

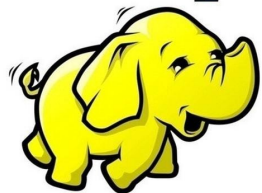


**Apache Hadoop
Ozone**



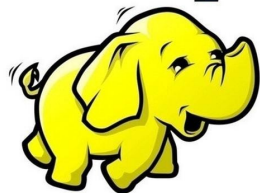
Marton Elek
elek@apache.org

hadoop



Marton Elek
elek@apache.org

hadoop

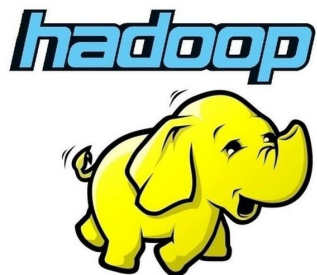


RATIS 

RAFT library for Java



Marton Elek
elek@apache.org

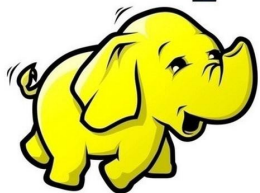


Apache Hadoop Ozone



Marton Elek
elek@apache.org

hadoop



RATIS 

RAFT library for Java

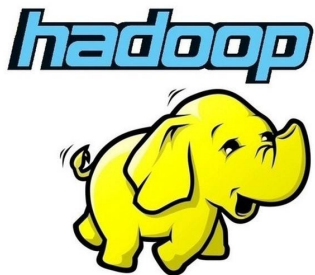


Apache Hadoop Ozone



Marton Elek
elek@apache.org

<https://github.com/elek>



Apache Hadoop Ozone



Marton Elek
elek@apache.org

<https://github.com/elek>

<https://flokkr.github.io>



DATASCIENCE

DATASCIENCE

Storage in the world

Storage in the world

hadoop-hdfs

Storage in the world

hadoop-hdfs

hadoop-aws/aliyun/azure

Storage in the world

hadoop-hdfs

hadoop-aws/aliyun/azure



Consistency

Storage in the world

hadoop-hdfs

hadoop-aws/aliyun/azure



Consistency



Expensive (PB scale)

Storage in the world

hadoop-hdfs

hadoop-aws/aliyun/azure



Consistency



Expensive (PB scale)



Slower (no local data)

Storage in the world

hadoop-hdfs

hadoop-aws/aliyun/azure



Consistency



Expensive (PB scale)



Slower (no local data)




Easy to use / no management




Tool support


Storage in the world


hadoop-hdfs


 Can't handle small files


hadoop-aws/aliyun/azure

 Consistency

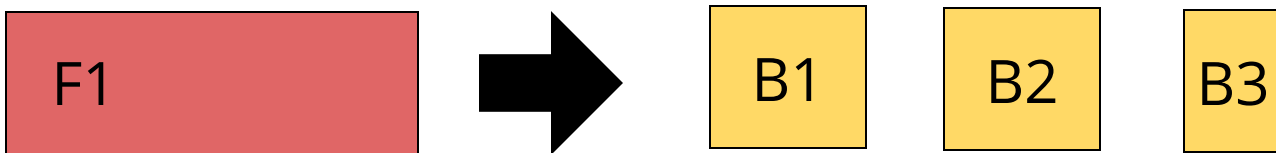
 Expensive (PB scale)

 Slower (no local data)

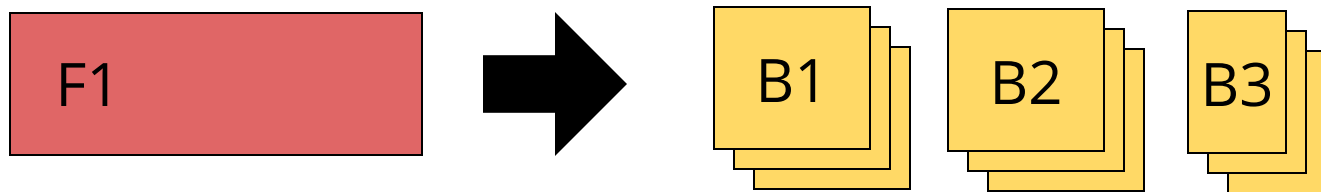
 Easy to use / no management

 Tool support

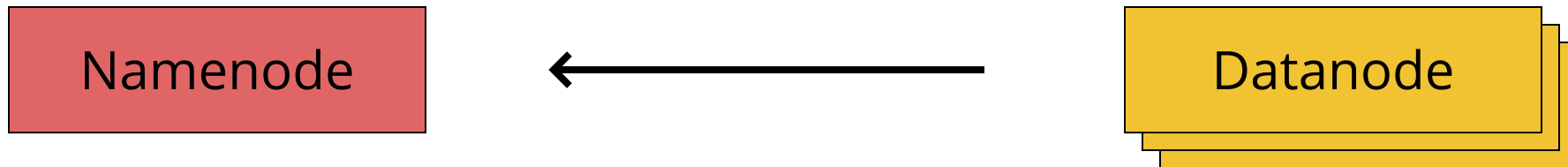
Replication: split the files



Replication: split the files



Replication: split the files



Replication: split the files

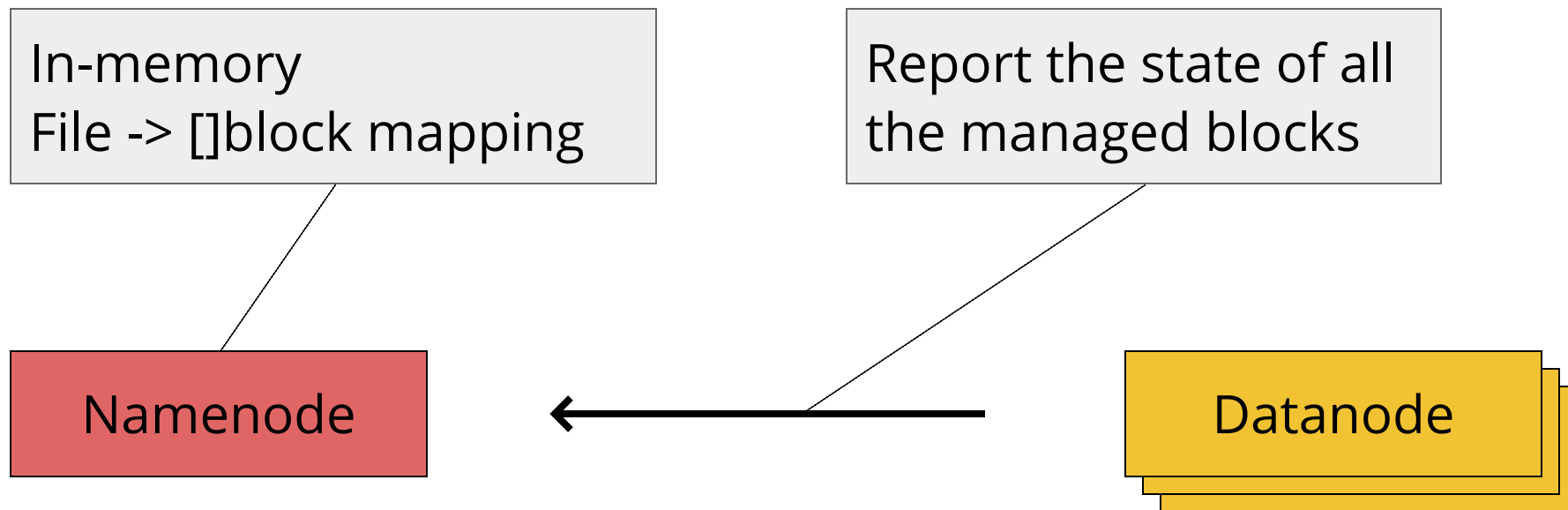
In-memory
File -> []block mapping

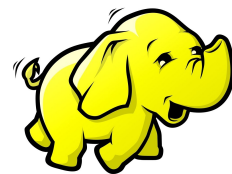
Namenode



Datanode

Replication: split the files





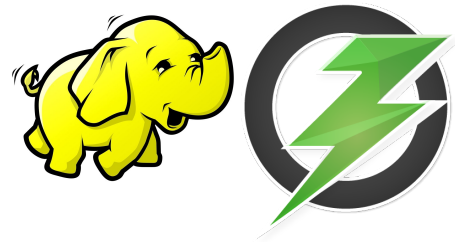
Previously: HDFS

Apache HDFS is designed for large objects --
not for many small objects

Small files create

- Memory pressure on namenode
- Higher network traffic (BlockReports)

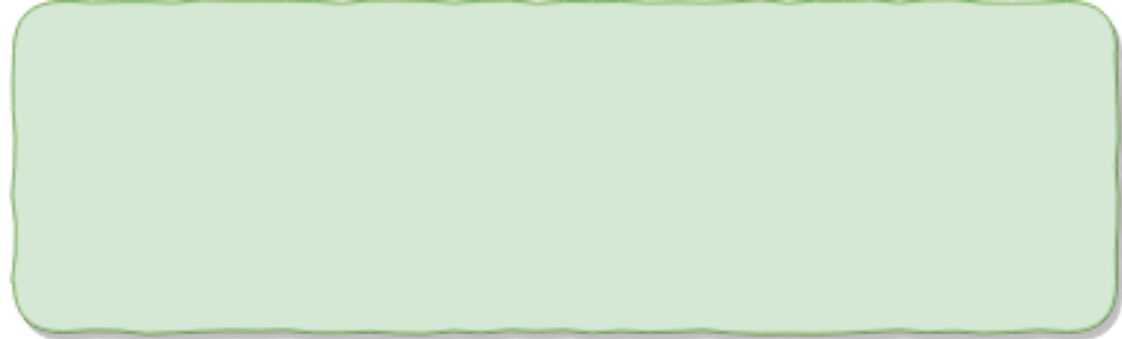
Both of these create scalability issues on Namenode



Ozone and HDFS (past)

- HDFS-5477 Separate Block Management (open)
- HDFS-8286 Partial namespace in memory (open)
- HDFS-1052 HDFS federation (resolved)
- HDFS-10467 Router based federation (resolved/in-progress)
- HDFS-7240 "Scaling HDFS" (Ozone)

Ozone borrows many idea learned from these efforts and is a super set of these approaches



Namenode



Datanode



Datanode



Datanode



Datanode

File -> BlockId[]

BlockId -> DatanodeId[]

Namenode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode



File -> BlockId[]

BlockId -> DatanodeId[]

Namenode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

File -> BlockId[]



BlockId -> DatanodeId[]

HDDS

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Key -> BlockId[]

Ozone



BlockId -> DatanodeId[]

HDDS

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Key -> SuperBlockId[]

Ozone



SuperBlockId -> DatanodeId[]

HDDS



Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Key->

OZONE

Path->

HDFS

SuperBlockId -> DatanodeId[]

HDDS

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Key->

OZONE

Path->

HDFS

FileSystemBlk->

QUADRA

SuperBlockId -> DatanodeId[]

HDDS

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode



HIVE



HBASE

Key->

OZONE

Path->

HDFS

FileSystemBlk->

QUADRA

SuperBlockId -> DatanodeId[]

HDDS

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

Block[]

Datanode

What is Ozone?

- **Ozone** provides Object Store semantic (S3) for Hadoop storage
- **HDDS**: On top of a lower level replication layer (Hadoop Distributed Data Store)

Note: we have two new Hadoop subproject



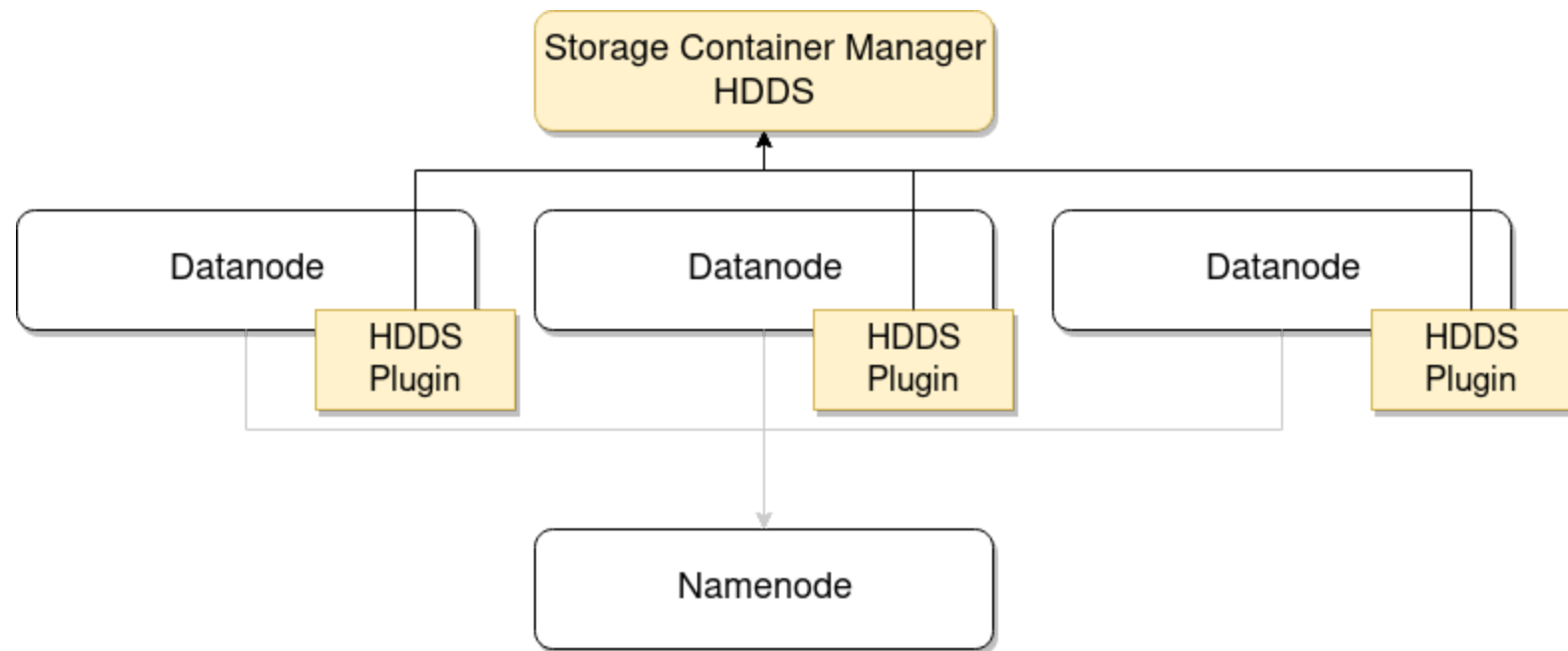
Ozone and HDFS (present)

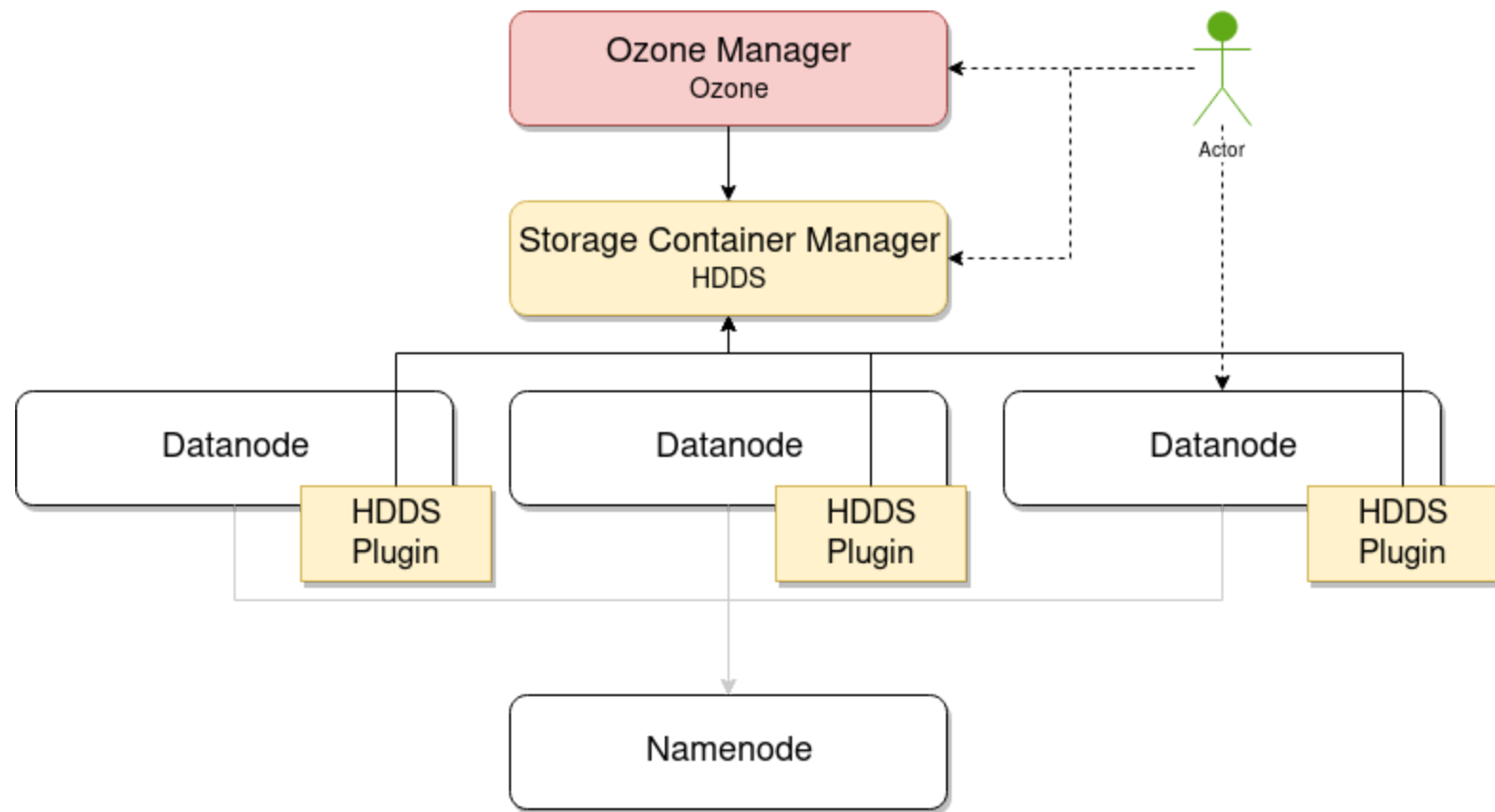
- Apache Hadoop Ozone is a subproject of Apache Hadoop
- Ozone releases are independent releases
- By default it's turned off in the standard hadoop build

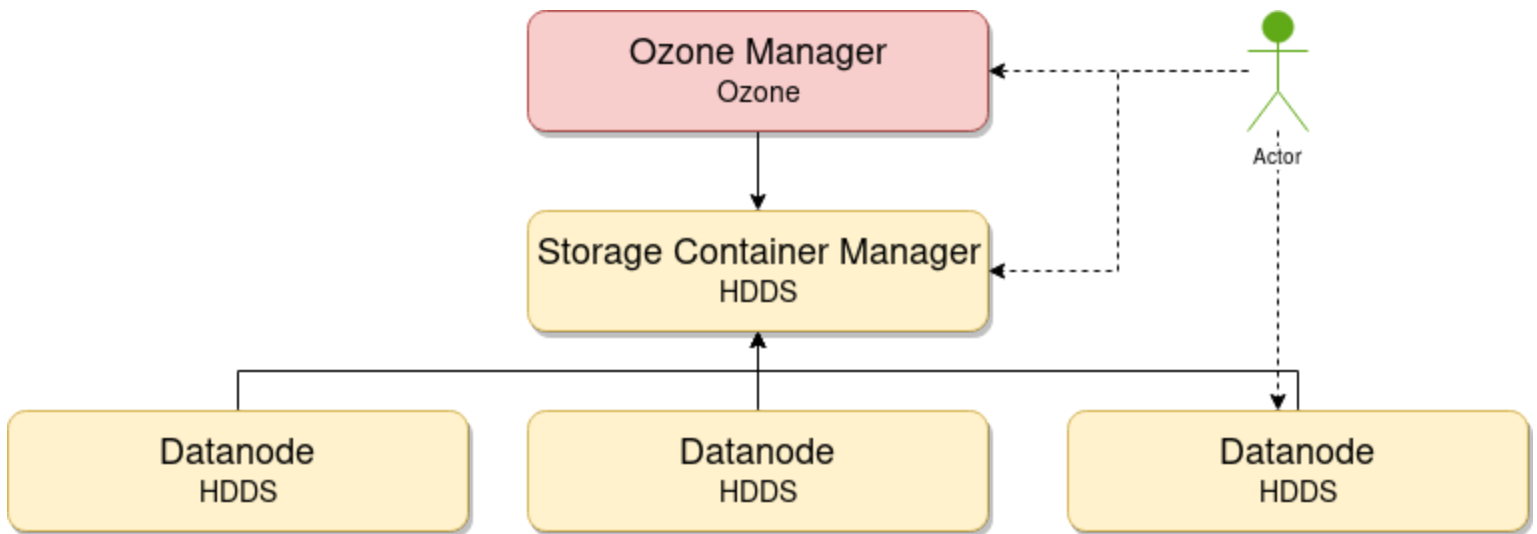


Ozone runtime

- **Worker nodes:** Loadable module running in/out of process with HDFS datanode
- **Master nodes:** two master component instead of the one namenode
 - Storage Container Manager
 - Ozone Manager









Inside HDDS

- Replicating super-blocks (containers)
 - They can contain multiple blocks --> small block reports
- Container replication
 - Open containers: Using Apache Ratis (Raft protocol impl.)
 - Closed containers: Copy (using advanced strategies)



Ozone:

- Consistent
- Fast
- Cloud native
- Easy to use



Roadmap

- **0.2.1-alpha: first release**
- **0.3.0-alpha: s3g + stability**
- 0.4.0-alpha: security
- 0.5.0-beta: HA

DATASCIENCE

DATASCIENCE

Copysets

Copysets: Reducing the Frequency of Data Loss in Cloud Storage

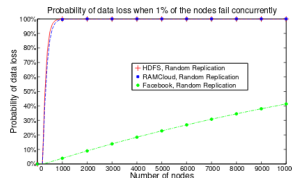
Asaf Cidon, Stephen Rumble, Ryan Stutsman,
Sachin Katti, John Ousterhout and Mendel Rosenblum
Stanford University

cidon@stanford.edu, {rumble,stutsman,skatti,ouster,mendel}@cs.stanford.edu

ABSTRACT

Random replication is widely used in data center storage systems to prevent data loss. However, random replication is almost guaranteed to lose data in the common scenario of simultaneous node failures due to cluster-wide power outages. Due to the high fixed cost of each incident of data loss, many data center operators prefer to minimize the frequency of such events at the expense of losing more data in each event.

We present Copysset Replication, a novel general-



<https://web.stanford.edu/~skatti/pubs/usenix13-copysets.pdf>

Tiered replctn

Tiered Replication: A Cost-effective Alternative to Full Cluster Geo-replication

Asaf Cidon¹, Robert Escriva², Sachin Katti¹, Mendel Rosenblum¹, and Emin Gün Sirer²

¹Stanford University

²Cornell University

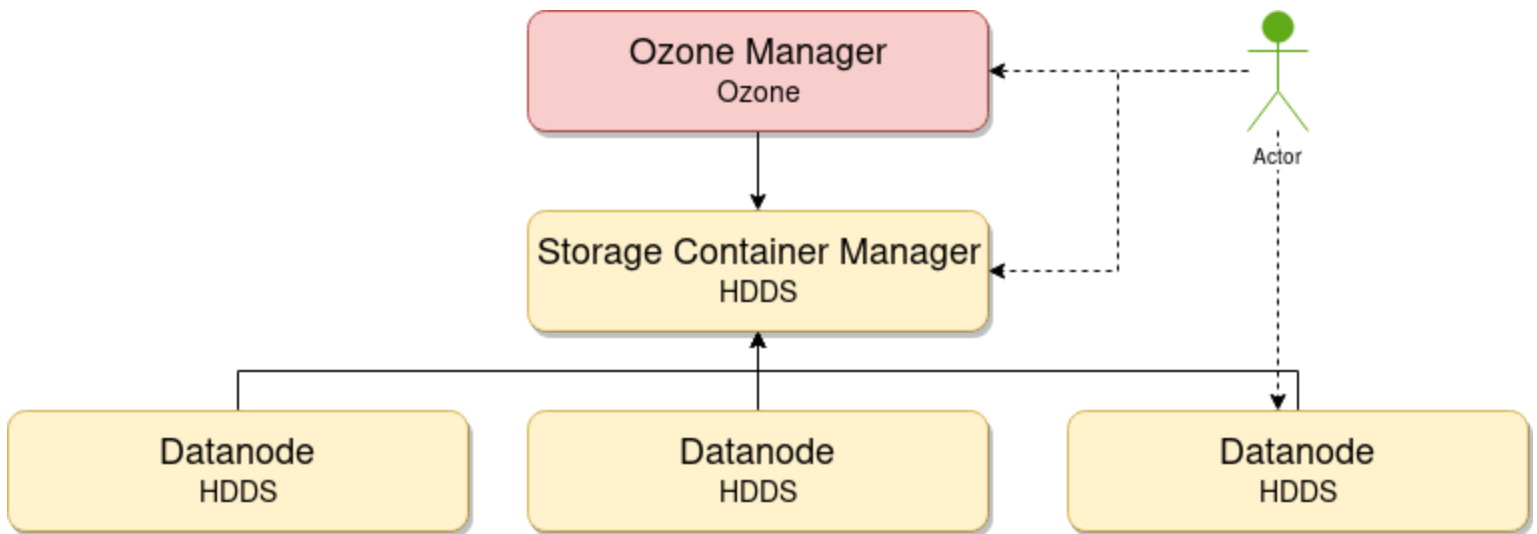
ABSTRACT

Cloud storage systems typically use three-way random replication to guard against data loss within the cluster, and utilize cluster geo-replication to protect against correlated failures. This paper presents a much lower cost alternative to full cluster geo-replication. We demonstrate that in practical settings, using two replicas is sufficient for protecting against independent node failures, while using three random replicas is inadequate for pro-

GFS [15] and Azure [6] typically replicate their data on three random machines to guard against data loss within a single cluster, and geo-replicate the entire cluster to a separate location to guard against correlated failures.

In prior literature, node failure events are broadly categorized into two types: independent node failures and correlated node failures [4, 5, 7, 14, 25, 38]. Independent node failures are defined as events during which nodes fail individually and independently in time (e.g.,

<https://www.usenix.org/system/files/conference/atc15/atc15-paper-cidon.pdf>

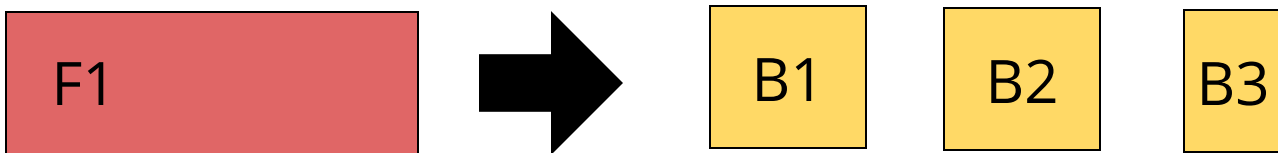




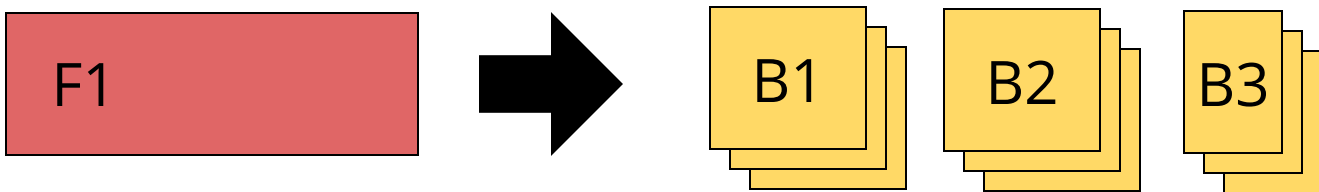
Reminder

- Open containers:
 - Writes are not yet finished
 - Replicated with RAFT (Apache Ratis)
- Closed containers
 - Immutable
 - Copied by the replication manager (HDDS -- SCM)

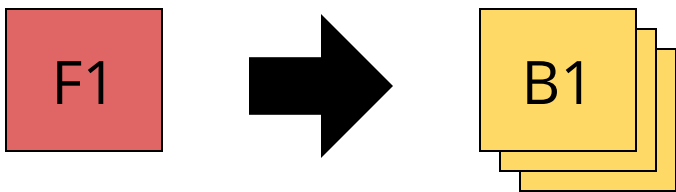
Replication: split the files



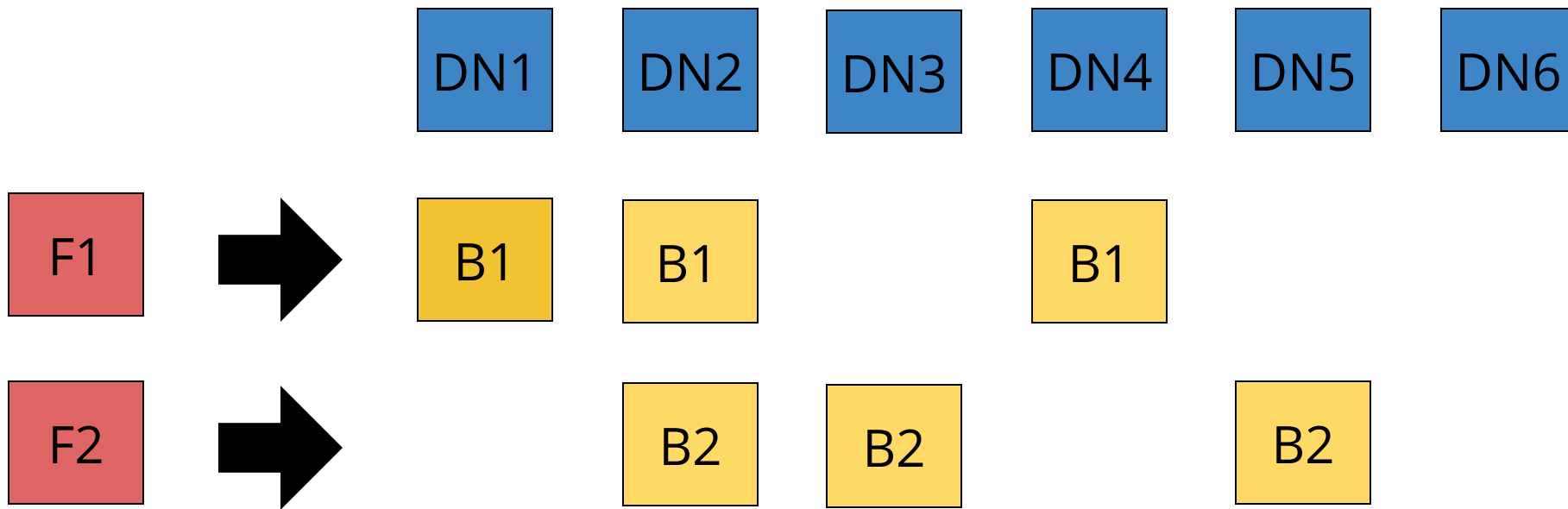
Replication: split the files



Replication: split the files



Replication





Random replication

- 6 datanodes --> 20 different 3-node set:
 - [1 2 3] [2 5 7] [6 8 9] ...
- 2000 blocks
 - [1 2 3] --> ~100 blocks
 - [1 2 4] --> ~100 blocks
 - [2 5 7] --> ~100 blocks

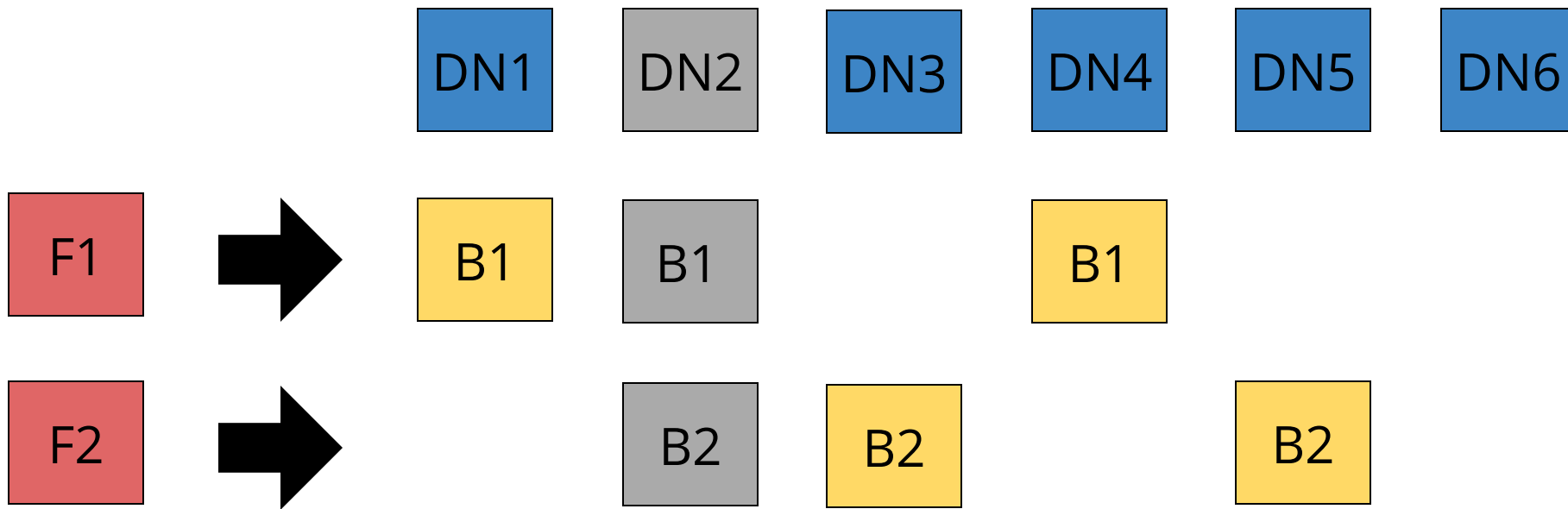


Kill one Datanode



Datanode #2

Replication



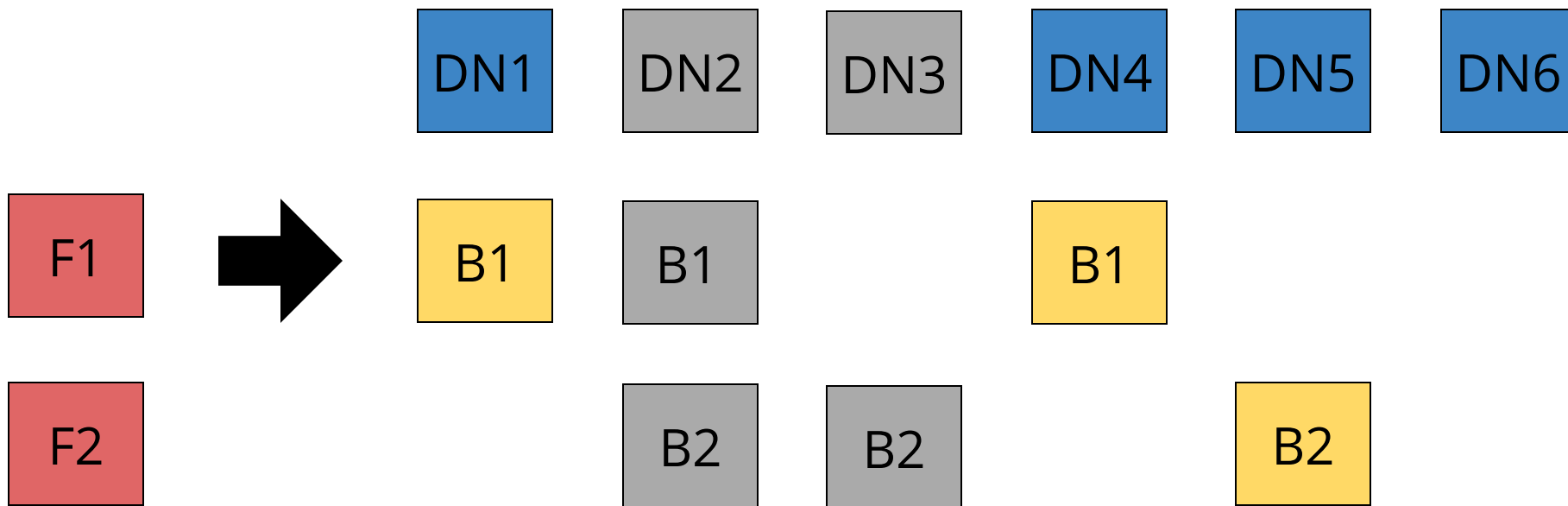


Kill second Datanode



Datanode #3

Replication: split the files



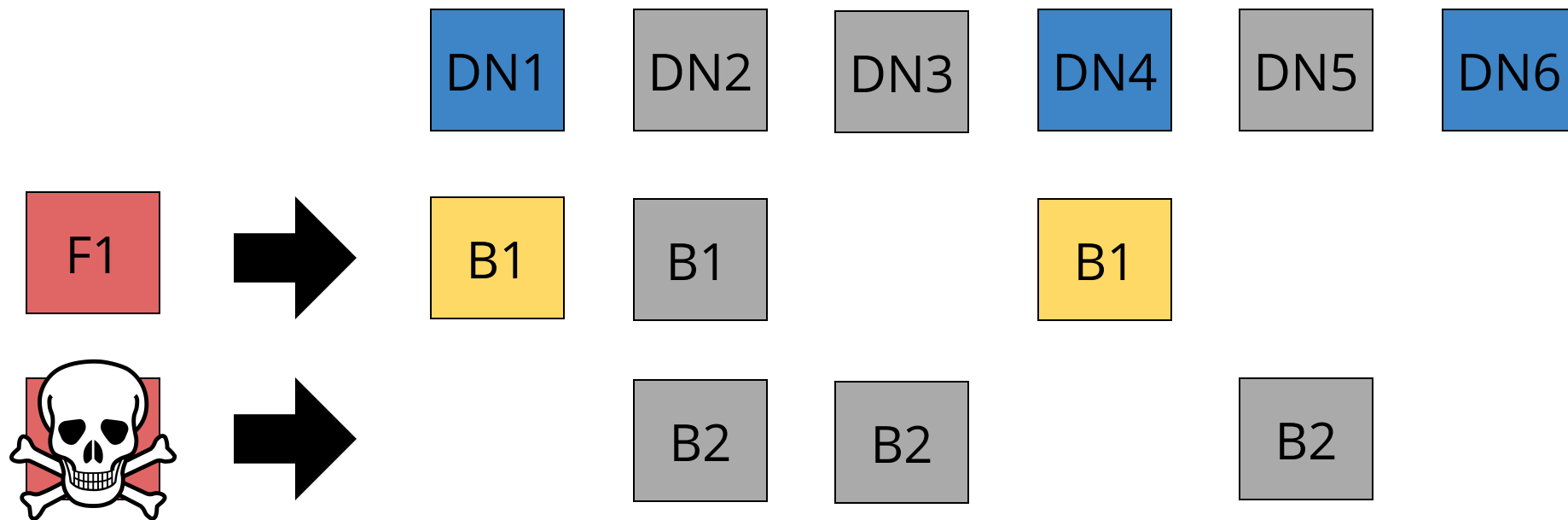


Kill third Datanode



Datanode #5

Replication: split the files



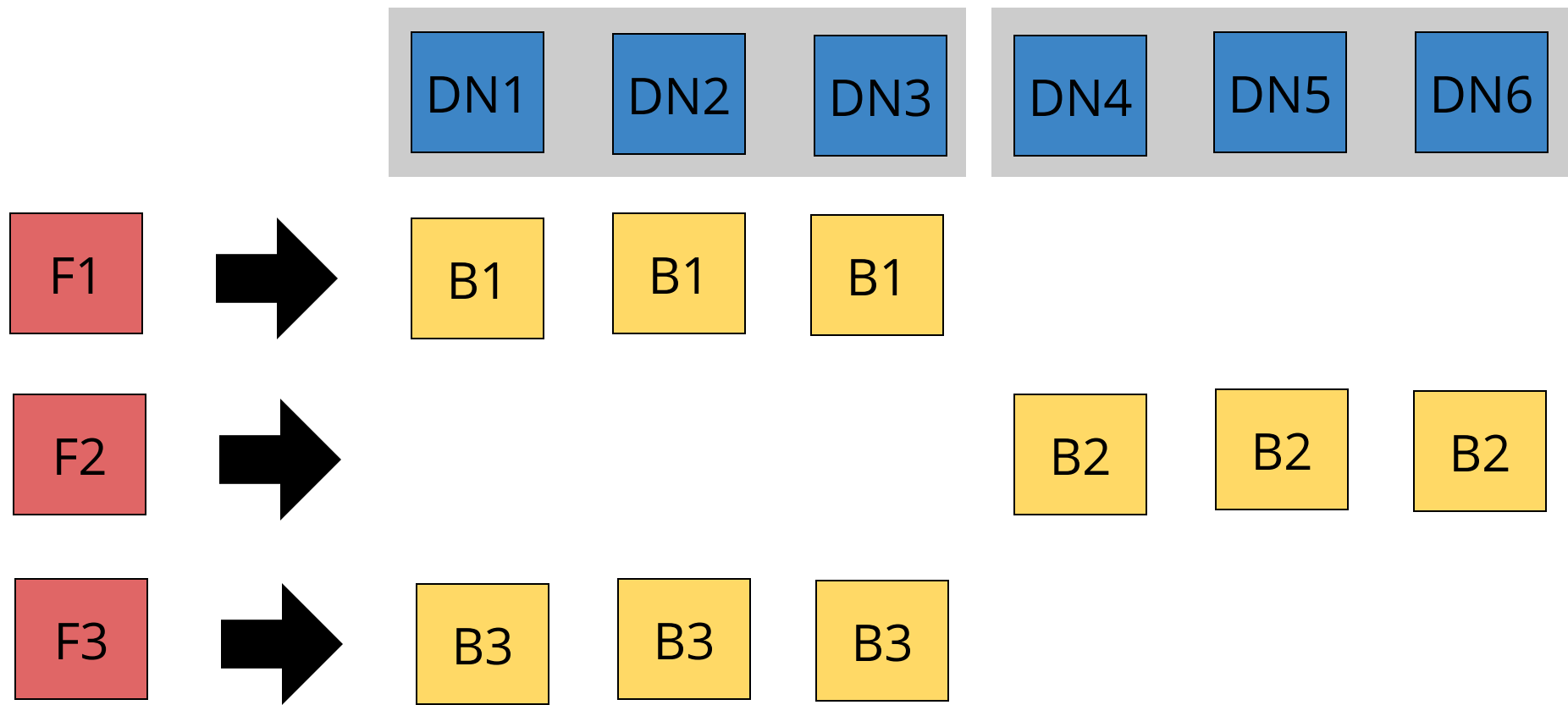
**Does it help
to use more datanodes?**



Scale it up?

- 600 datanodes --> $600!/597!/3!$ different 3-node set:
 - $100 * 599 * 598 = 35\,820\,200$
- 3 500 000 000 -> blocks
 - [1 2 3] --> ~100 blocks
 - [1 2 4] --> ~100 blocks
 - [2 5 7] --> ~100 blocks

**How can we do it
better?**





2 group

- 6 datanodes --> group / copyset
 - [1 2 3], [4 5 6]
- 2000 -> blocks
 - [1 2 3] --> ~1000 blocks
 - [4 5 6] --> ~1000 blocks

Kill 3 datanodes randomly

Kill 3 datanodes randomly

- [2 5 6] -> no data loss

Kill 3 datanodes randomly

- [2 5 6] -> no data loss
- [1 3 6] -> no data loss

Kill 3 datanodes randomly

- [2 5 6] -> no data loss
- [1 3 6] -> no data loss
- [1 3 4] -> no data loss

Kill 3 datanodes randomly

- [2 5 6] -> no data loss
- [1 3 6] -> no data loss
- [1 3 4] -> no data loss
- [1 2 3] -> BUMM





2 groups

- 6 datanodes --> 2 copysets
 - [1 2 3], [4 5 6]
- 2000 -> blocks
 - [1 2 3] --> ~1000 blocks
 - [4 5 6] --> ~1000 blocks
- 3 DN failure --> data loss: 2:20
- Lost data: 1000 blocks

Random

- 6 datanodes --> 20 copysets
 - [13 6], [4 7 9]
- 2000 -> blocks
 - [1 2 3] --> ~100 blocks
 - [4 5 6] --> ~100 blocks
- 3 DN failure --> data loss: 20:20
- Lost data: 100 blocks

2 groups

10% chance
50% loss

Random

100% chance
10% loss



Problem?

[Random] recovery

Only one failure (datanode #1) Failed

[(1) 2 3] --> copy 100 blocks from 2/3

[(1) 2 4] --> copy 100 blocks from 2/4

[(1) 2 5] --> copy 100 blocks from 2/5

[(1) 2 6]

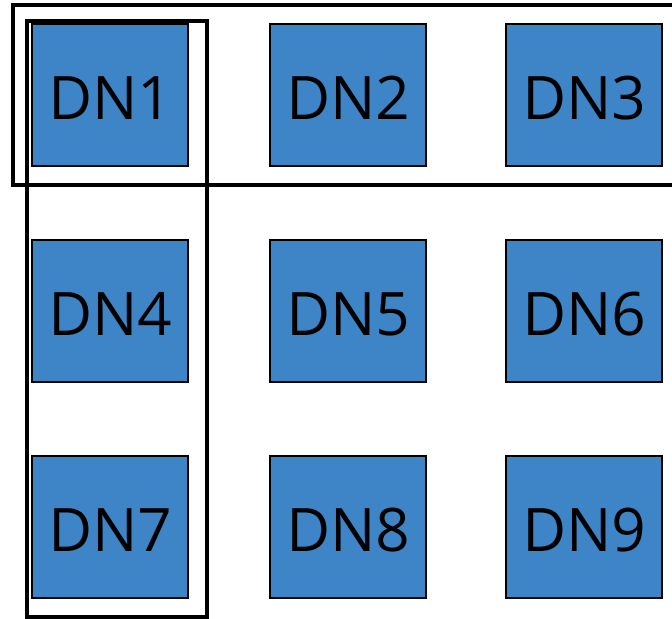
...

[2 groups] recovery

Only one failure (datanode #1)

[4 5 6] -> ok

[_ 2 3] -> 1000 blocks should be replicated from [2 3]



6 groups: 3 col, 3 row

3 random node failures:

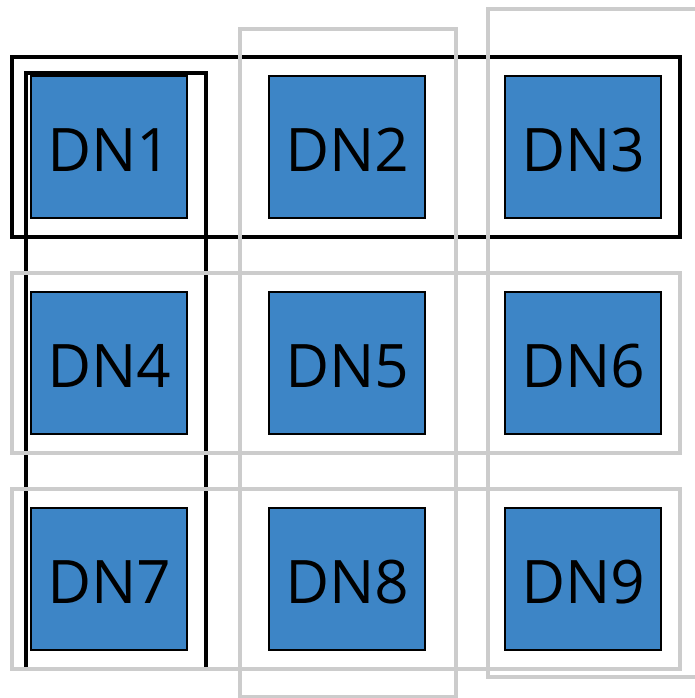
possible in 84 ways

Change to data loss:

$$6/84 = 7\%$$

Source for replication:

4 datanodes



Summary

	Random	2 group	6 group/ copysets
--	--------	---------	----------------------

Chance of data
loss (3 failure)

100%

3.5%

7%

Source of replic.
(1 failure)

all

2

4



Summary

- Two main forces:
 - Number of distinct sets (copysets)
 - Number of source datanode to recover data
- Random replication is not the most effective way
- We can do better
 - Lower chance to loose data
 - More data to loose



Summary (Copolysets)

- Two main forces:
 - Number of distinct sets (copysets)
 - Number of source datanode to recover data
- Random replication is not the most effective way
- We can do better
 - Lower chance to loose data
 - More data to loose



Apache Hadoop Ozone

- S3 compatible Object store for Hadoop
- On top of the Hadoop Distributed Storage layer
 - Advanced replication
 - Advanced block report handling (We love small files)
- Easy to use and run in containerized environments



Q&A