Hadoop Storage Reloaded

5 lessons Ozone learned from HDFS

aws S3 protocol

Hadoop FS

CSI

Apache Hadoop Ozone

hadoop.apache.org/ozone

THE APACHE SOFTWARE FOUNDATION

Márton Elek

- Hadoop PMC, Ratis PMC
- Cloudera
  - Principal software engineer
- twitter.com/@anzix
- Kubernetes + Apache Big-data:
  - **github.com/elek**/flekszible
  - flokkr.github.io

Hadoop storage

Why?

- First release of Hadoop: 2006
- 2020: We are living in a **different** word
- Streaming? Cloud?
  → small files
- Machine Learning? Cloud-native tools?
  → Hadoop Compatible File System API may not enough

Hadoop Storage

HDFS

Hadoop Storage

HDFS

HCFS cloud connectors

Hadoop Storage

HDFS

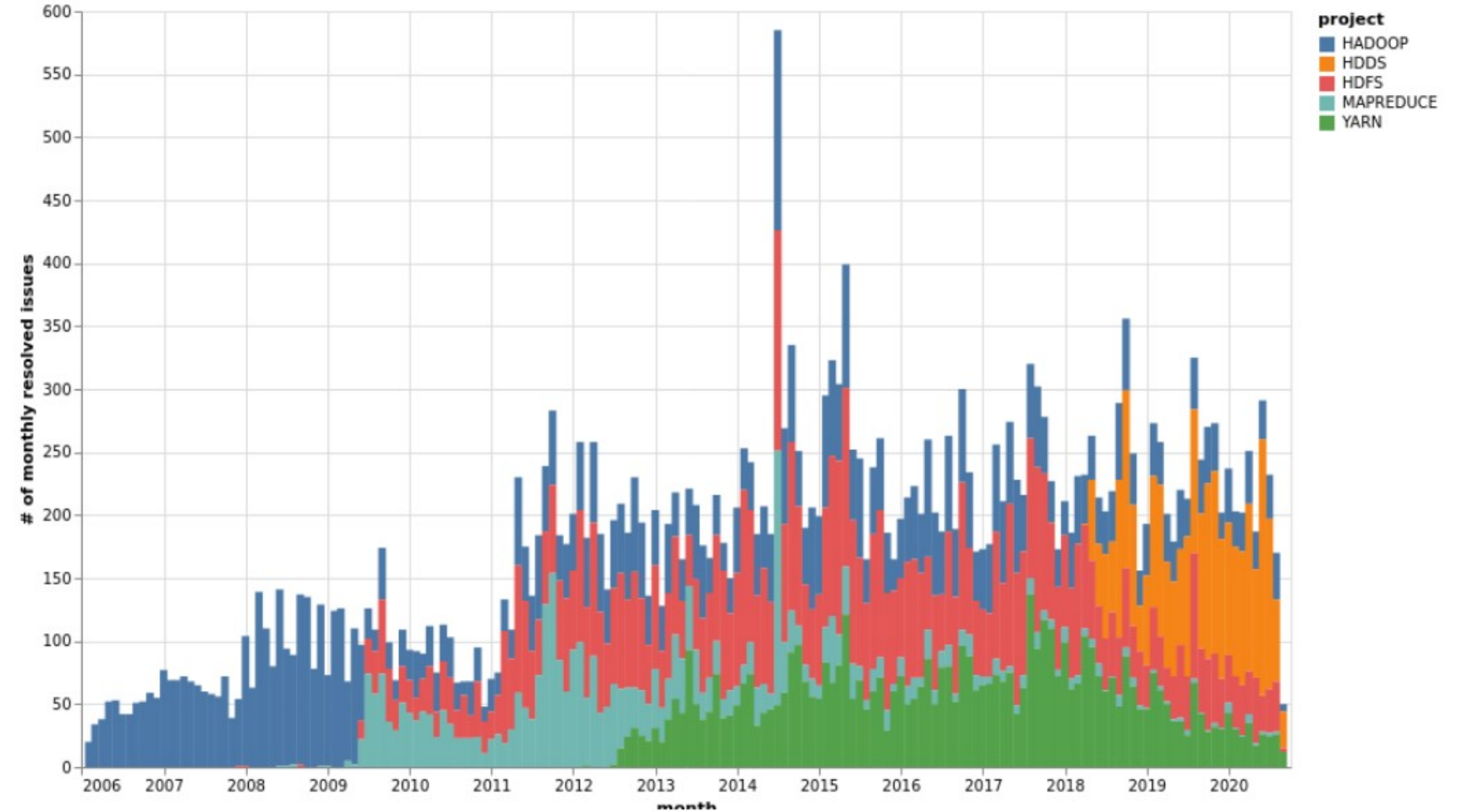
- small files ??
- using from ML/Data science / s3 ??

HCFS cloud connectors

- Consitency ??
- On-prem ??

History.md

- Started as a feature branch in Hadoop
- Merged in 2018 to Hadoop trunk
  - Built by optional profile
  - Separated release lifecycle
  - Separated subproject (“HDDS”)
- 2019 Q4: Moved to a separated git repostory (apache/hadoop-ozone)
- 2020.03: First beta release
- 2020.09 Ozone 1.0.0
- 2020.10 Vote: Apache ~~Hadoop~~ Ozone


project
HADOOP
HDDS
HDFS
MAPREDUCE
YARN
# of monthly resolved issues
600
550
500
450
400
350
300
250
200
150
100
50
0
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
month

Apache Hadoop Ozone

- **SCALEABILITY**
- **USABILITY**

1 billion keys?
DONE

Apache Hadoop Ozone

- Ozone is a scalable, redundant, and distributed object store for Hadoop

SCALABLE

Ozone is designed to scale to tens of billions of files and blocks and, in the future, even more.

CONSISTENT

Ozone is a strongly consistent object store. This consistency is achieved by using protocols like RAFT.

CLOUD-NATIVE

Ozone is designed to work well in containerized environments like YARN and Kubernetes.

SECURE

Ozone integrates with kerberos infrastructure for access control and supports TDE and on-wire encryption.

MULTI-PROTOCOL SUPPORT

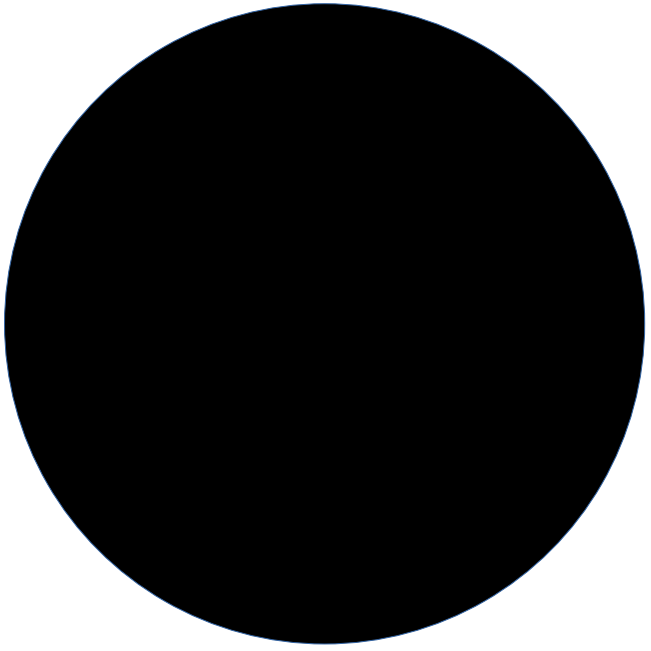
Ozone supports different protocols like S3 and Hadoop File System APIs.

HIGHLY AVAILABLE

Ozone is a fully replicated system that is designed to survive multiple failures.

“Ozone is a
spiritual successor
to HDFS”

Five differences?

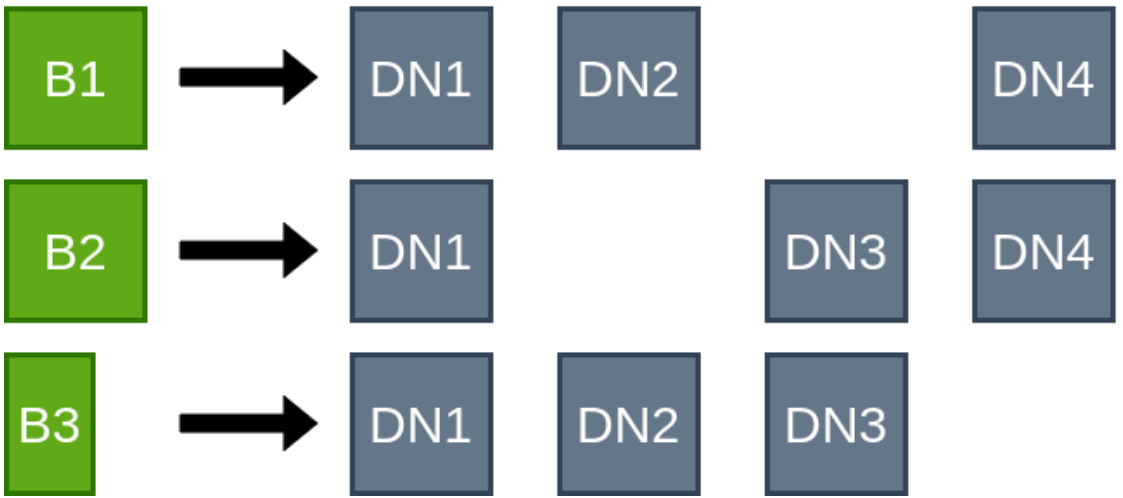


Key/blockspace

How to Store files?

Split file to blocks

Store replicas of blocks on Datanodes

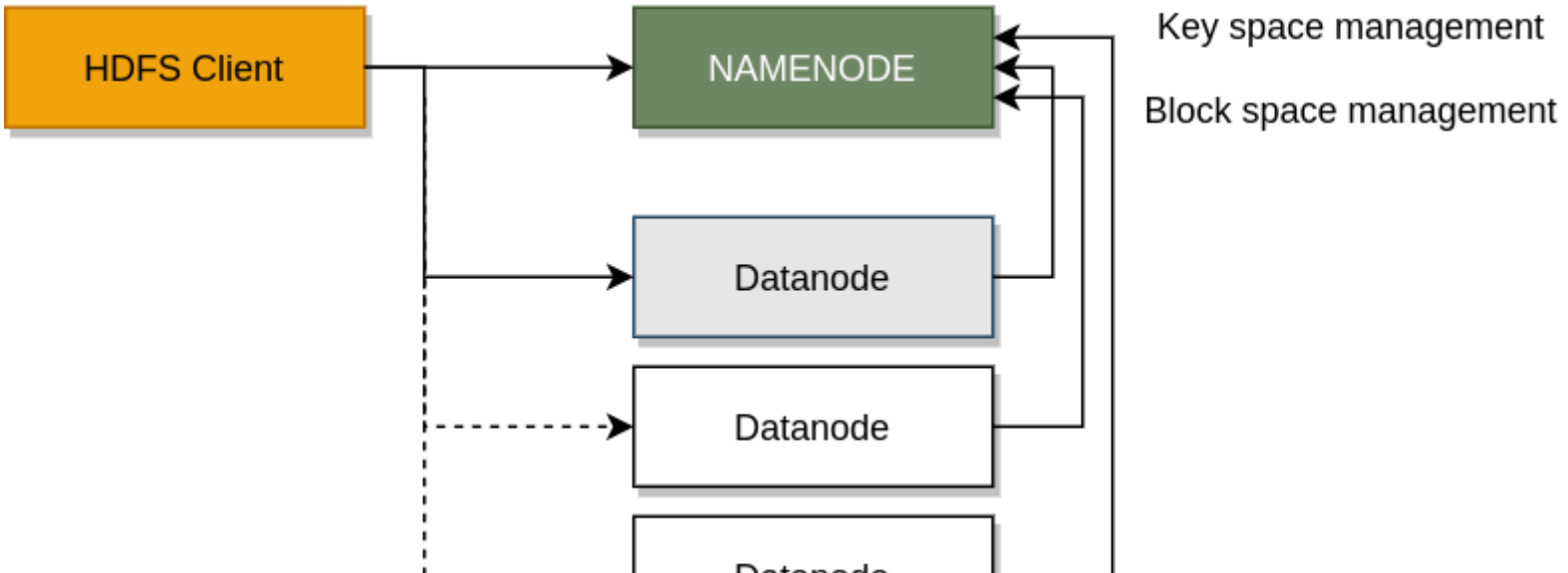


How to Store files?

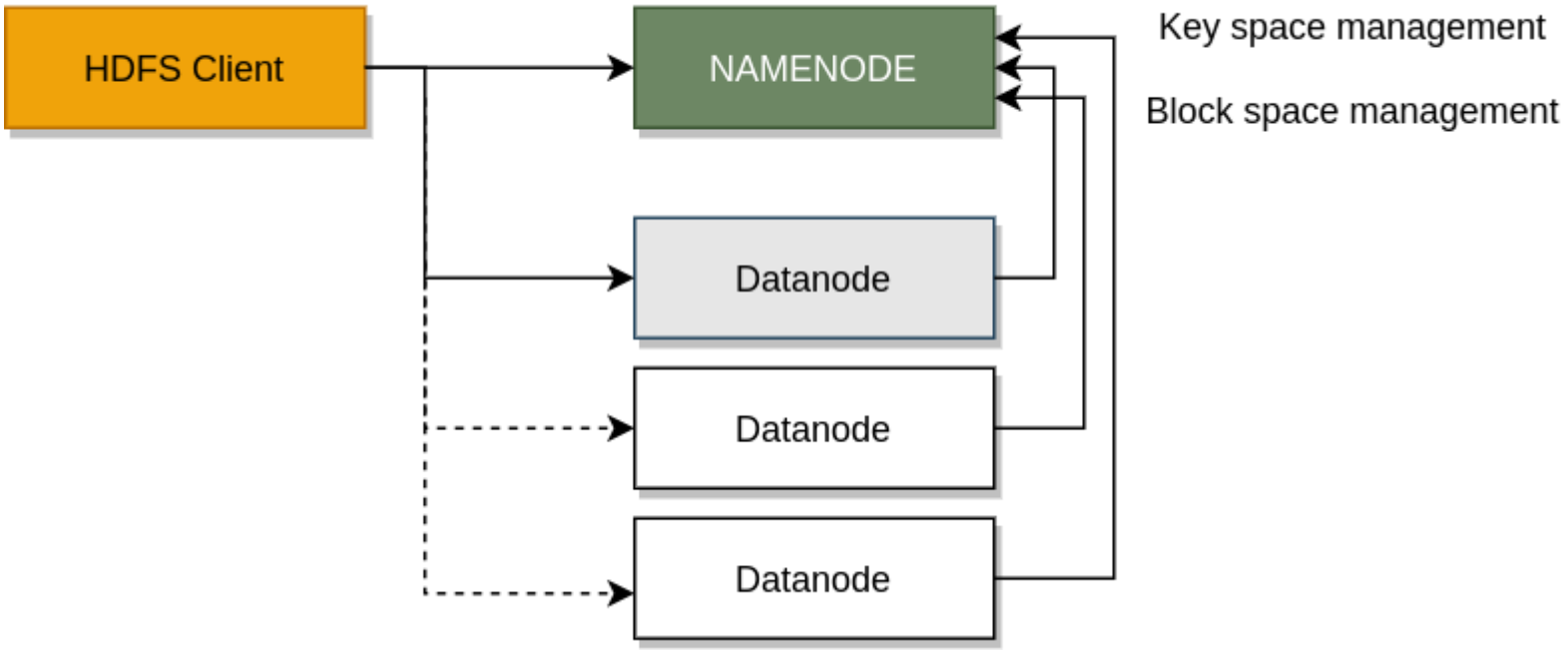
- File → Block[]
- Block → Datanode[]

How to Store files?

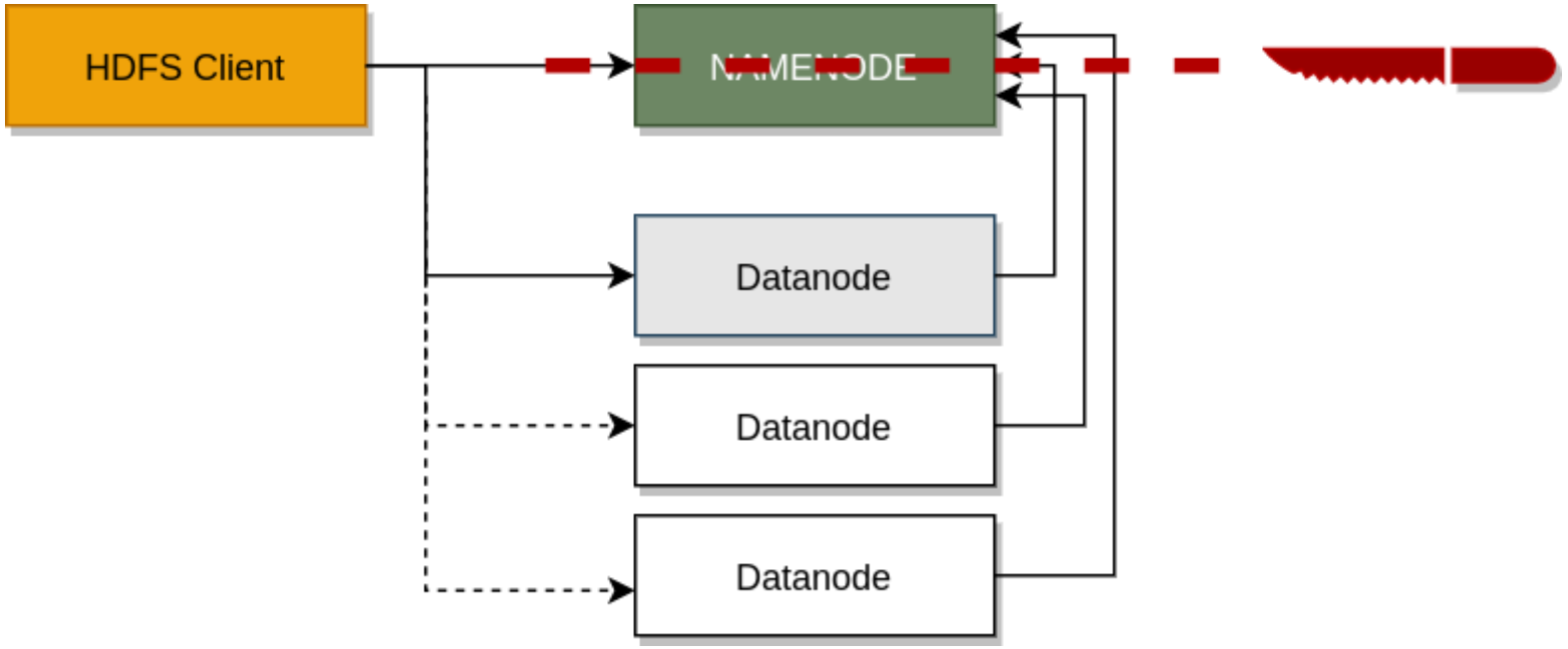
- File → Block[]
- Block → Datanode[]



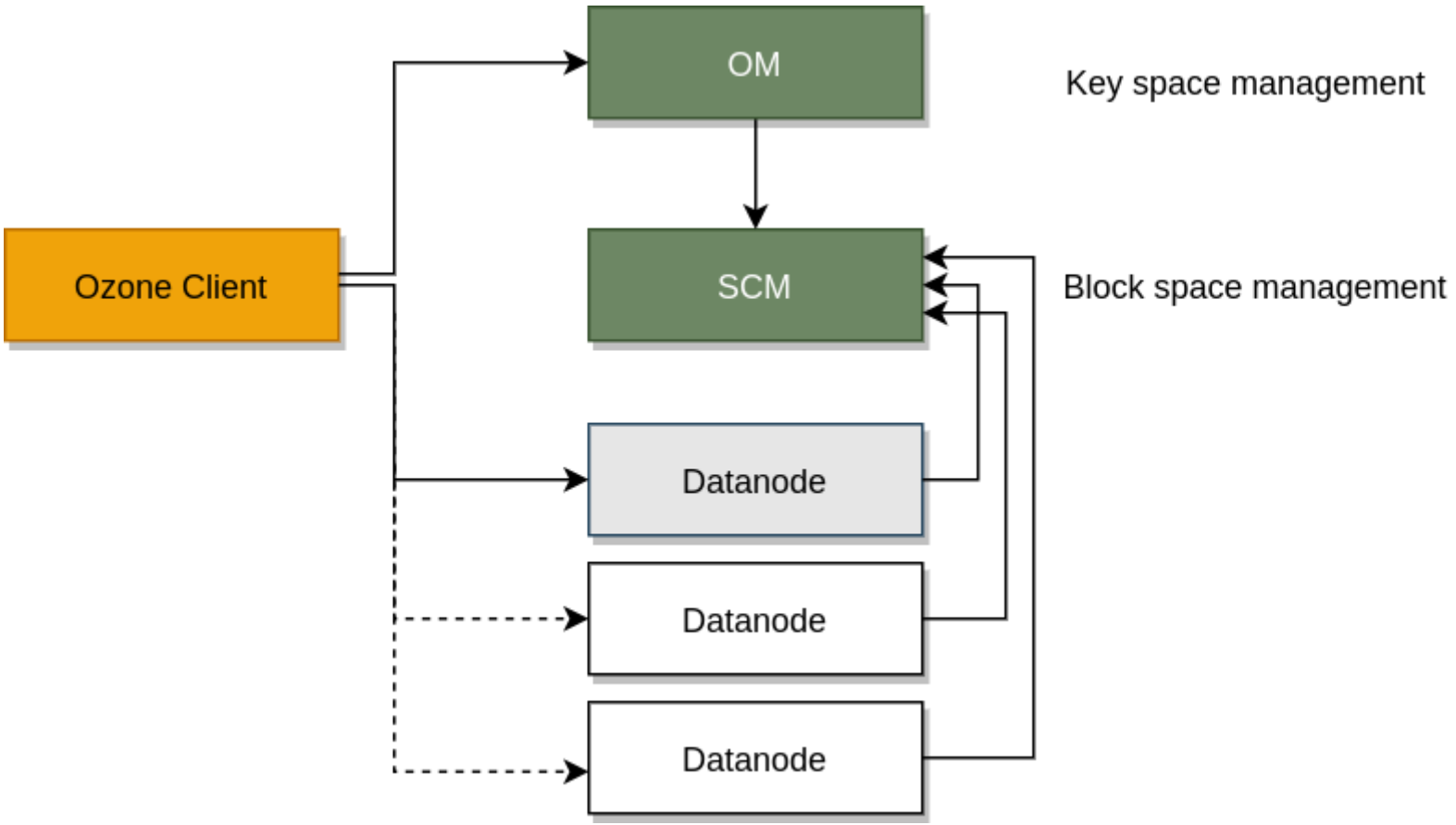
HDFS components



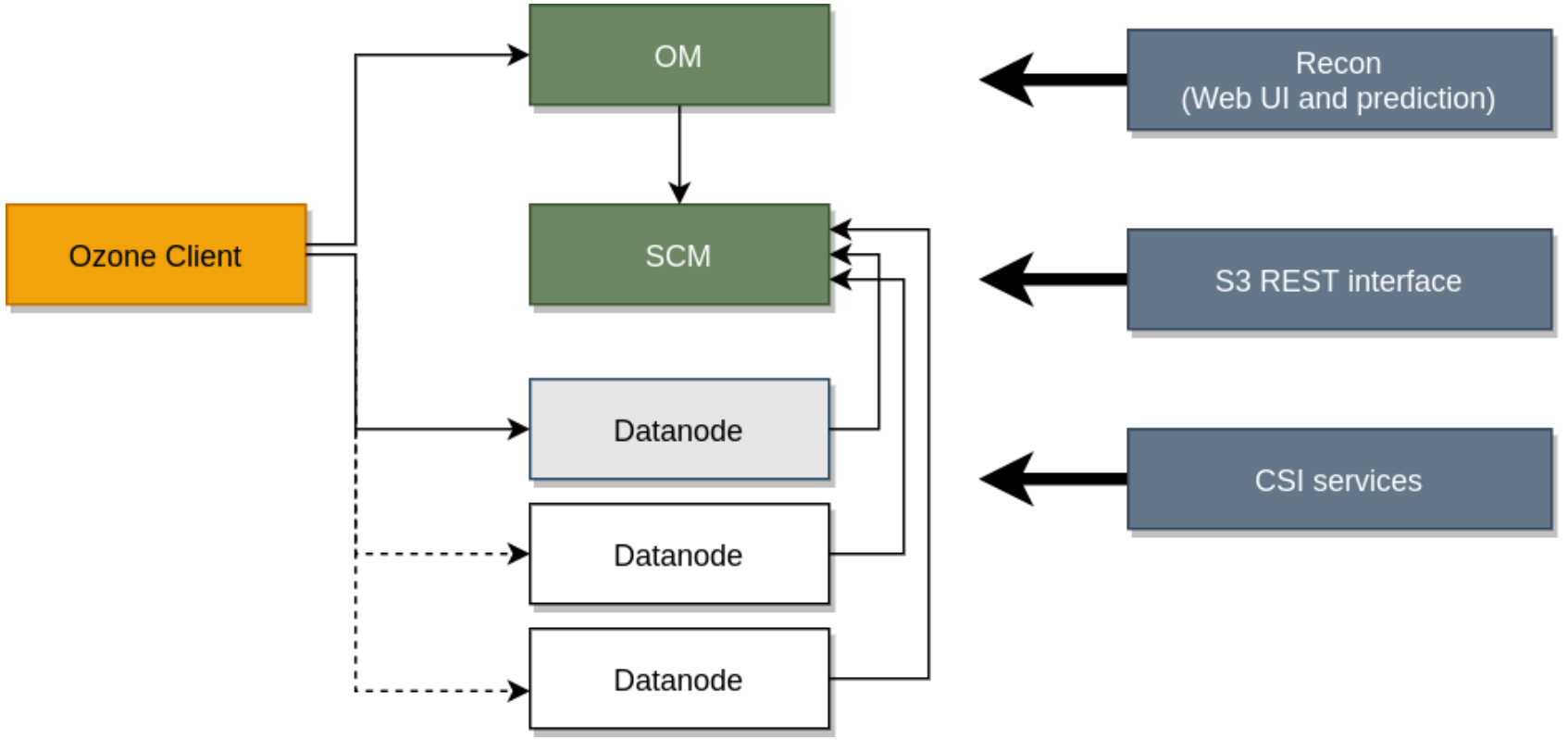
Separate key / block space management



Ozone components

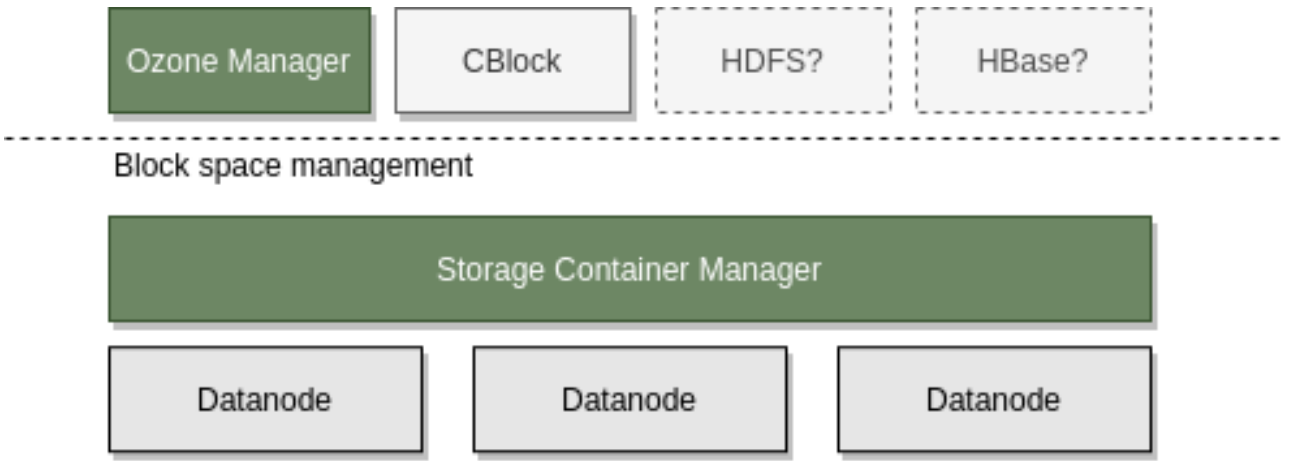


Ozone components (full picture)

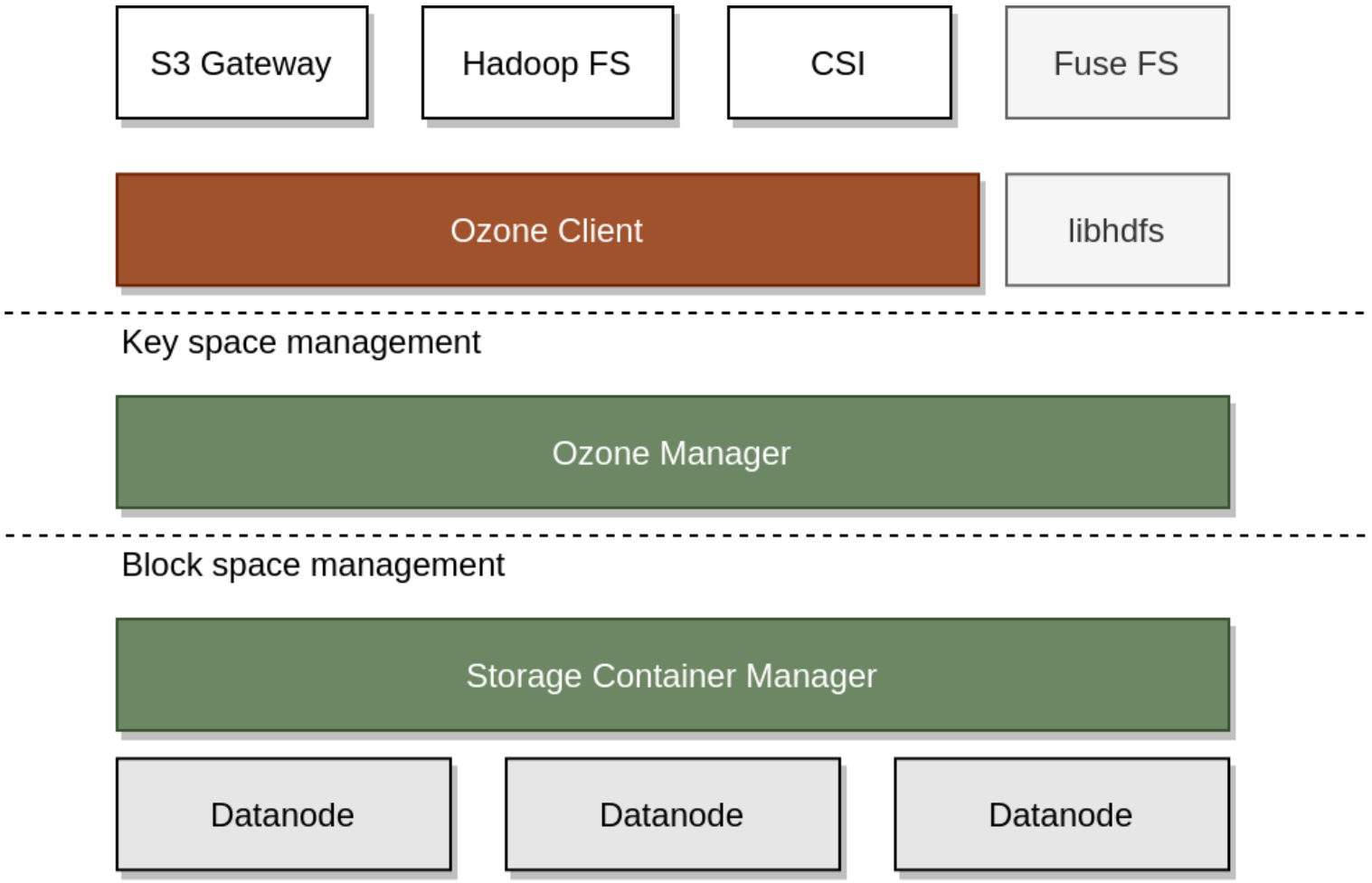


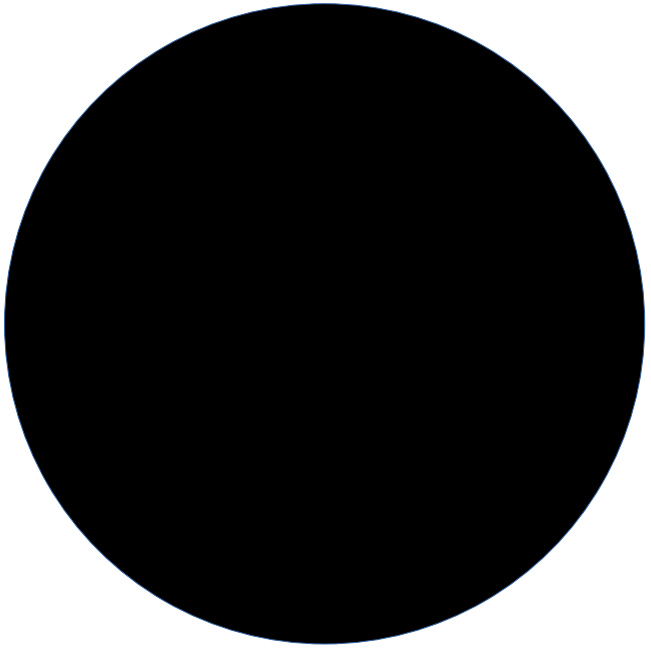
Original Vision

- Use block layer by other applications not just for Object Store



Ozone Layers





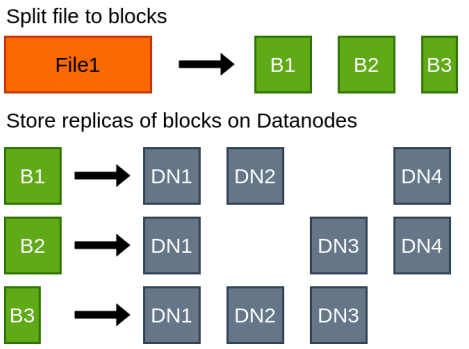
Unit of replication

Small files problems

- The legendary problem with HDFS
- Limits:
  - <~2-300 million files
  - <~3-500 million files with Hadoop dev

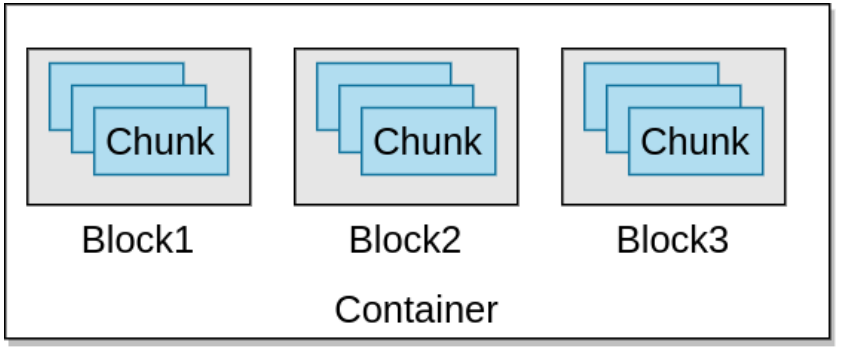
Why is this limit?

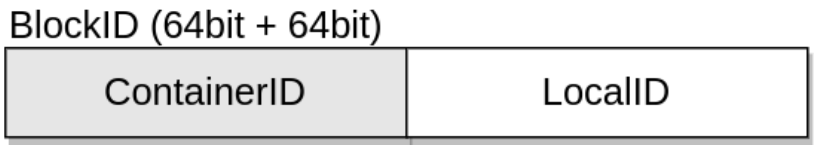
- Each file requires at least one new block
- Each block is replicated in an independent way
- Replication reported back to Namenode:
  - network traffic
  - memory pressure



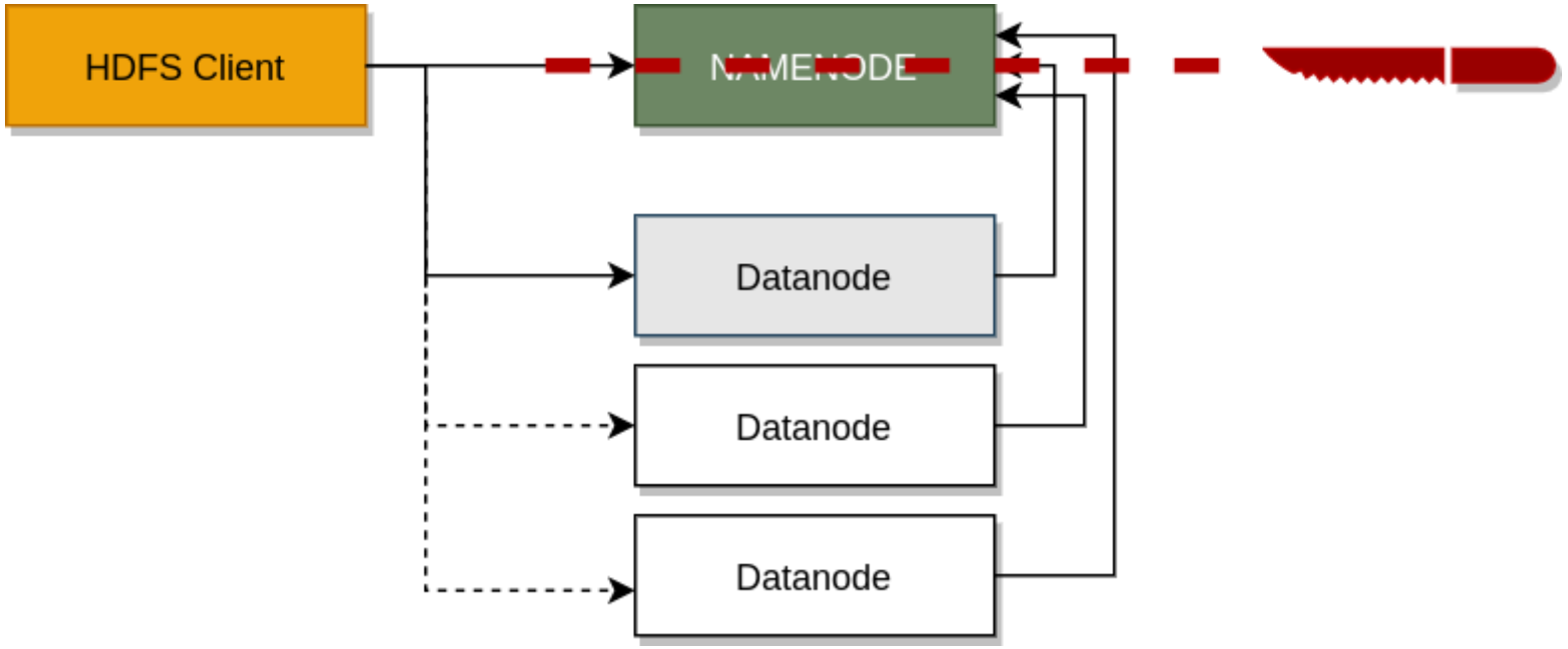
Replicate conatainers

- Container is **the unit of replication**
- Multiple blocks are replicated and **reported** together

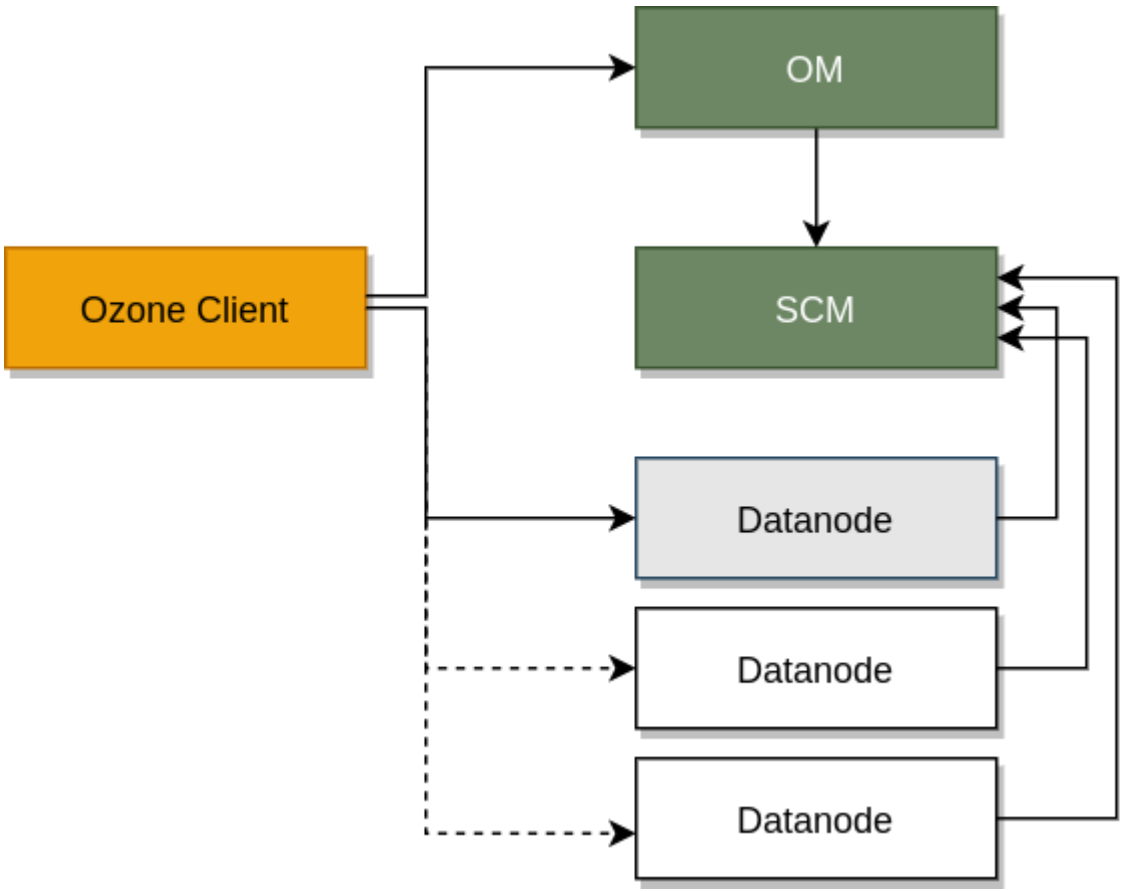


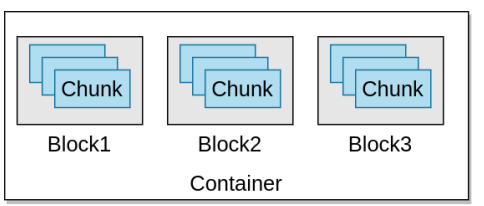


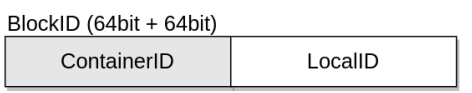
Separate key / block space management

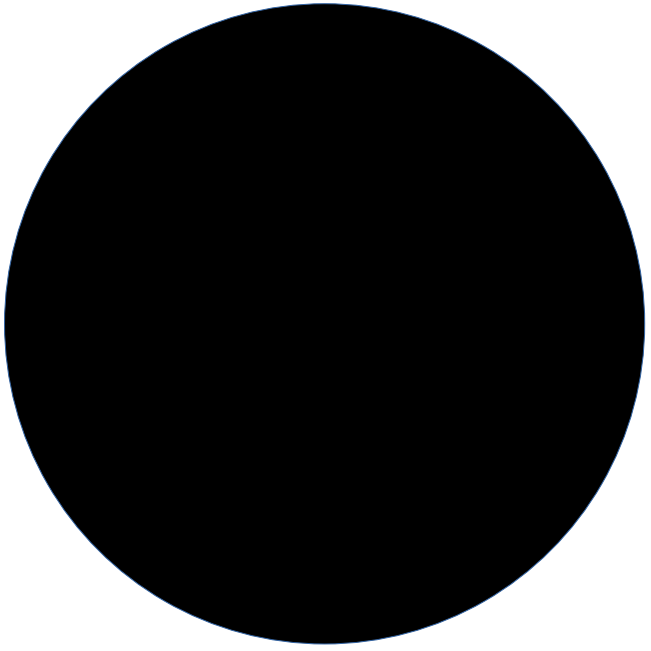


Ozone components









Replication

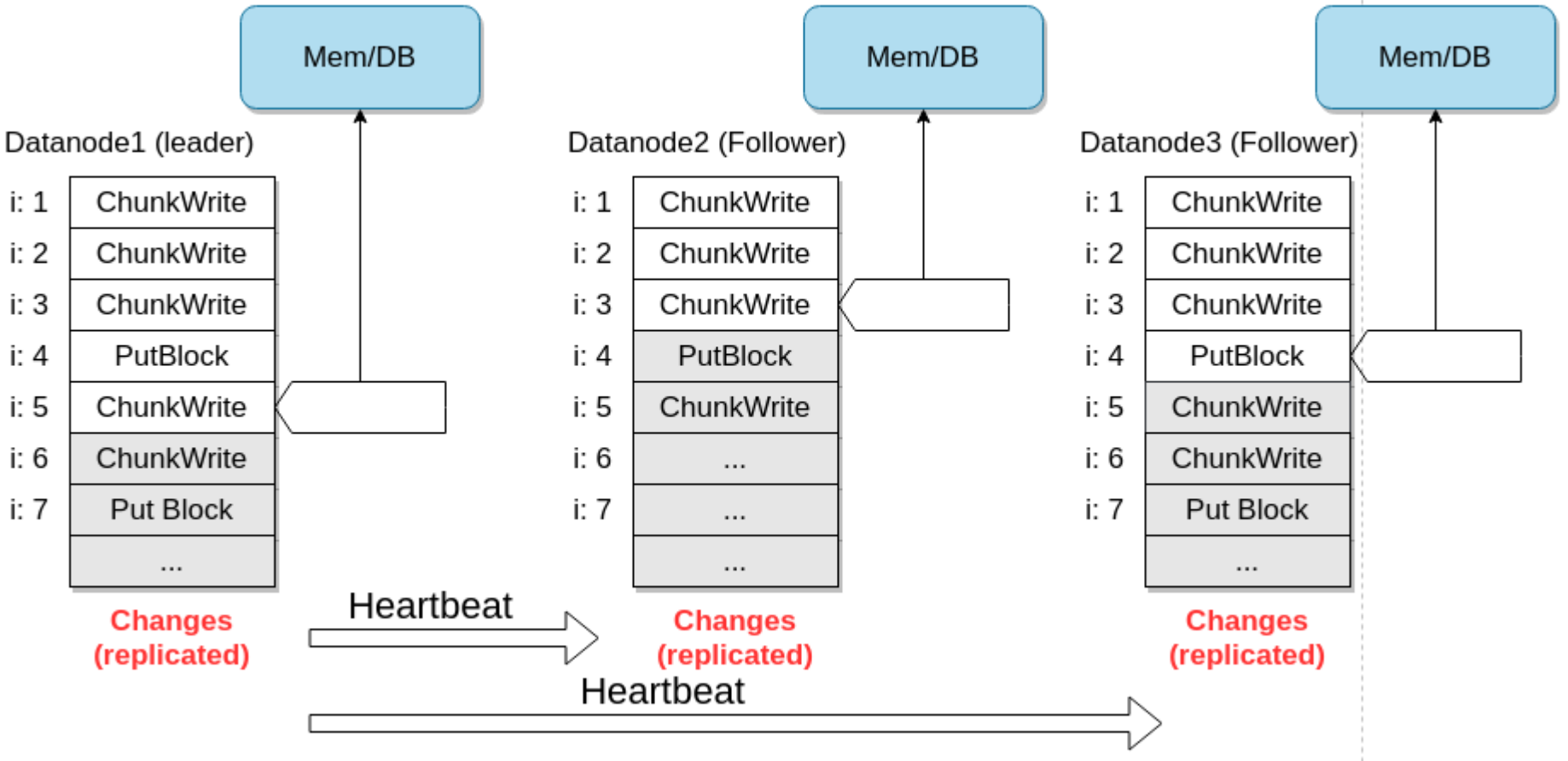
Don’t re-invent
THE WHEEL

How to replicate

- We need to replicate two types of data
  - DATA
  - metadata
- Ozone uses standard replication mechanism: Raft

Raft

- *“Raft is a consensus algorithm for managing a **replicated log**”*
- *“Raft more **understandable** than Paxos and also provides a better foundation for build-ing practical systems”*


Mem/DB
Mem/DB
Mem/DB
Datanode1 (leader)
Datanode2 (Follower)
Datanode3 (Follower)
i: 1 ChunkWrite
i: 2 ChunkWrite
i: 3 ChunkWrite
i: 4 PutBlock
i: 5 ChunkWrite
i: 6 ChunkWrite
i: 7 Put Block
...
i: 1 ChunkWrite
i: 2 ChunkWrite
i: 3 ChunkWrite
i: 4 PutBlock
i: 5 ChunkWrite
i: 6 ...
i: 7 ...
...
i: 1 ChunkWrite
i: 2 ChunkWrite
i: 3 ChunkWrite
i: 4 PutBlock
i: 5 ChunkWrite
i: 6 ChunkWrite
i: 7 Put Block
...
Changes (replicated)
Heartbeat
Changes (replicated)
Heartbeat
Changes (replicated)

In Search of an Understandable Consensus Algorithm

Diego Ongaro and John Ousterhout
Stanford University

Abstract

Raft is a consensus algorithm for managing a replicated log. It produces a result equivalent to (multi-)Paxos, and it is as efficient as Paxos, but its structure is different from Paxos; this makes Raft more understandable than Paxos and also provides a better foundation for building practical systems. In order to enhance understandability, Raft separates the key elements of consensus, such as leader election, log replication, and safety, and it enforces a stronger degree of coherency to reduce the number of states that must be considered. Results from a user study demonstrate that Raft is easier for students to learn than Paxos. Raft also includes a new mechanism for changing the cluster membership, which uses overlapping majorities to guarantee safety.

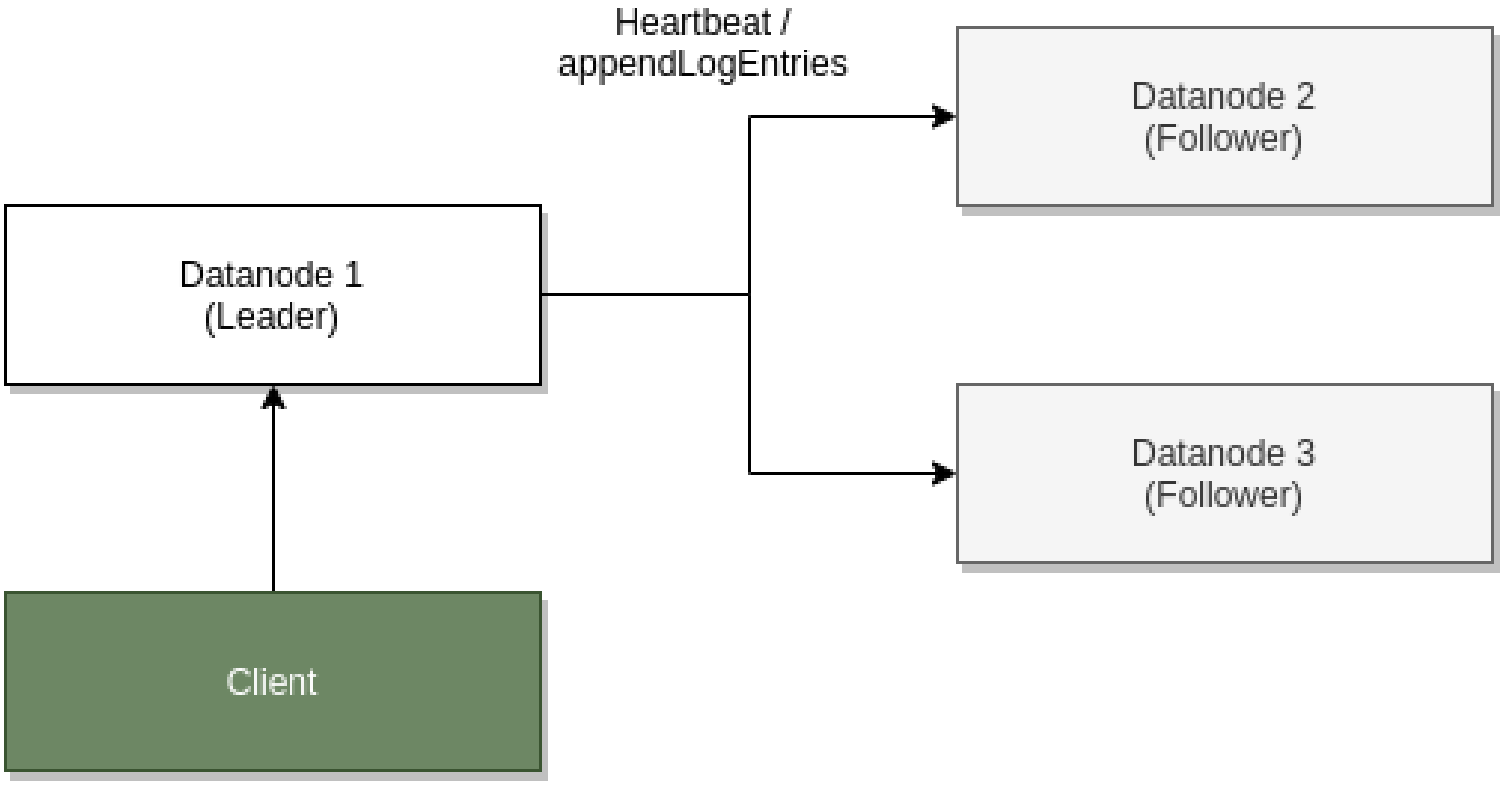
to understand than Paxos: after learning both algorithms, 33 of these students were able to answer questions about Raft better than questions about Paxos.

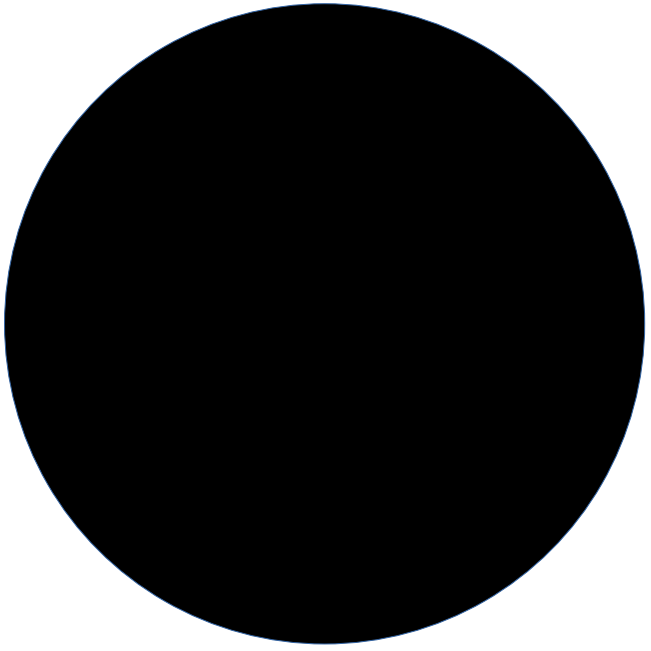
Raft is similar in many ways to existing consensus algorithms (most notably, Oki and Liskov's Viewstamped Replication [27, 20]), but it has several novel features:

- **Strong leader:** Raft uses a stronger form of leadership than other consensus algorithms. For example, log entries only flow from the leader to other servers. This simplifies the management of the replicated log and makes Raft easier to understand.
- **Leader election:** Raft uses randomized timers to elect leaders. This adds only a small amount of mechanism to the heartbeats already required for any consensus algorithm, while resolving conflicts simply and rapidly.

Apache Ratis (Incubator)

- Embeddable (!) RAFT implementation
- Pluggable
  - pluggable transport, state machine
- High performance
  - Advanced batching, ad-hoc heartbeats
- Used in:
  - datanode groups (pipelines)
  - HA (leader nodes: SCM, OM)


Heartbeat /
appendLogEntries
Datanode 2
(Follower)
Datanode 1
(Leader)
Datanode 3
(Follower)
Client



Persistence

STORAGE=

DATA+METADATA

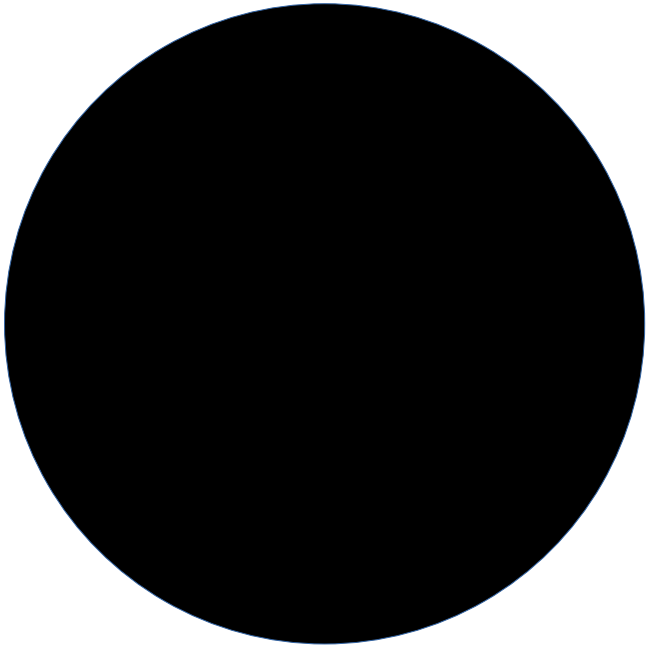
How to persist metadata?

- Store everything in the memory?
  - Memory is a hard limit
- Store in external database?
  - Additional complexity
- Memory and/or local store?
  - Using fast local store and/or memory

Don’t re-invent
THE WHEEL

RocksDB

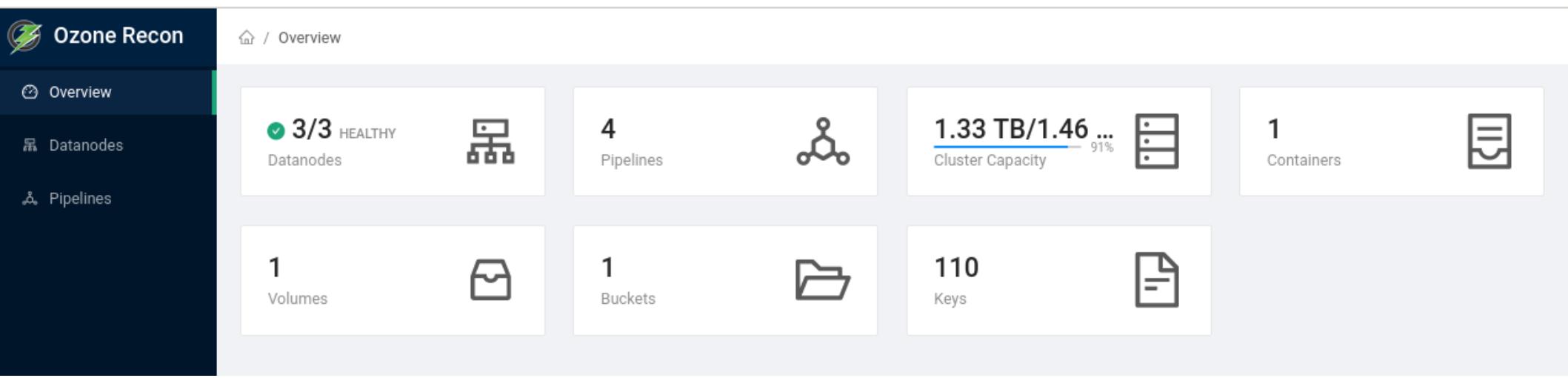
- Well tested local key value store
- Fork of LevelDb
- Based on log-structured merge trees
- Widely used (MySQL, Mongo, ...)
- Don’t need to keep everything in the memory

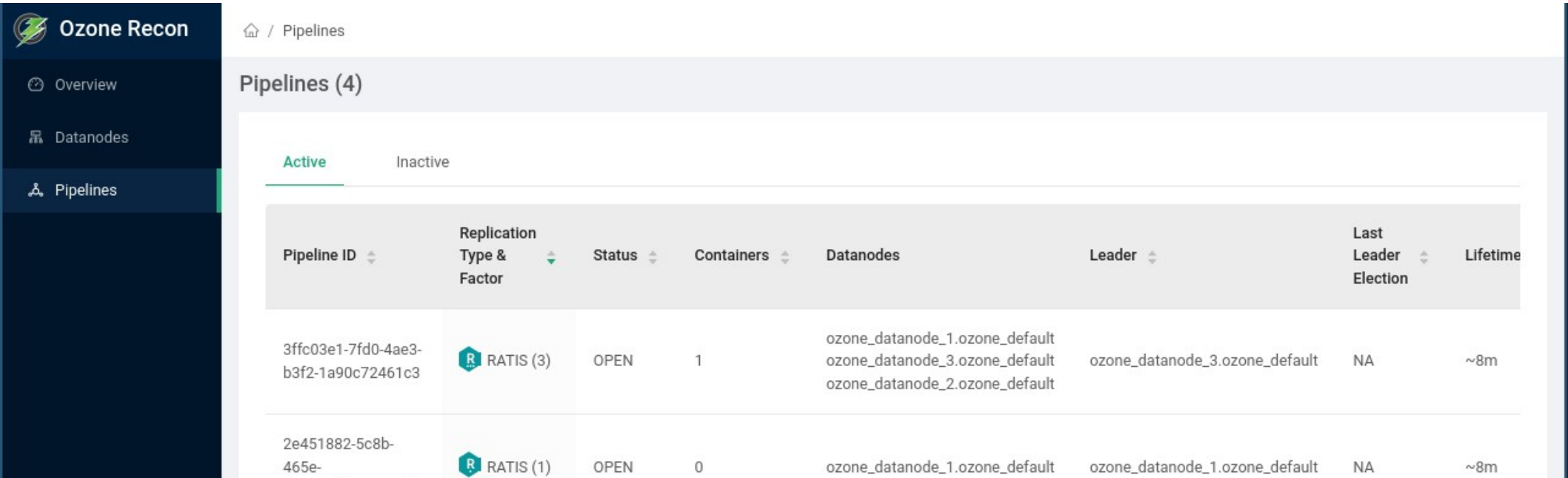


Usability

How to ….?

- Start
  - *cd compose/ozone && docker-compose up*
- Use it from different apps
  - **S3 + HCFS + CSI**
- Devtools & Ops tools
  - config tags and groupping


Ozone Recon
Overview
Datanodes
Pipelines
/ Overview
3/3 HEALTHY
Datanodes
4
Pipelines
1.33 TB/1.46 ...
91%
Cluster Capacity
1
Containers
1
Volumes
1
Buckets
110
Keys


Ozone Recon
Overview
Datanodes
Pipelines
/ Pipelines
Pipelines (4)
Active
Inactive
Pipeline ID
Replication Type & Factor
Status
Containers
Datanodes
Leader
Last Leader Election
Lifetime
3ffc03e1-7fd0-4ae3-b3f2-1a90c72461c3
RATIS (3)
OPEN
1
ozone_datanode_1.ozone_default
ozone_datanode_3.ozone_default
ozone_datanode_2.ozone_default
ozone_datanode_3.ozone_default
NA
~8m
2e451882-5c8b-465e-
RATIS (1)
OPEN
0
ozone_datanode_1.ozone_default
ozone_datanode_1.ozone_default
NA
~8m

aws S3 protocol

Hadoop FS

CSI

Apache Hadoop Ozone

hadoop.apache.org/ozone

THE APACHE SOFTWARE FOUNDATION

YouTube HU

Search

Home

Trending

Subscriptions

Library

History

Your videos

Watch later

Ozone Explained

Show more

Apache Hadoop Ozone unofficial

40 subscribers

HOME VIDEOS PLAYLISTS CHANNELS

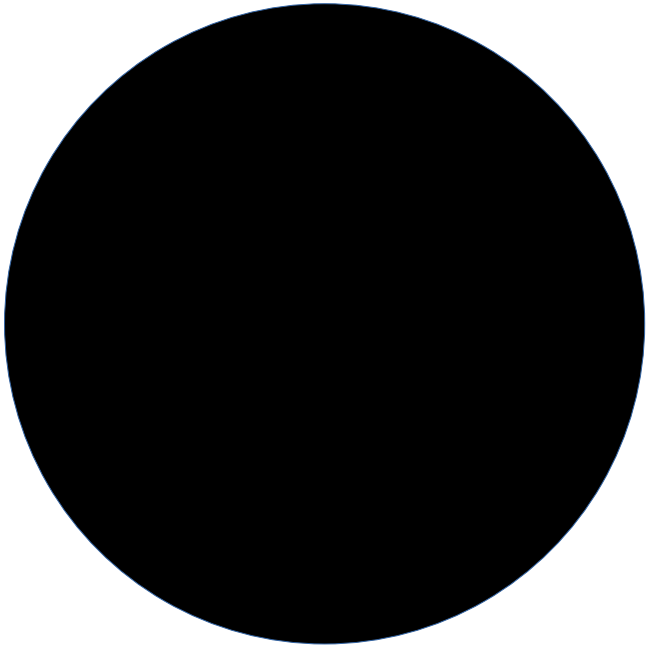
Created playlists

Ozone Explained

VIEW FULL PLAYLIST

Ozone Development

VIEW FULL PLAYLIST



Q&A

Márton Elek

- elek@apache.org
- twitter.com/anzix
- Kubernetes + Apache Bigdata:
  - **github.com/elek**/flekszible
  - flokkr.github.io