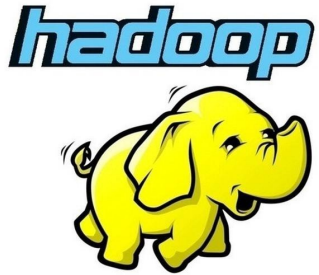
DATASCIENCE



RATIS

RAFT library for Java

Apache Hadoop Ozone

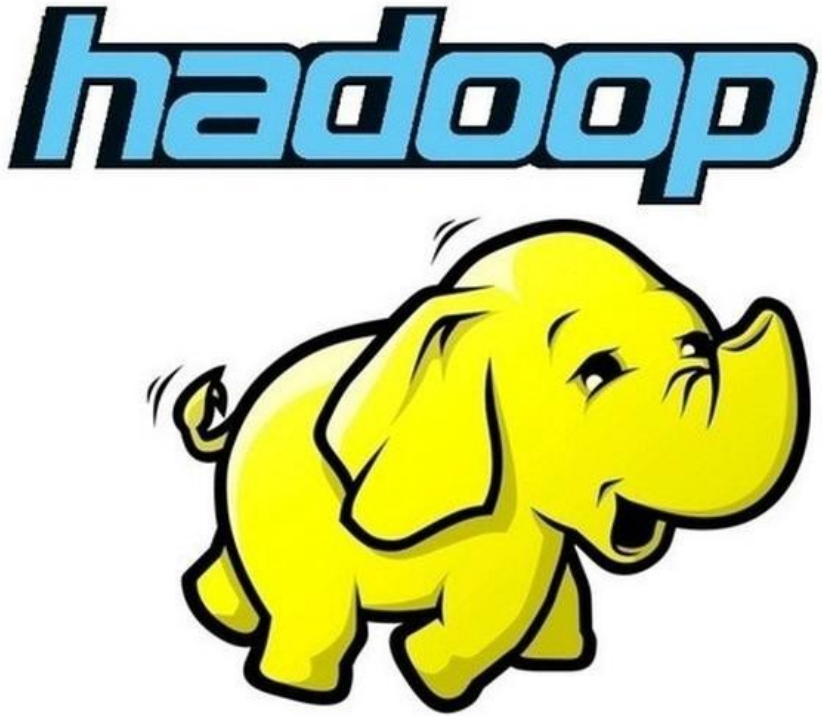
https://github.com/elek/flekszible

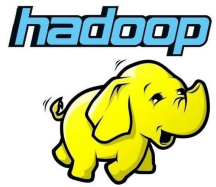
https://github.com/elek

https://flokkr.github.io

Marton Elek
elek@apache.org

DATASCIENCE


hadoop

Storage in the  world

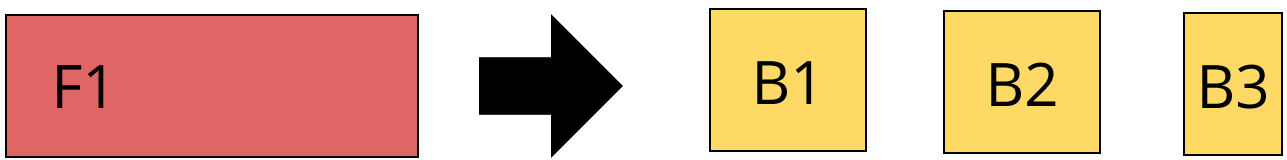
hadoop-hdfs

- 😕 Can't handle small files
- 😕 Only HadoopFS protocol is supported

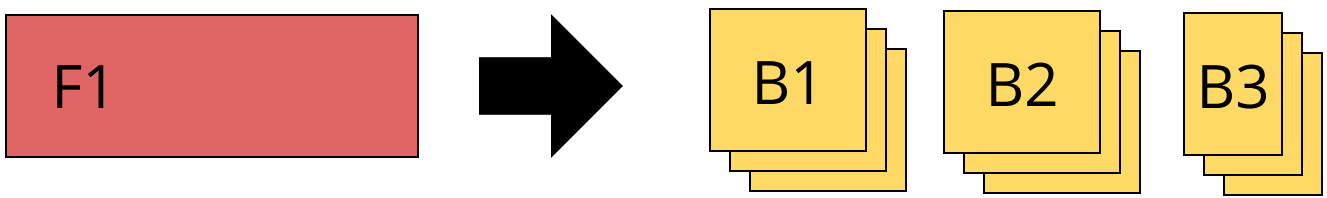
hadoop-aws/aliyun/azure

- 😕 (Eventual) Consistency
- 😕 Cost (PB scale)
- 😕 Slower (no local data)
- ☺ Easy to use / no management
- ☺ Tool support

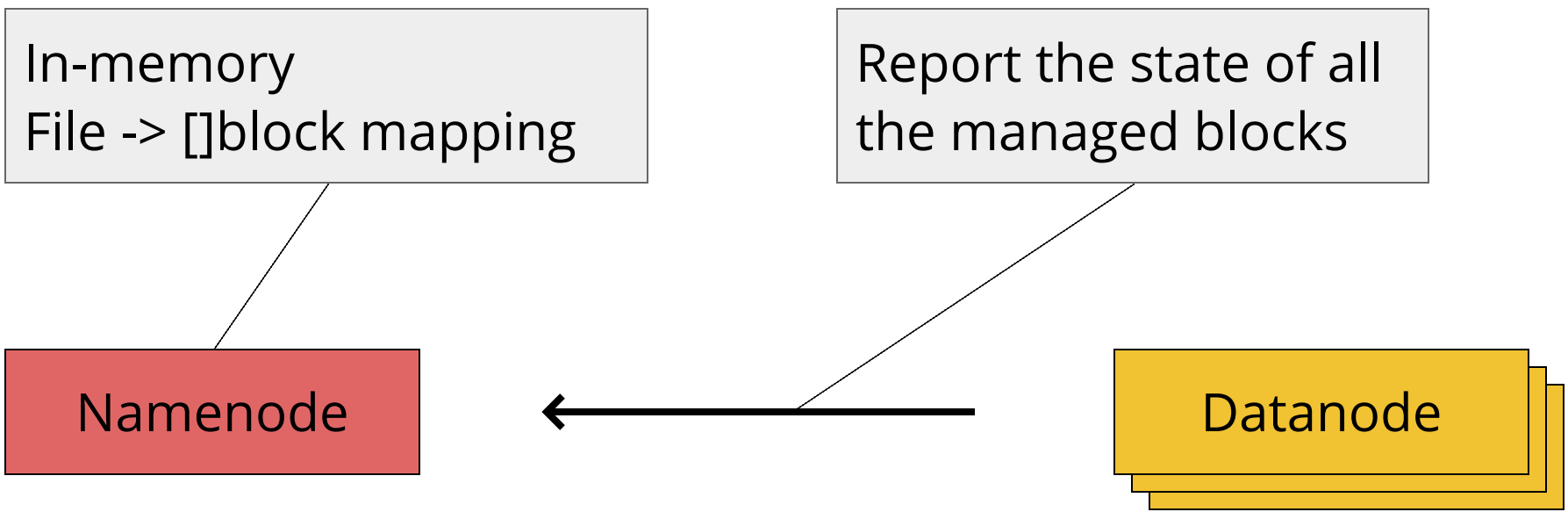
Replication: split the files

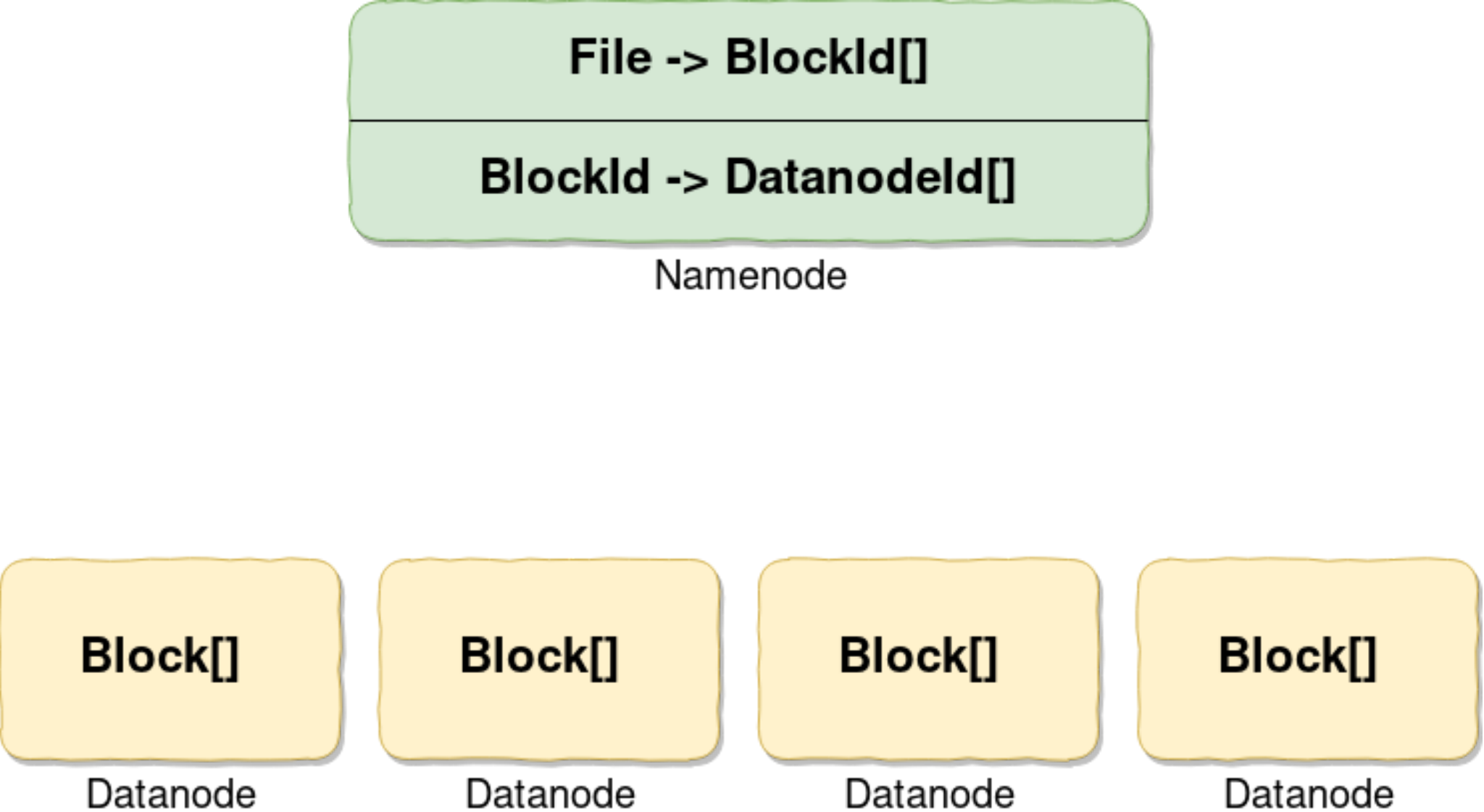


Replication: split the files



Replication: split the files




File -> BlockId[]
BlockId -> DatanodeId[]
Namenode
Block[]
Block[]
Block[]
Block[]
Datanode
Datanode
Datanode
Datanode

File -> BlockId[]

BlockId -> DatanodeId[]

Namenode

Block[]

Datanode

Block[]

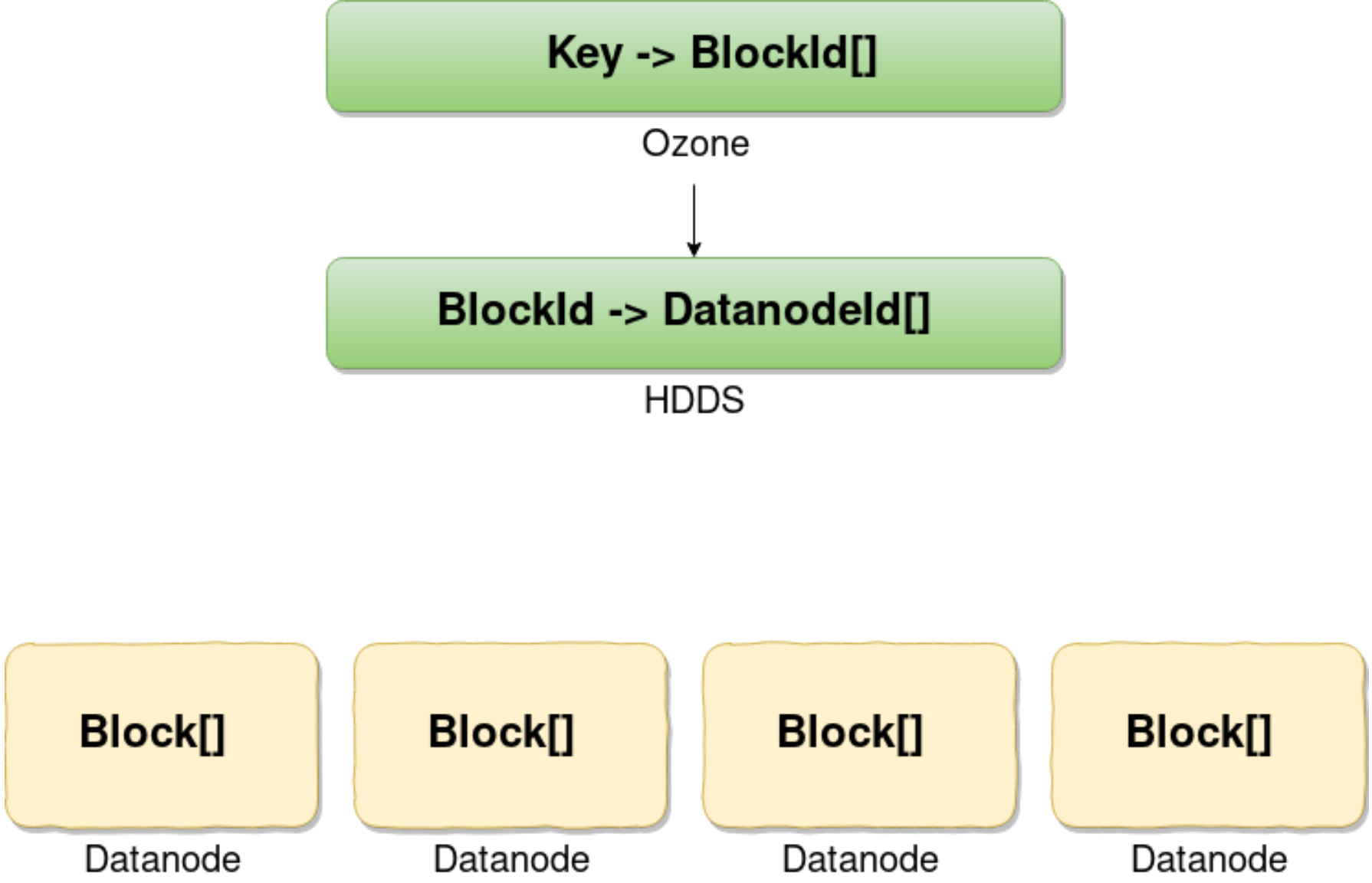
Datanode

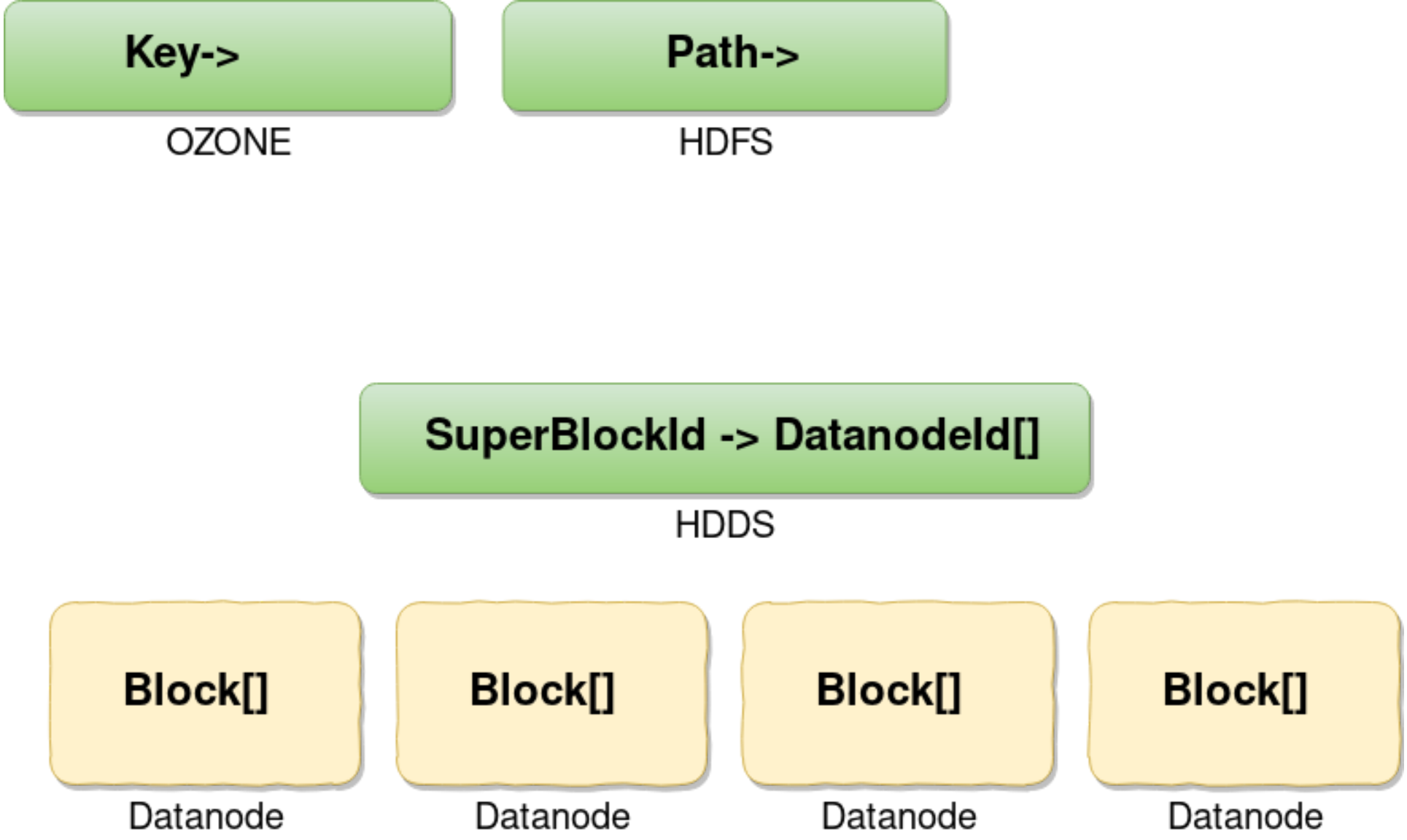
Block[]

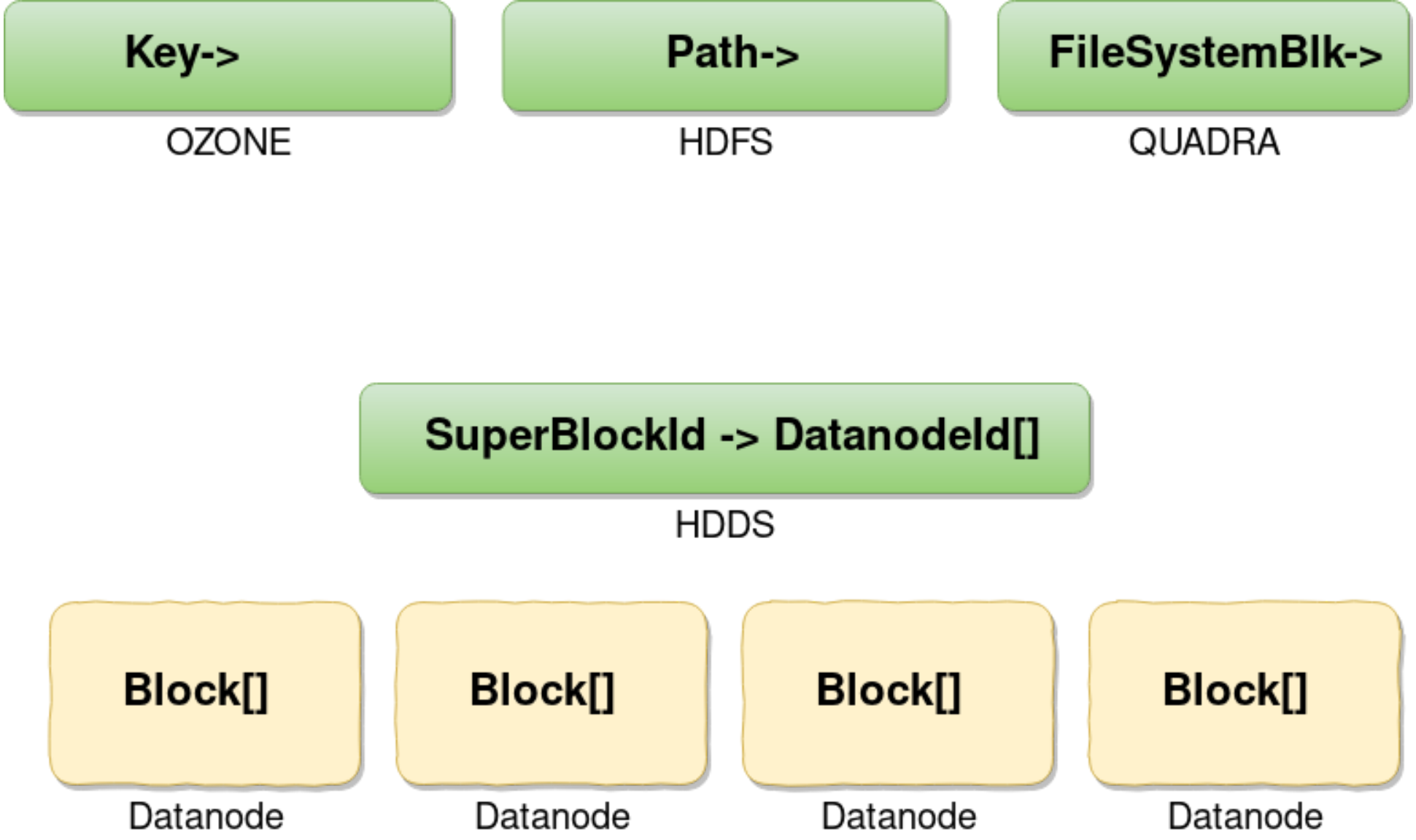
Datanode

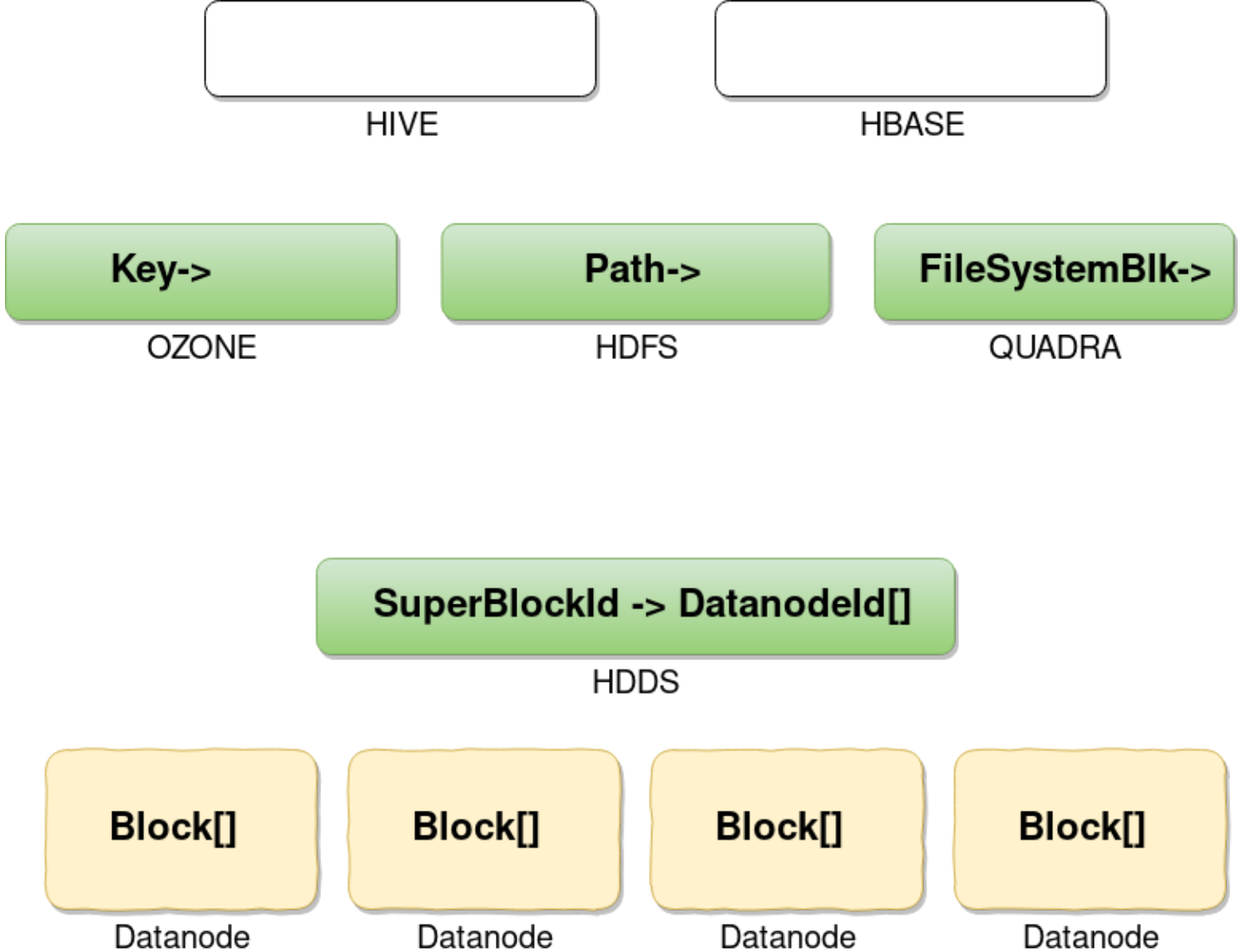
Block[]

Datanode


Key -> BlockId[]
Ozone
BlockId -> DatanodeId[]
HDDS
Block[]
Block[]
Block[]
Block[]
Datanode
Datanode
Datanode
Datanode


Key->
Path->
OZONE
HDFS
SuperBlockId -> DatanodeId[]
HDDS
Block[]
Block[]
Block[]
Block[]
Datanode
Datanode
Datanode
Datanode


Key->
Path->
FileSystemBlk->
OZONE
HDFS
QUADRA
SuperBlockId -> DatanodeId[]
HDDS
Block[]
Block[]
Block[]
Block[]
Datanode
Datanode
Datanode
Datanode


HIVE
HBASE
Key->
Path->
FileSystemBlk->
OZONE
HDFS
QUADRA
SuperBlockId -> DatanodeId[]
HDDS
Block[]
Block[]
Block[]
Block[]
Datanode
Datanode
Datanode
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)

- Consistent
- Fast
- Cloud native
- Easy to use

DATASCIENCE

DATASCIENCE

Copysets

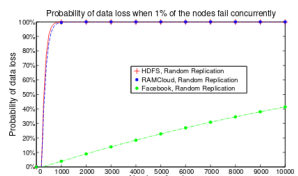
Copysets: Reducing the Frequency of Data Loss in Cloud Storage

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 Copyset 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 Cidon[1], Robert Escriva[2], Sachin Katti[1], Mendel Rosenblum[1], and Emin Gün Sirer[2]

[1]Stanford University
[2]Cornell 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-

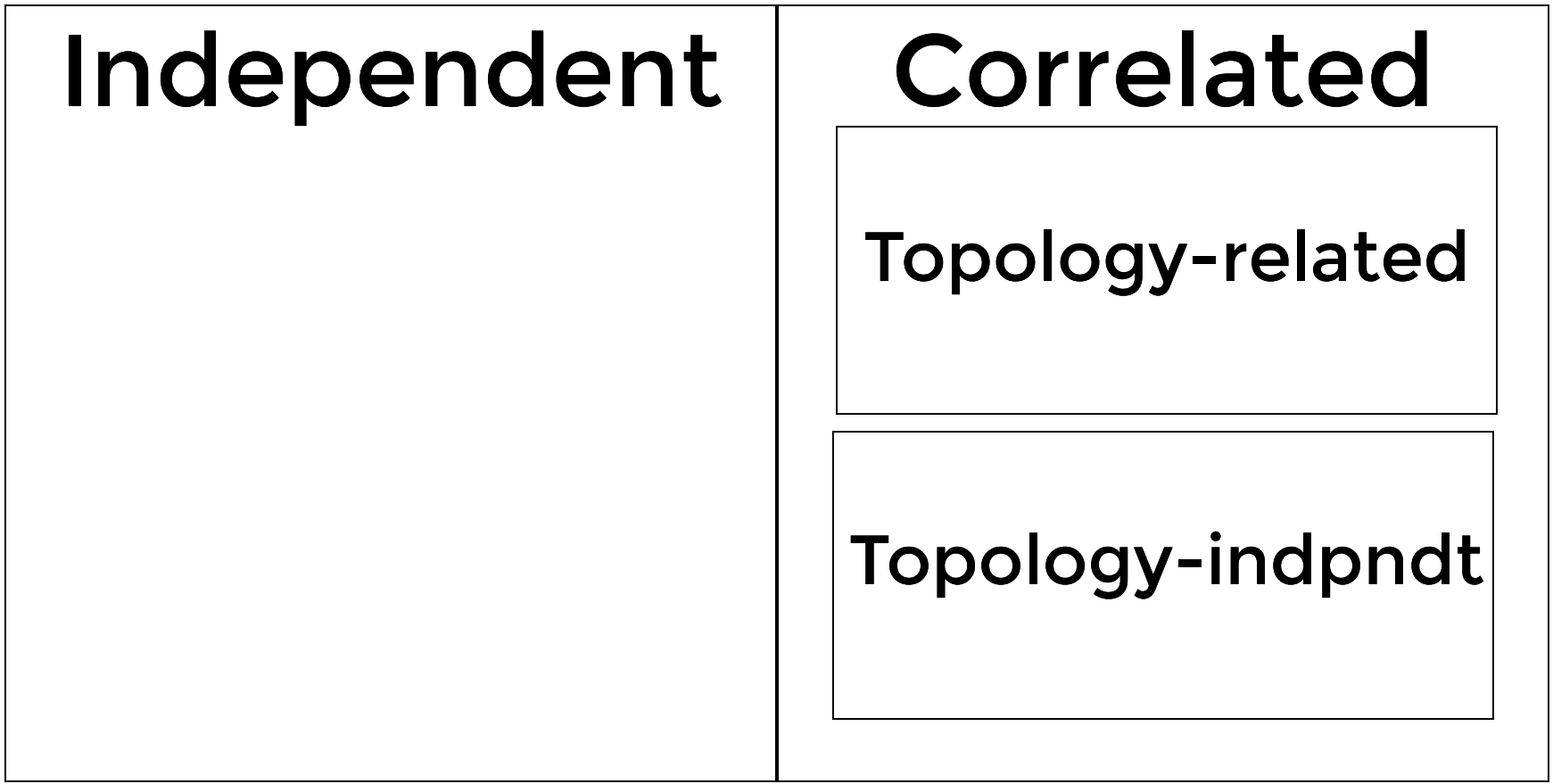
GFS [15] and Azure [6] typically replicate their data on three random machines to guard against data loss within a single cluster, and geo-replicate the entire cluster to a separate location to guard against correlated failures.

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.,

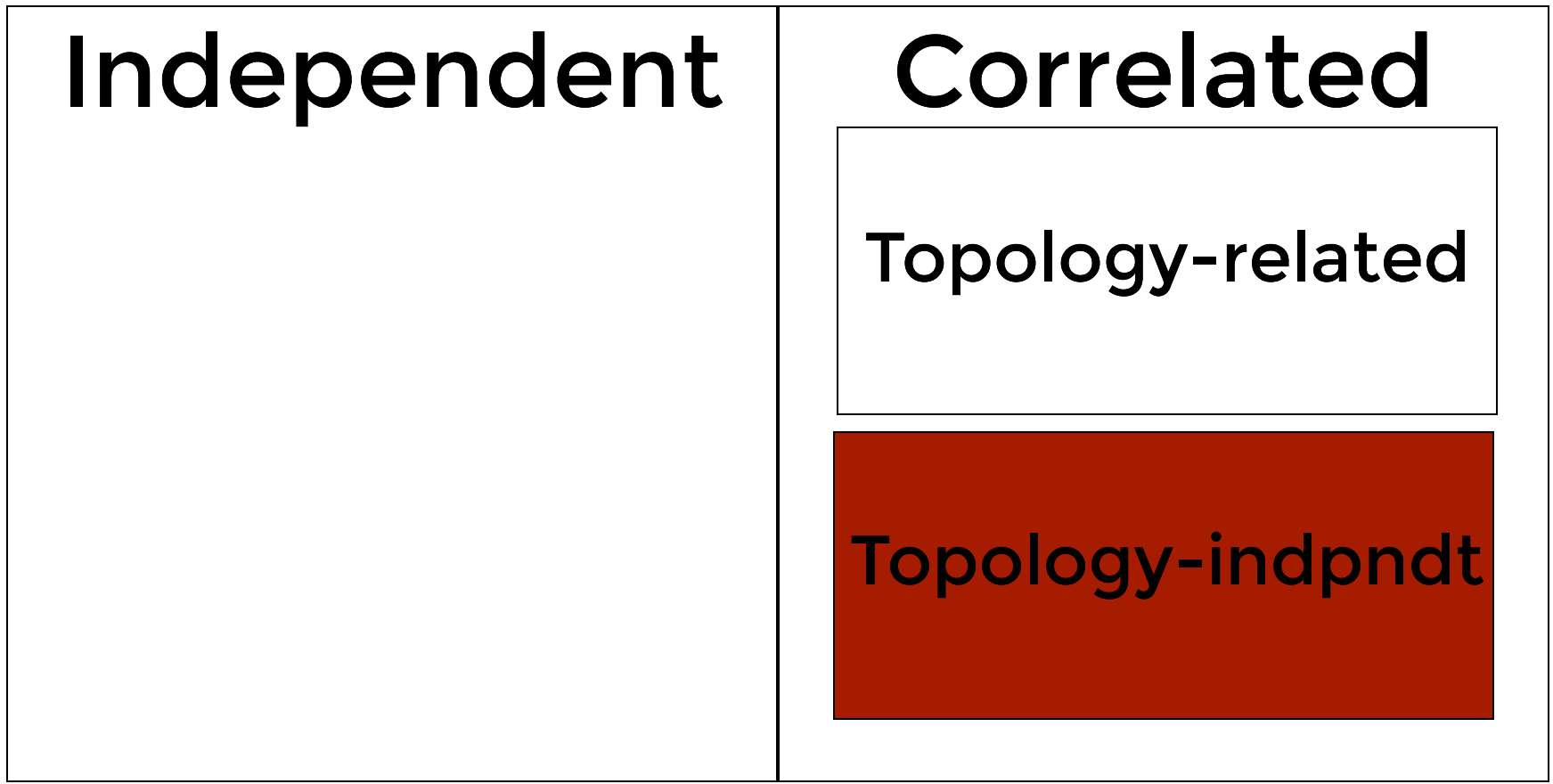
https://www.usenix.org/system/files/conference/atc15/atc15-paper-cidon.pdf

What could go wrong?

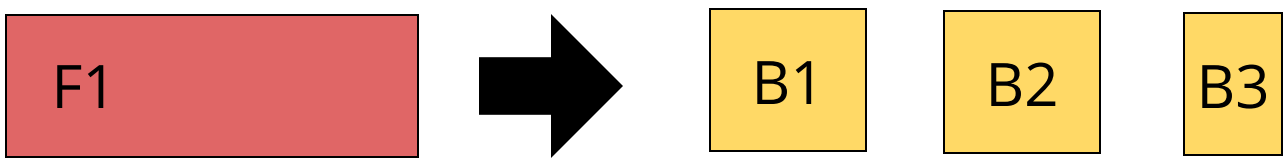
Node failures



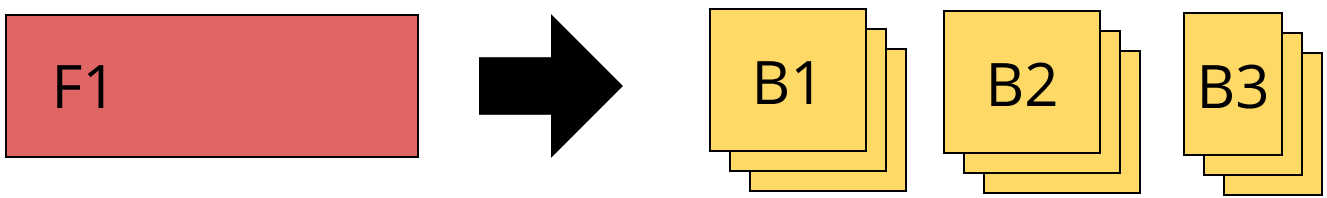
Node failures



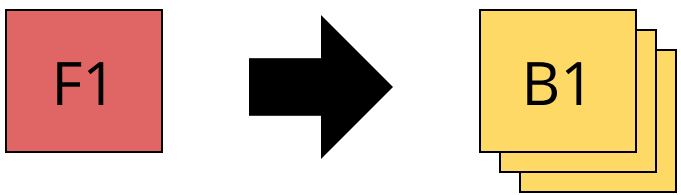
Replication: split the files



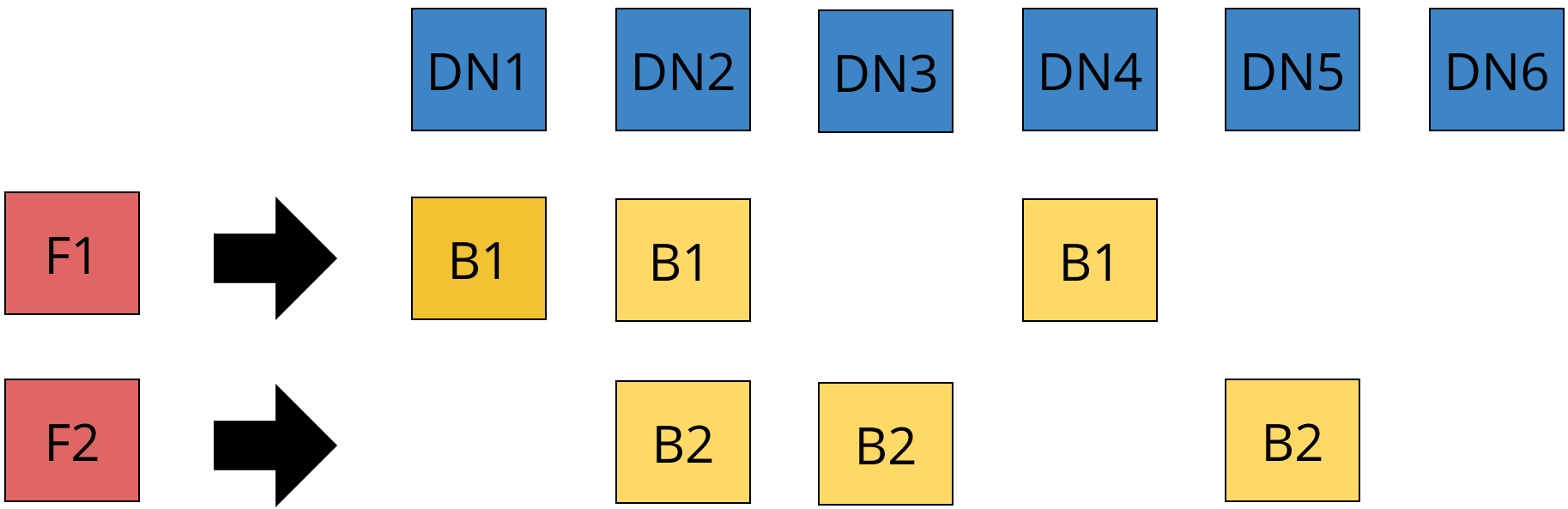
Replication: split the files



Replication: split the files



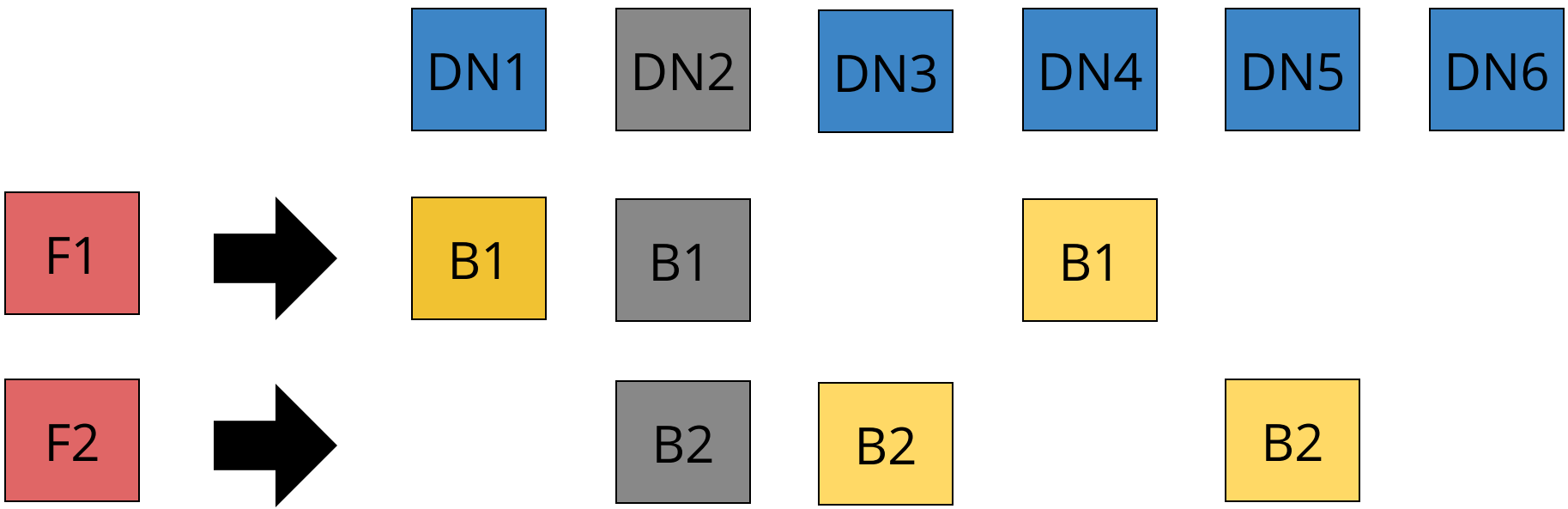
Replication



Kill one Datanode

Datanode #2

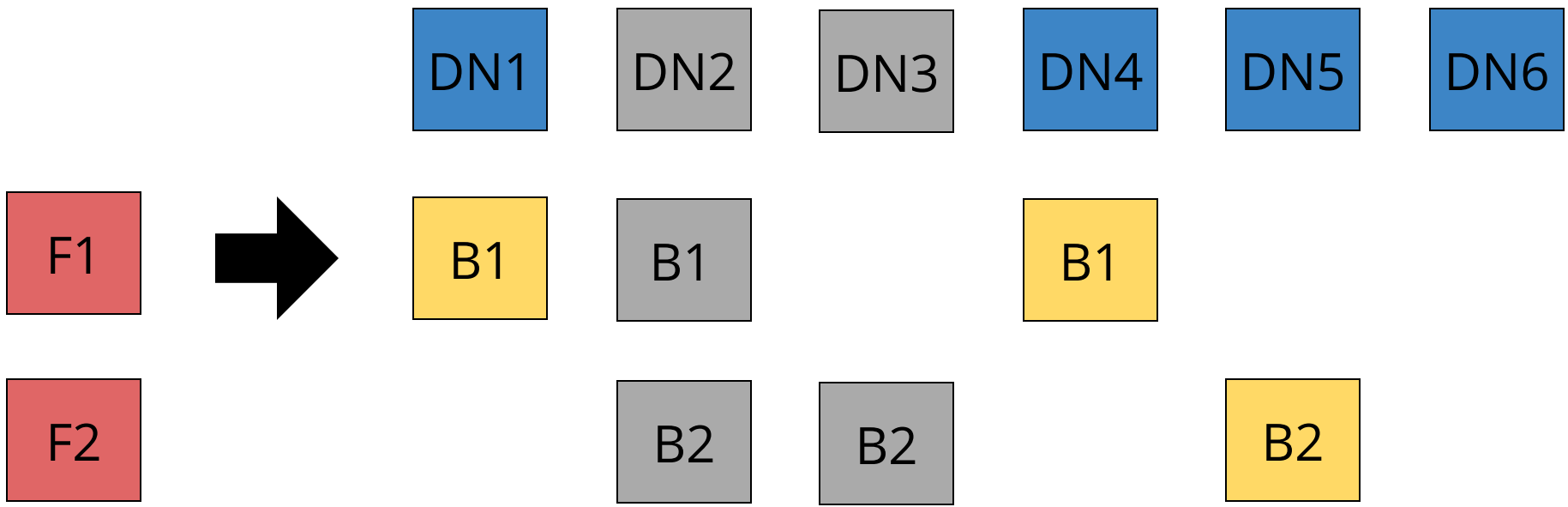
Replication



Kill second Datanode

Datanode #3

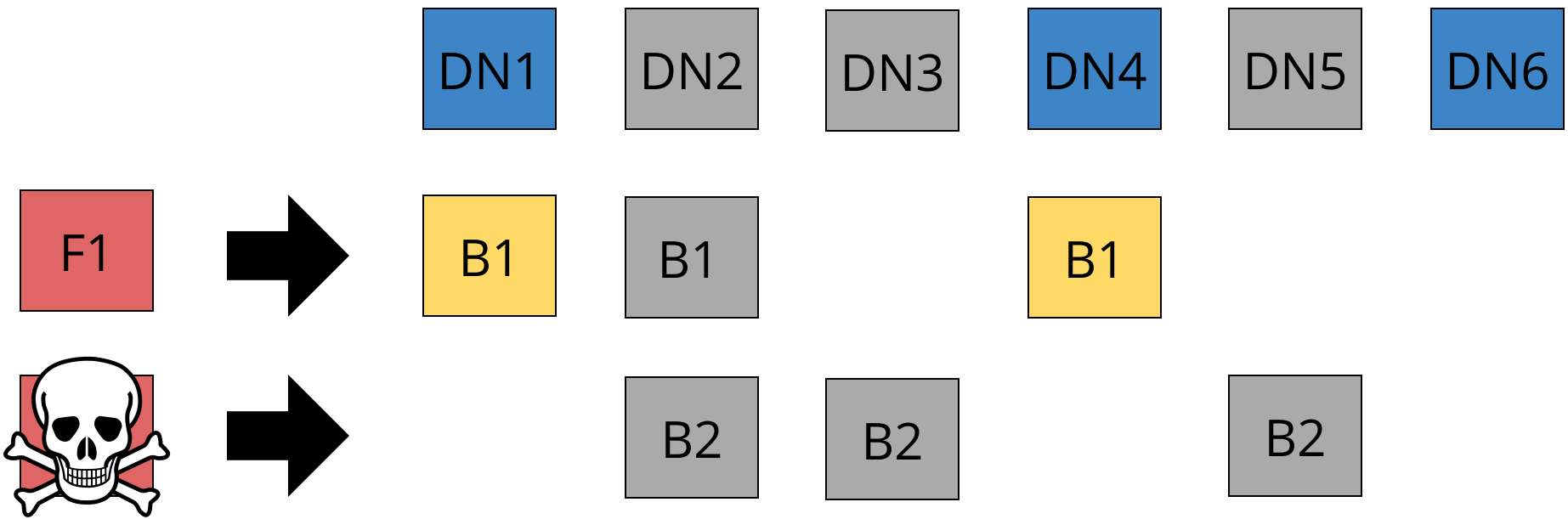
Replication: split the files



Kill third Datanode

Datanode #5

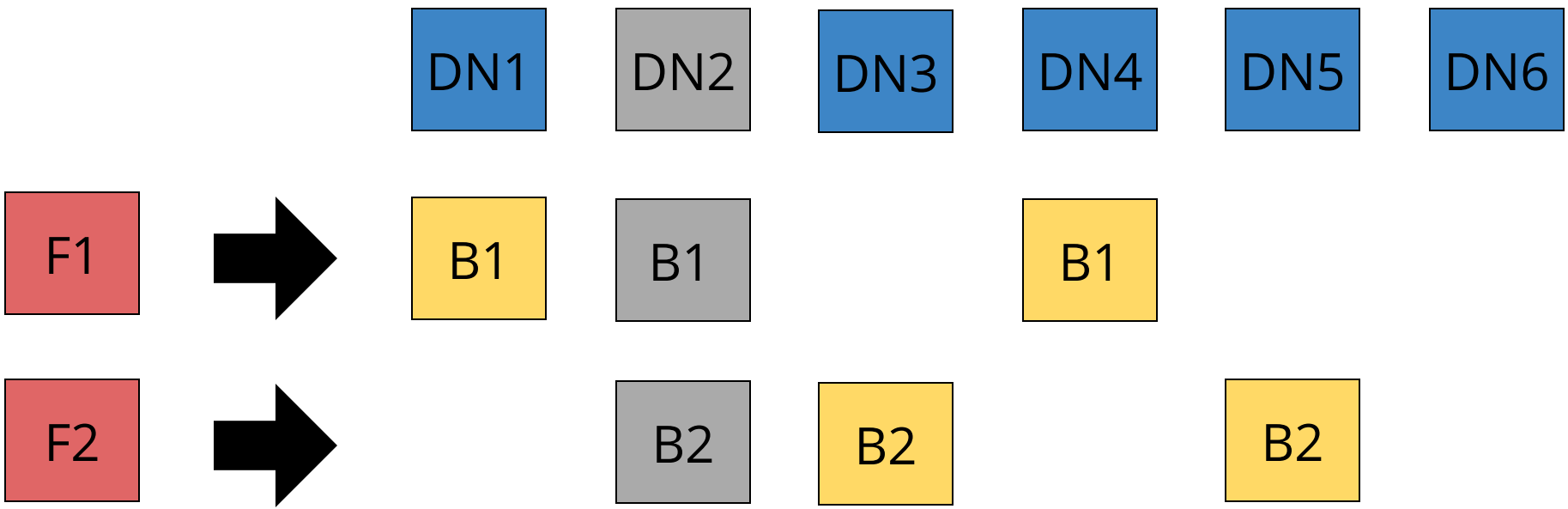
Replication: split the files

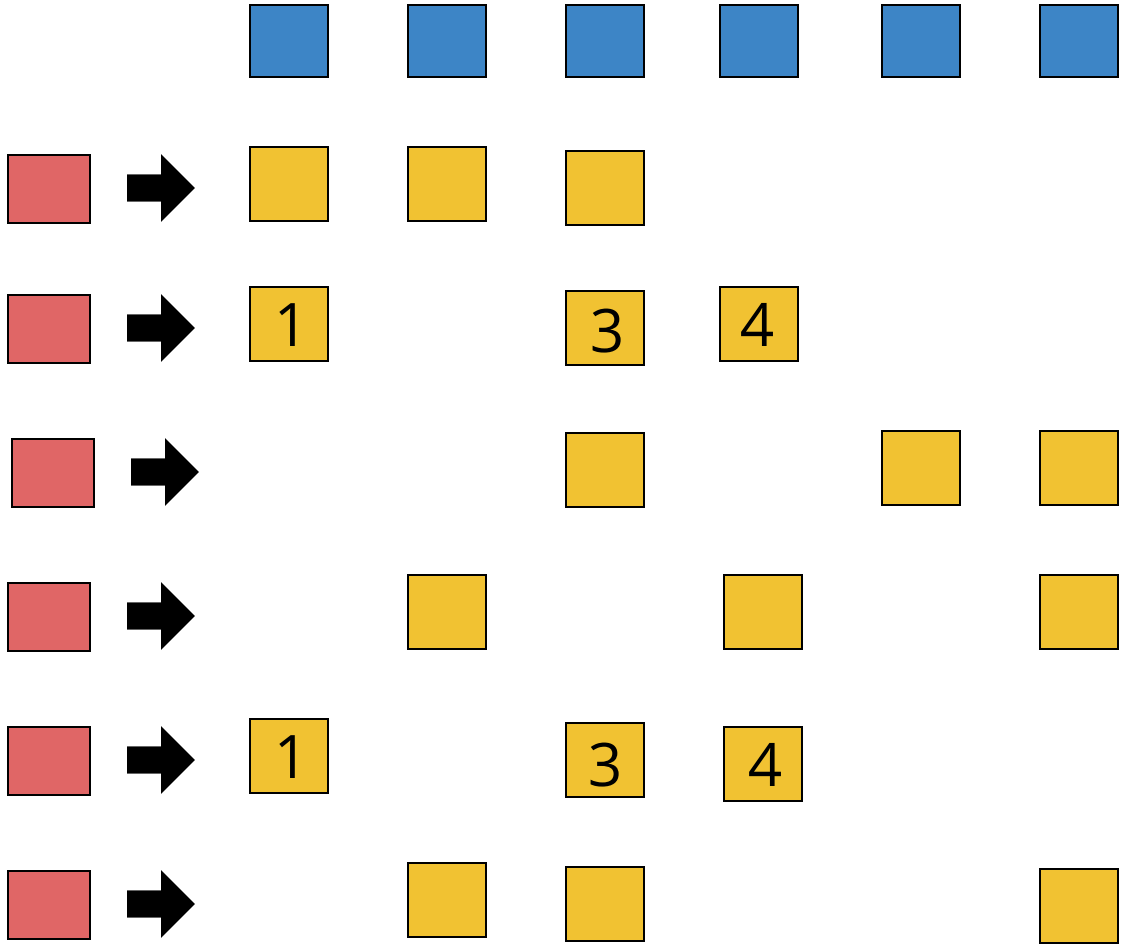


Does it help

if I use more datanodes?

Replication




1
3
4
1
3
4

Random replication

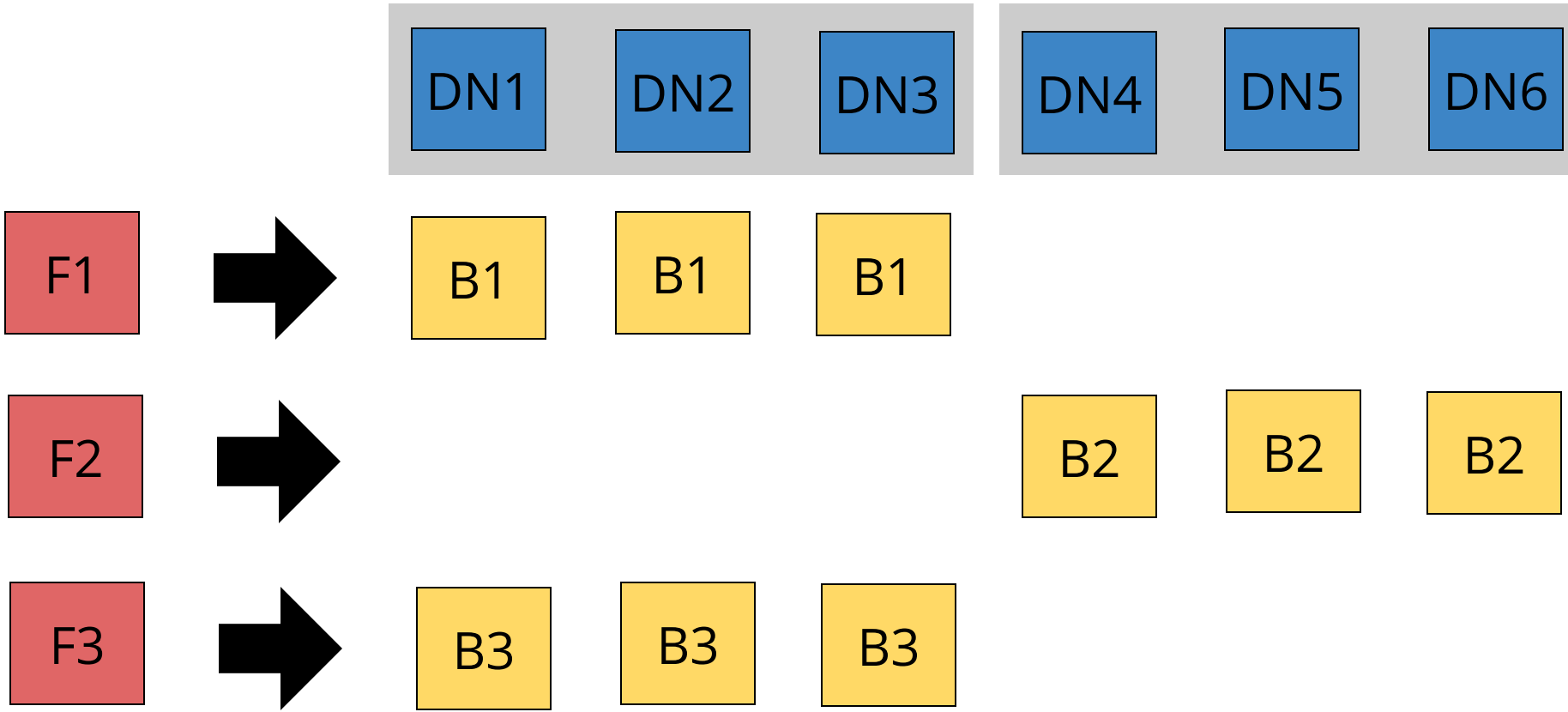
- 6 datanodes --> 20 different 3-node set: $\binom{6}{3} = 20$
  - [1 3 4][1 2 3] [2 5 7] [6 8 9] ...
- 2000 blocks (20 * 100)
  - [1 2 3] --> ~100 blocks
  - [1 2 4] --> ~100 blocks
  - [2 5 7] --> ~100 blocks

Scale it up?

- 600 datanodes --> 600!/597!/3! different 3-node set:
  - 100 * 599 * 598 = 35 820 200

$$\binom{600}{3} = 35820200$$

- 3 500 000 000 -> blocks
  - [1 2 3] --> ~100 blocks
  - [1 2 4] --> ~100 blocks
  - [2 5 7] --> ~100 blocks


DN1
DN2
DN3
DN4
DN5
DN6
F1
F2
F3
B1
B1
B1
B2
B2
B2
B3
B3
B3

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

$$\binom{6}{3} = 20$$

Kill 3 datanodes randomly

- [2 5 6] -> no data loss

$$\binom{6}{3} = 20$$

Kill 3 datanodes randomly

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

$$\binom{6}{3} = 20$$

Kill 3 datanodes randomly

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

$$\binom{6}{3} = 20$$

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

$$\binom{6}{3} = 20$$

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
  - [1 3 6], [4 7 9], ...
- 2000 -> blocks
  - [1 2 3] --> ~100 blocks
  - [4 5 6] --> ~100 blocks
- 3 DN failure --> dataloss: 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]

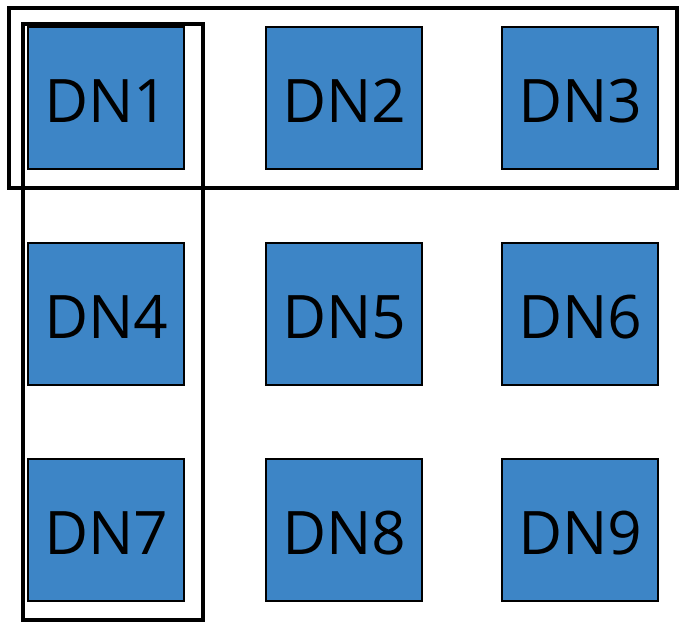
...

[2 groups] recovery

Only one failure (datanode #1)

[4 5 6] -> ok

[(1) 2 3] -> 1000 blocks should be replicated from [2 3]


DN1
DN2
DN3
DN4
DN5
DN6
DN7
DN8
DN9

6 groups: 3 col, 3 row

3 random node failures:

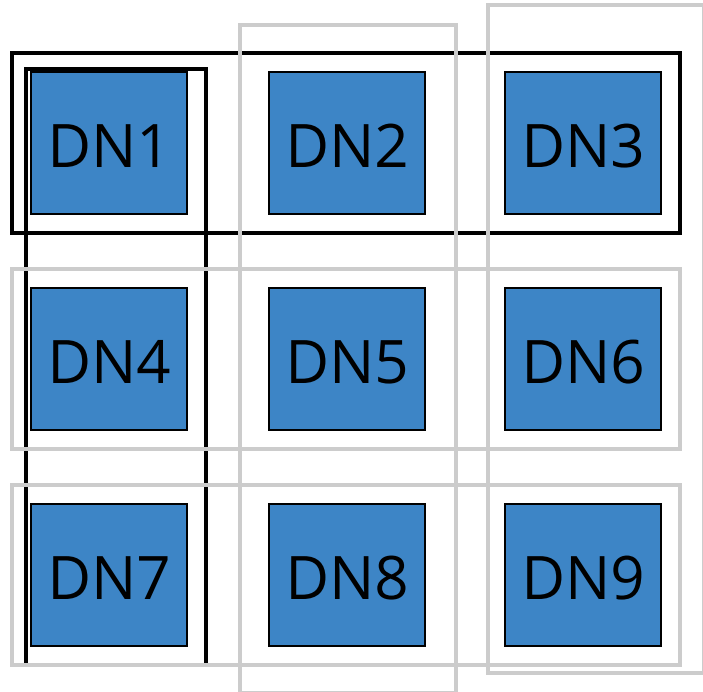
possible in 84 ways

Chance 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 (Copysets)

- 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