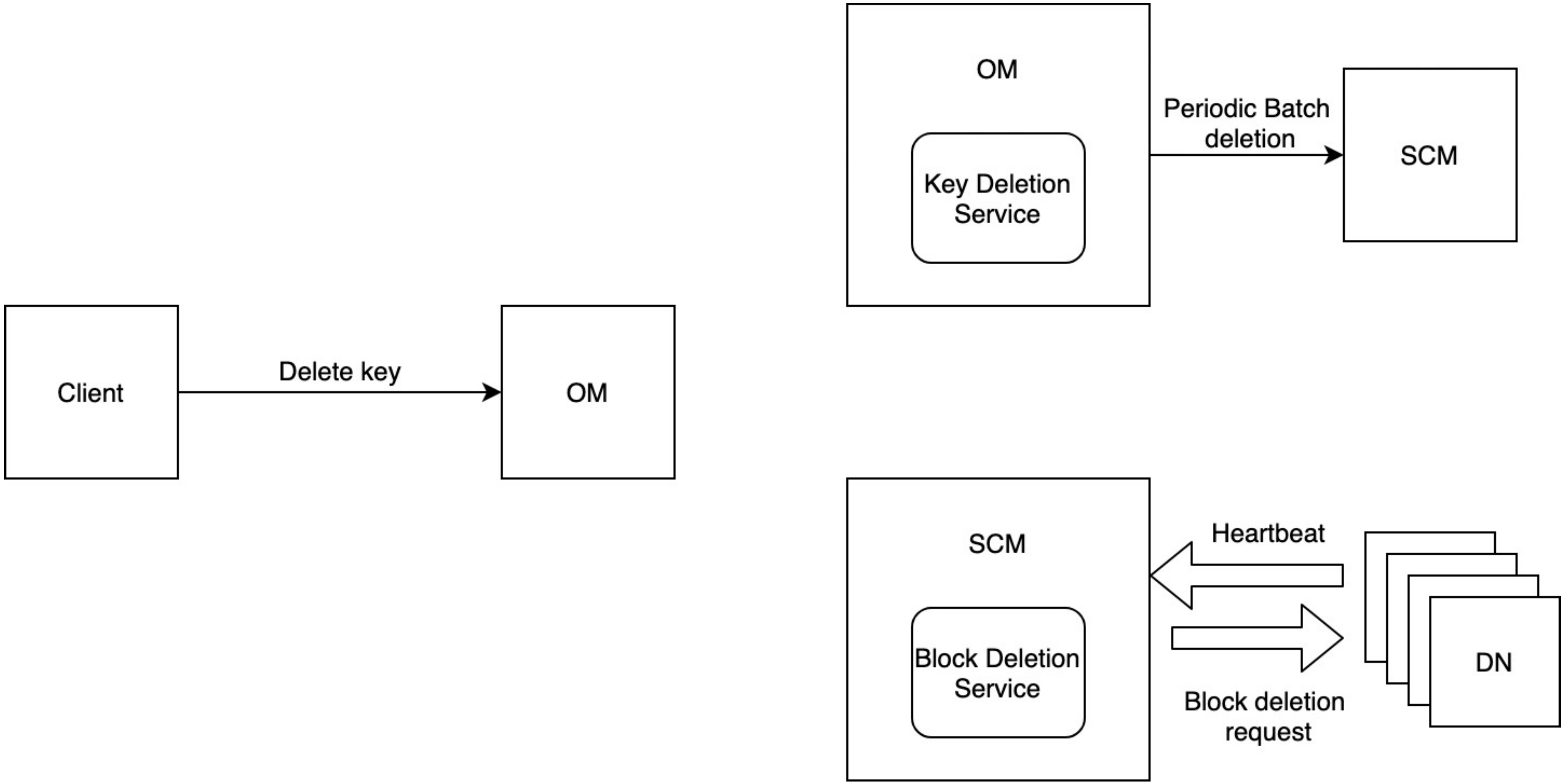
- DNs contain a set of containers
  - just like DNs used to contain blocks
- DNs send Container-reports to SCM
  - like block reports

Structure of a Storage Container



- Introduction
- Architecture
- Deletion
- Balancing

Block deletion



Client

Delete key

OM

OM

Key Deletion Service

Periodic Batch deletion

SCM

SCM

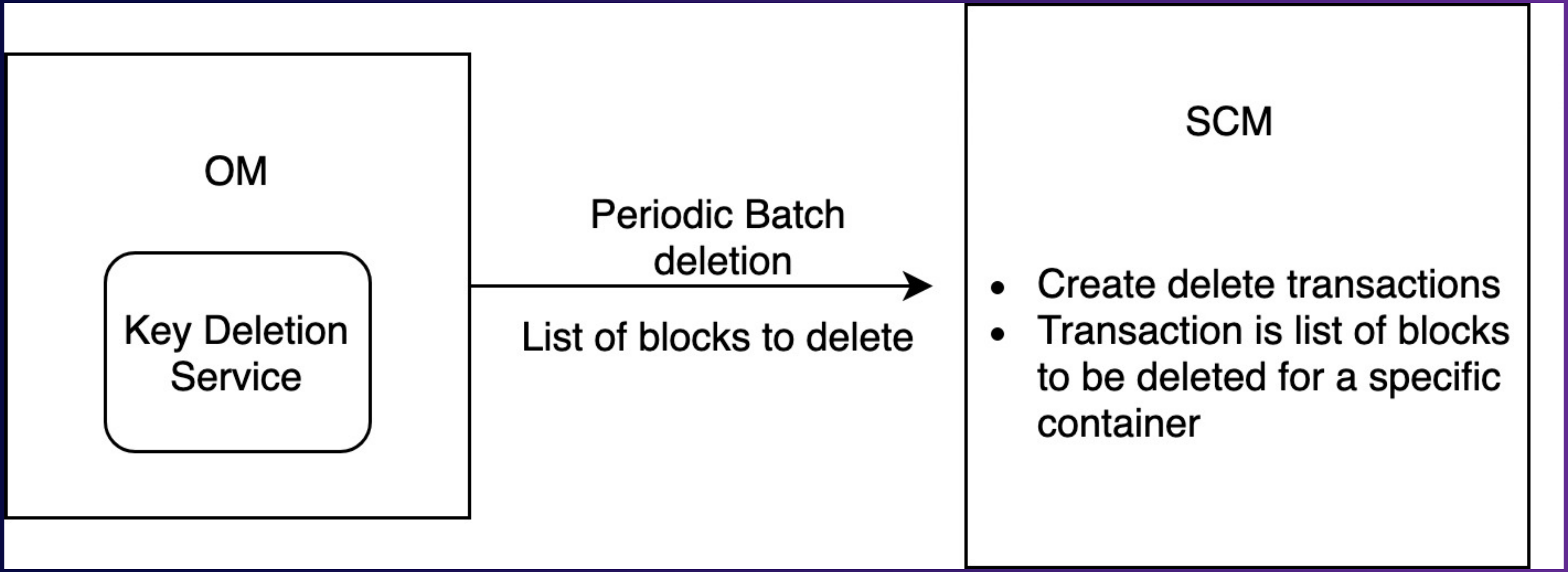
Block Deletion Service

Heartbeat

Block deletion request

DN

Block deletion



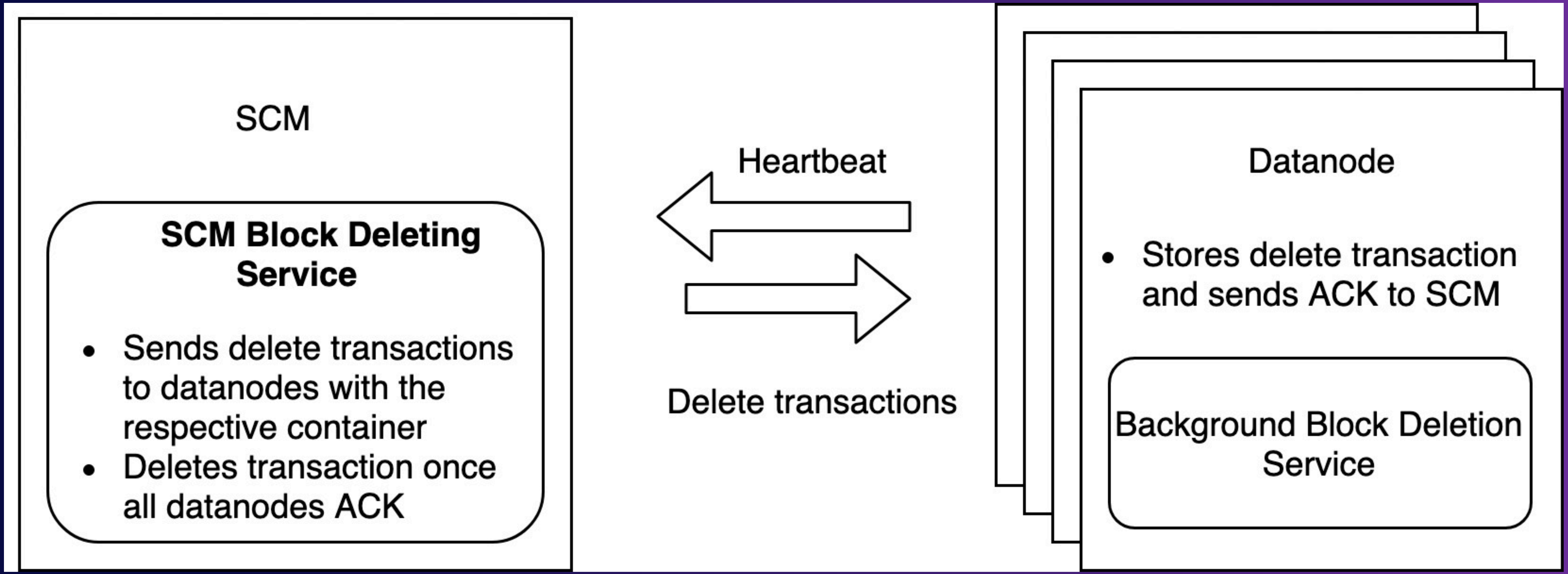
- Create delete transactions
- Transaction is list of blocks to be deleted for a specific container

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

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

HDFS

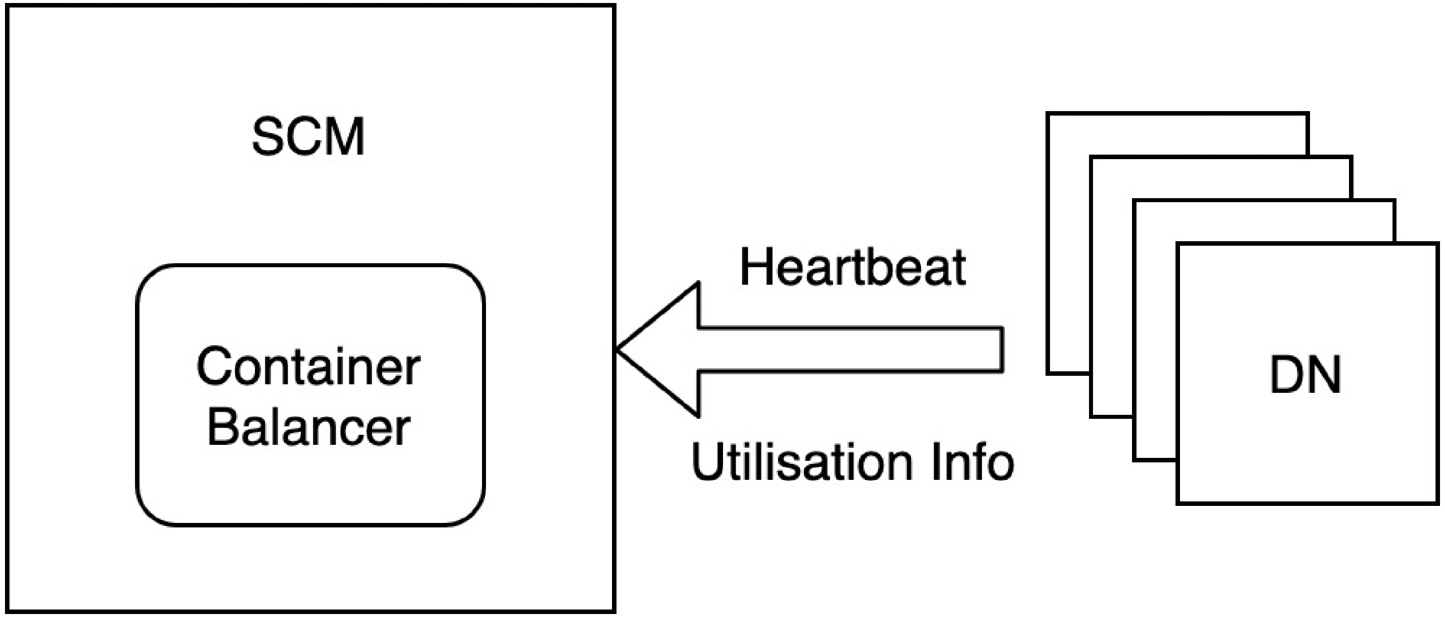
- HB = 3 secs
- Default 20000 blocks deletion per min
- Synchronous deletion by datanode

Ozone

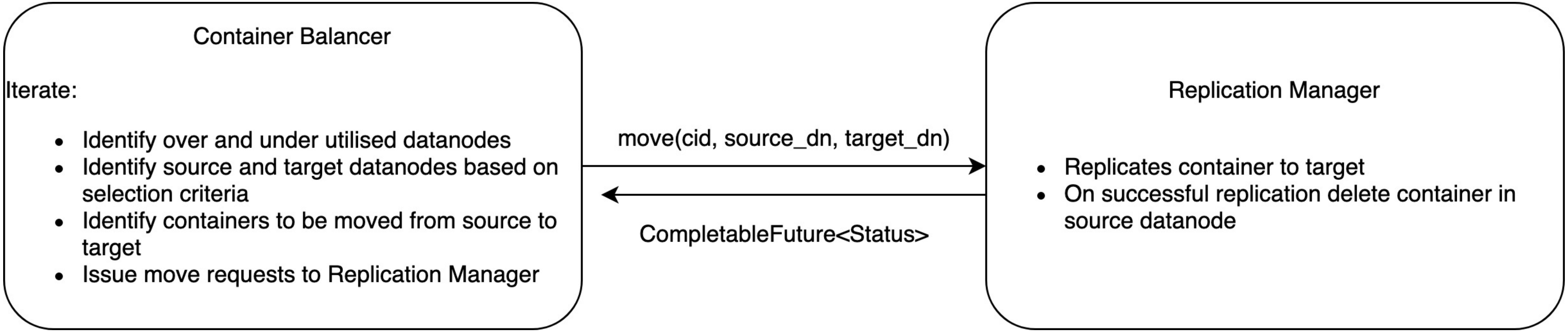
- HB = 30 secs
- Default 20000 blocks deletion per min
- Asynchronous deletion
- Reduces to a flow problem

- Introduction
- Architecture
- Deletion
- Balancing

Container Balancer



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

move(cid, source_dn, target_dn)

CompletableFuture<Status>

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

Container Balancer

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.