# Apache Hadoop Ozone

# How to loose data

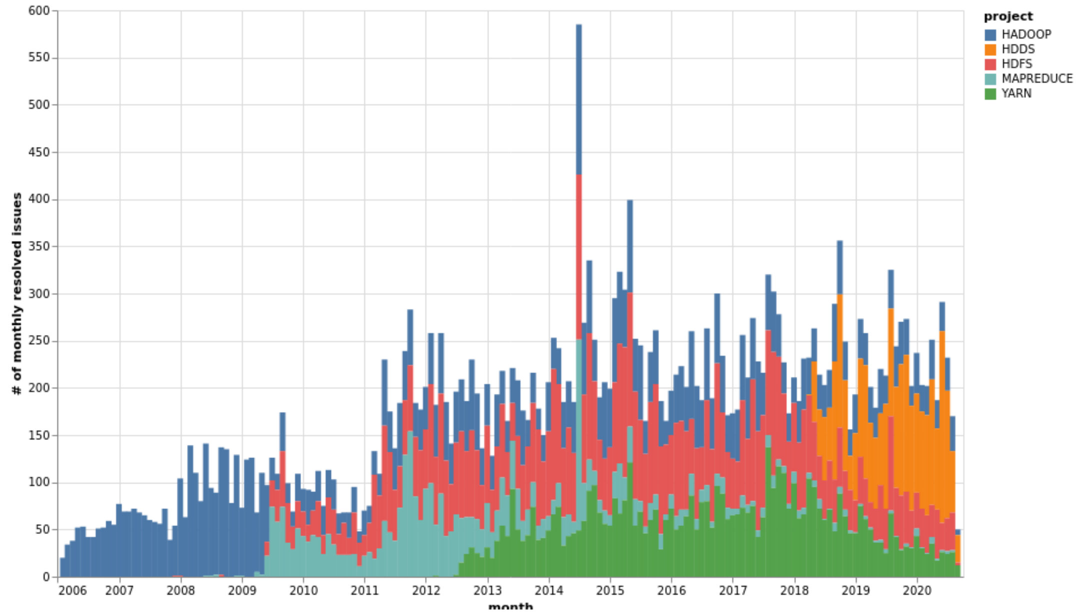
## In a safe way

# DATA+SCIENCE

DATA+SCIENCE

# Apache Hadoop

- Since 2006
- Big-data for everyone: on commodity hardware
- Storage: HDFS
- Still popular and used



# World has changed (2020)

- Scalability? Small files problem?
  - 2006: Hadoop is designed for Huge files
  - 2020: Streaming → small files → ?
- Usability
  - Hadoop is an ecosystem (Spark, Hive, Flink...)
  - How to use it from ML? From Python?

S3 protocol

Hadoop FS

CSI

# Apache Hadoop Ozone

hadoop.apache.org/ozone

THE APACHE SOFTWARE FOUNDATION

DATA+SCIENCE

# Copysets + Tiered replications

## 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-purpose replication technique that significantly reduces the frequency of data loss events. We implemented and evaluated Copyset Replication on two open source data center storage systems, HDFS and RAMCloud, and show it incurs a low overhead on all operations. Such systems require that each node's data be scattered across several nodes for parallel data recovery and access. Copyset Replication presents a near optimal tradeoff between the number of nodes on which the data is scattered and the probability of data loss. For example, in a 5000-node RAMCloud cluster under a power outage, Copyset Replication reduces the probability of data loss from 99.99% to 0.15%. For Facebook's HDFS cluster, it reduces the probability from 22.8% to 0.78%.

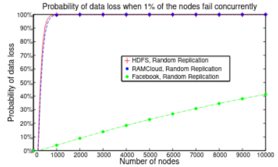



**Figure 1: Computed probability of data loss with $R = 3$ when 1% of the nodes do not survive a power outage. The parameters are based on publicly available sources [5, 7, 24, 25] (see Table 1).**

(0.5%-1%) [7, 25] do not come back to life after power has been restored. When a large number of nodes do not power up there is a high probability that all replicas of at least one chunk in the system will not be available.

Figure 1 shows that once the size of the cluster scales beyond 300 nodes, this scenario is nearly guaranteed to cause a data loss event in some of these systems. Such data loss events have been documented in practice by Ya-

## 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 protecting against correlated failures.

We present Tiered Replication, a replication scheme that splits the cluster into a primary and backup tier. The first two replicas are stored on the primary tier and are used to recover data in the case of independent node failures, while the third replica is stored on the backup tier and is used to protect against correlated failures. The key insight of our paper is that, since the third replicas are rarely read, we can place the backup tier on separate physical infrastructure or a remote location without affecting performance. This separation significantly increases the resilience of the storage system to correlated failures and presents a low cost alternative to geo-replication of an entire cluster. In addition, the Tiered

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., individual disk failure, kernel crash). Correlated failures are defined as failures in which several nodes fail simultaneously due to a common root cause [7, 11] (e.g., network failure, power outage, software upgrade). In this paper, we are focused on events that affect data durability rather than data availability, and are therefore concerned with node failures that cause permanent data loss, such as hardware and disk failures, in contrast to transient data availability events, such as software upgrades.

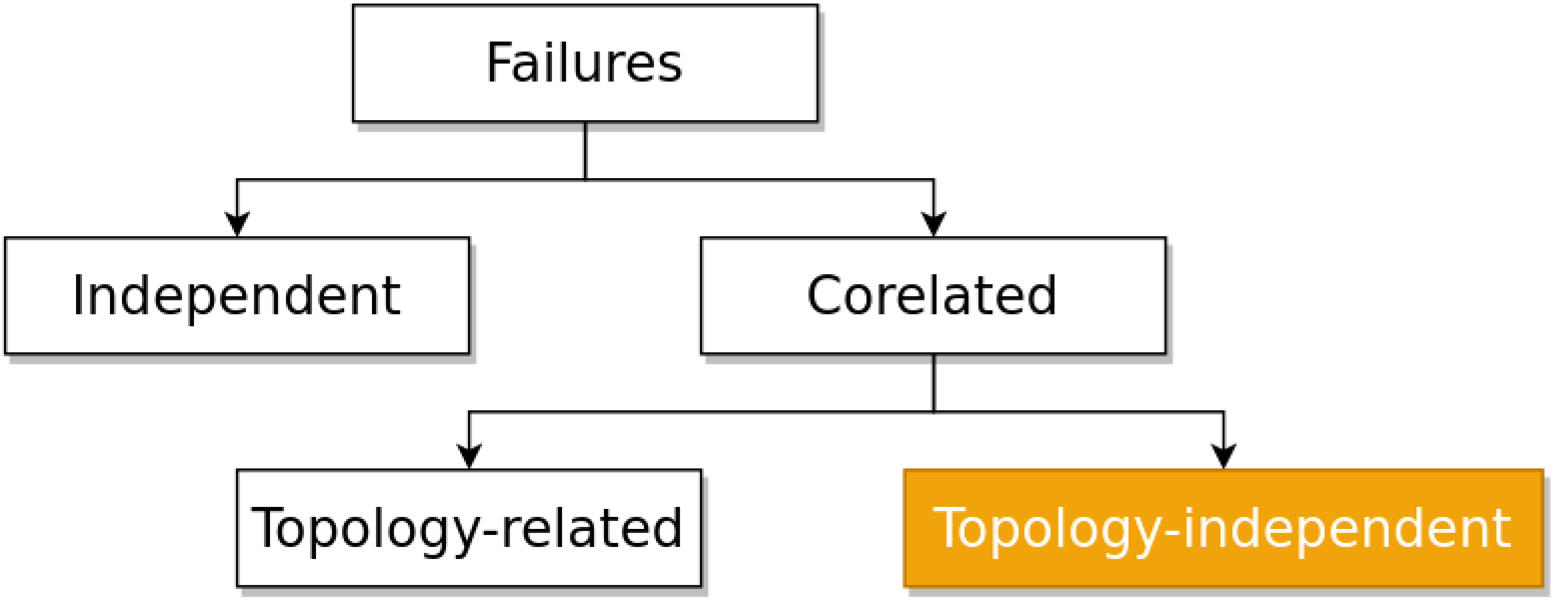
The conventional wisdom is that three-way replication is cost-effective for guarding against node failures within a cluster. We also note that, in many storage systems, the third replica was introduced mainly for durability and not for read performance [7, 8, 13, 34].

# Copy sets + Tiered replications

- https://web.stanford.edu/~skatti/pubs/usenix13-copysets.pdf
- https://www.usenix.org/system/files/conference/atc15/atc15-paper-cidon.pdf

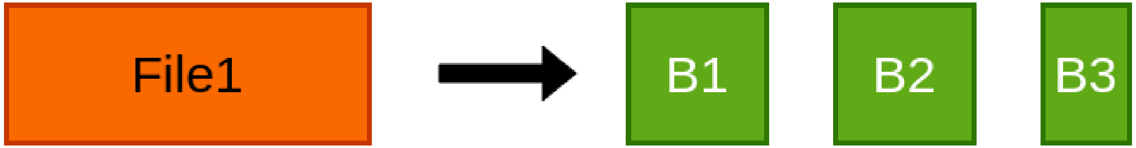
# What can go wrong?

# Failure types

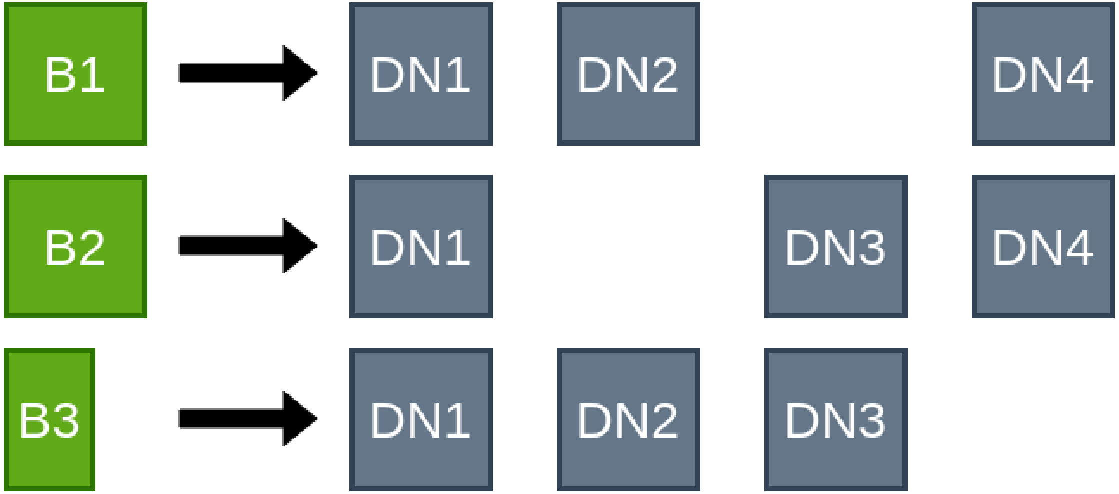


# How to Store files?

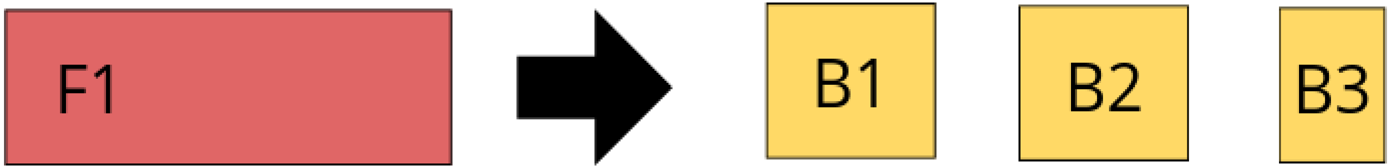
Split file to blocks



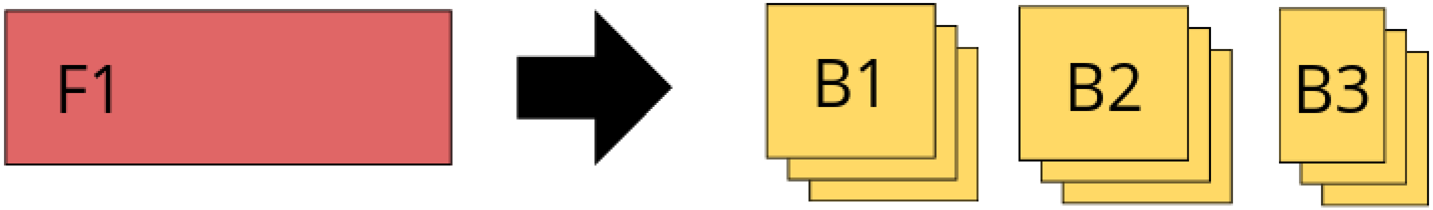

Store replicas of blocks on Datanodes



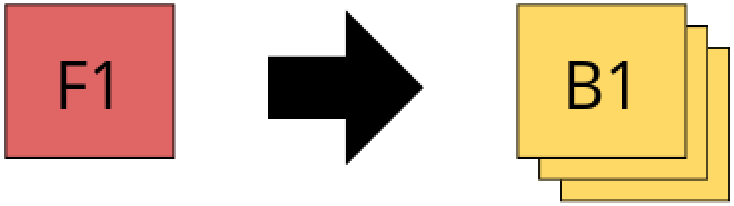
# Replication: split the files



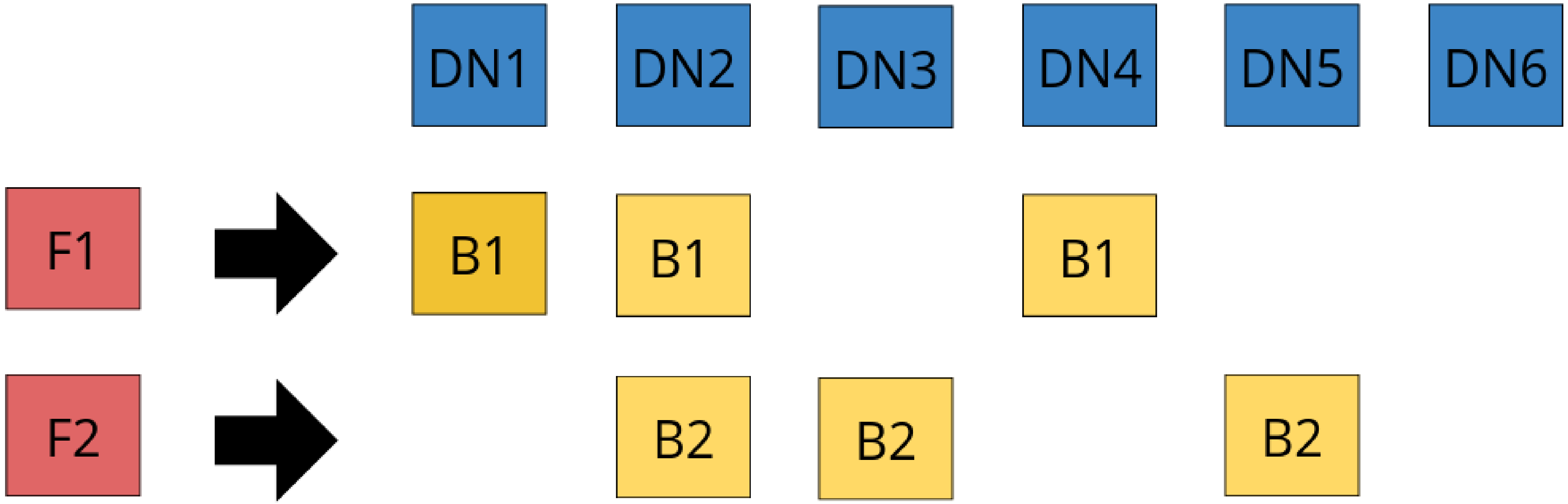
# Replication: split the files



# Replication: split the files

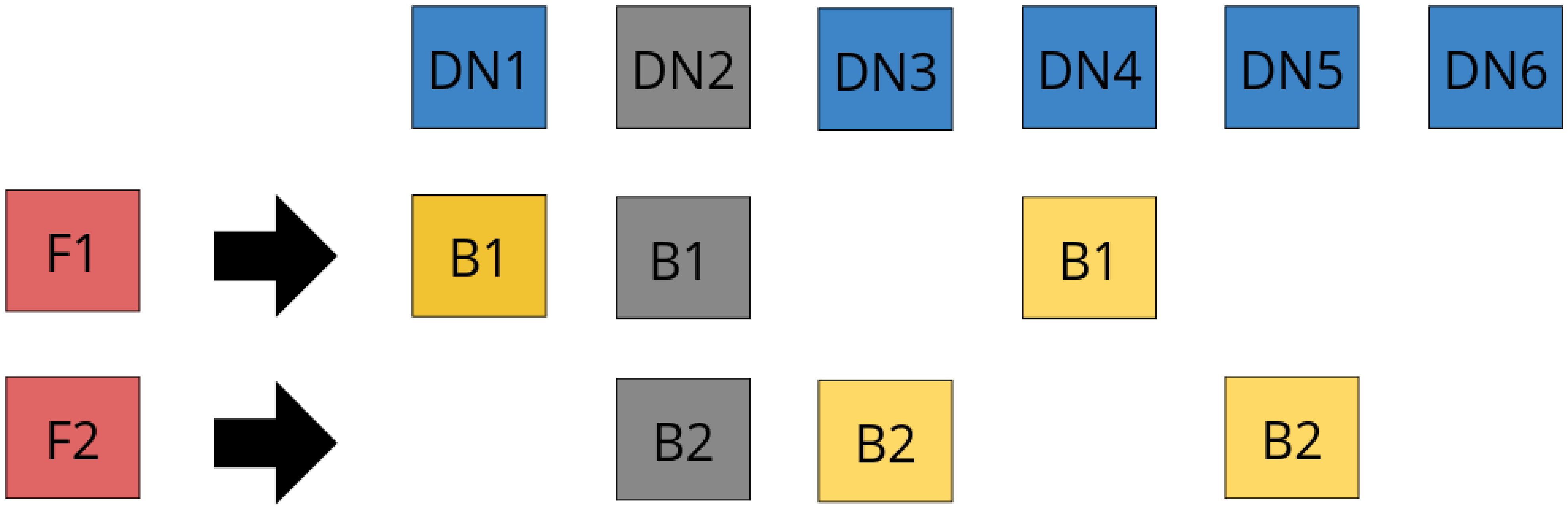


# Replicate 2 files → RANDOM

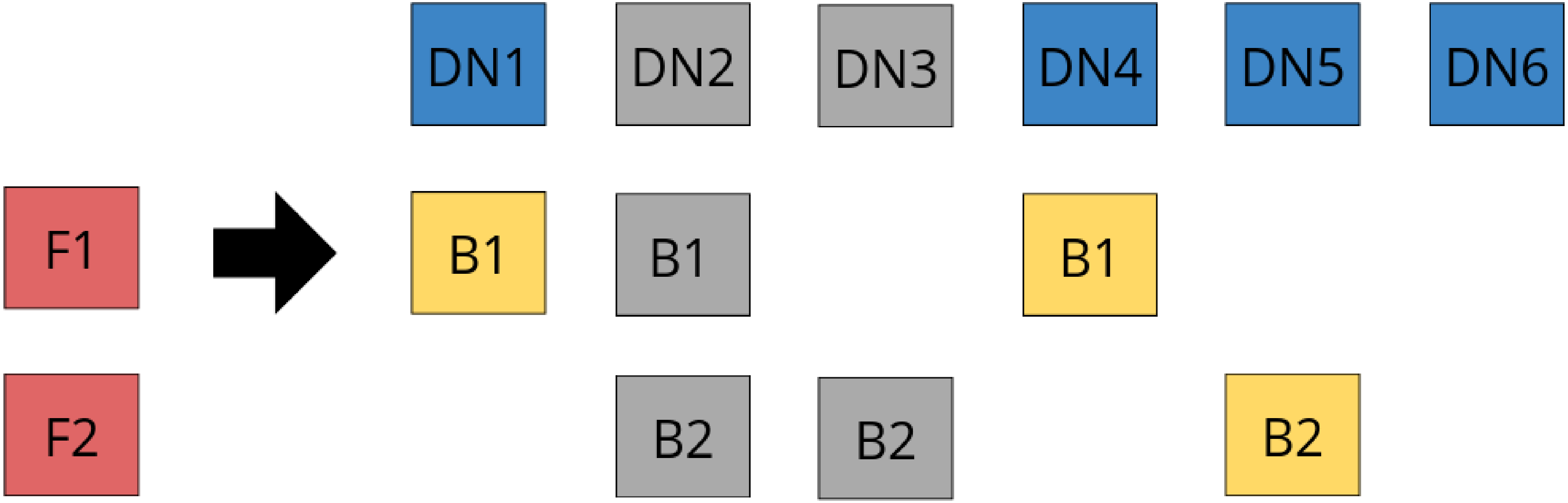


# Random failures

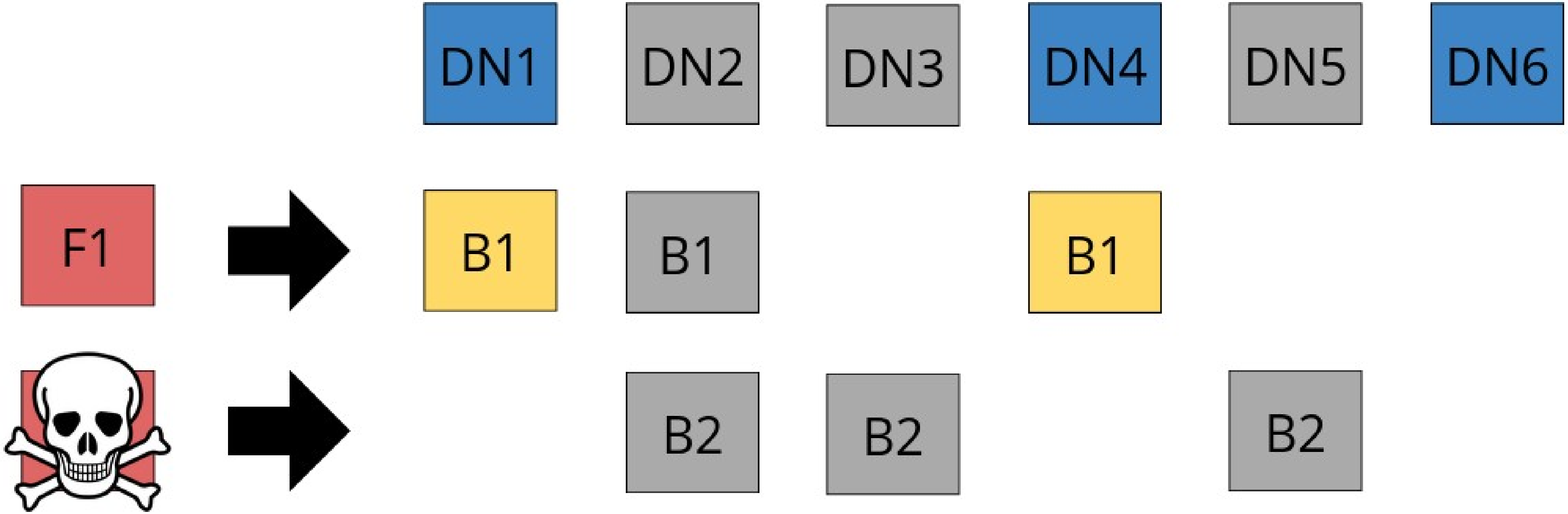
# Datanode 2 → failed



# Datanode 2 → failed



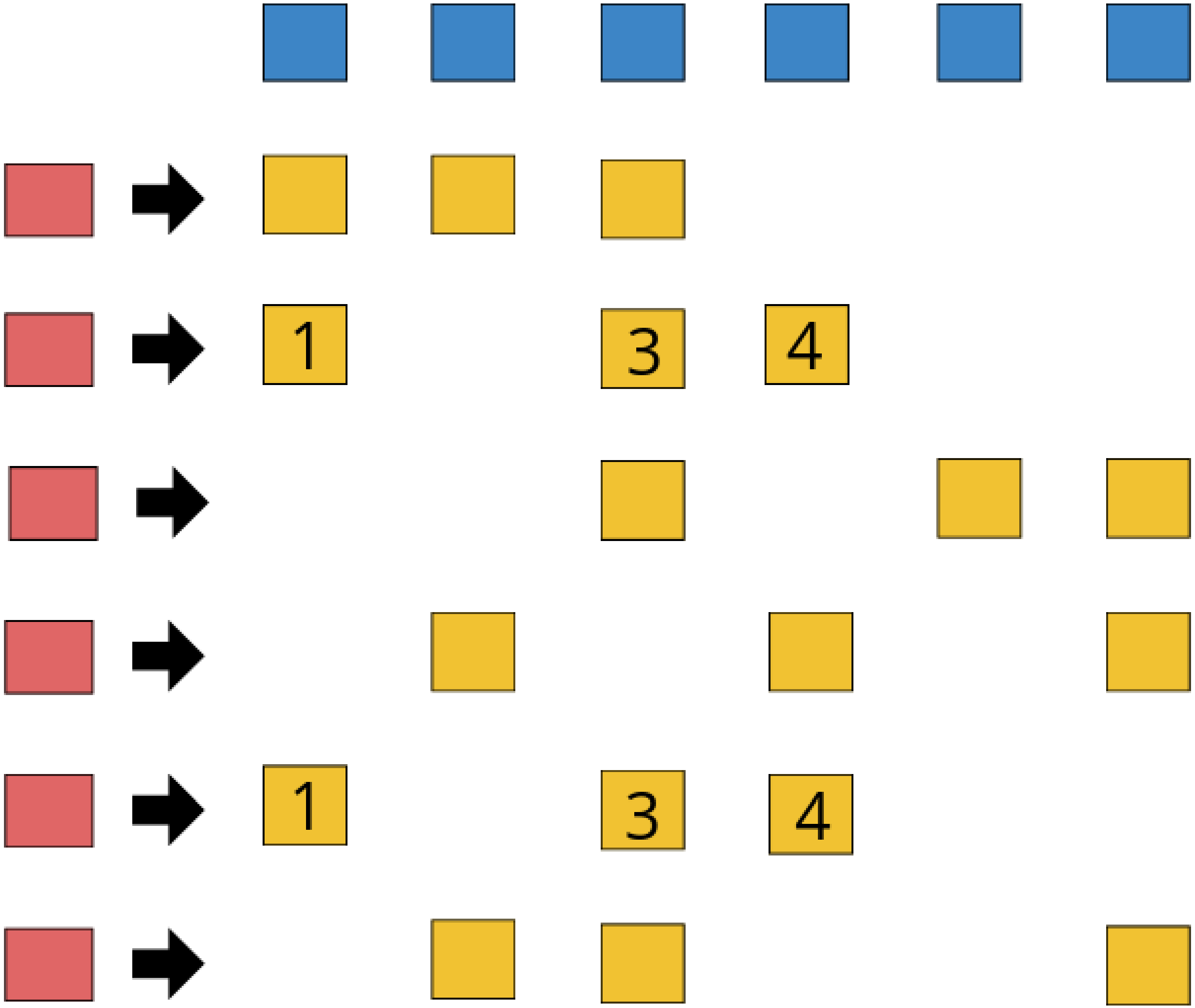
# Datanode 5 → failed



# Buy more hardware?

# Math: random replication

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


1
3
4
1
3
4

# Kill 3 datanode randomly

- [1 2 3] → data loss (100 blocks)

## Math: scale it up

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

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

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

# Kill 3 datanode randomly

- [1 2 3] → data loss (100 blocks)
- [3 5 6] → data loss (100 blocks)

## Math: scale it up

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

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

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

# Kill 3 datanode randomly

- [1 2 3] → data loss (100 blocks)
- [3 5 6] → data loss (100 blocks)
- [2 4 6] → data loss (100 blocks)

## Math: scale it up

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

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

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

# Math: 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
  - ...
  - [234 248 652] → ~100 blocks

# Kill 3 datanode randomly

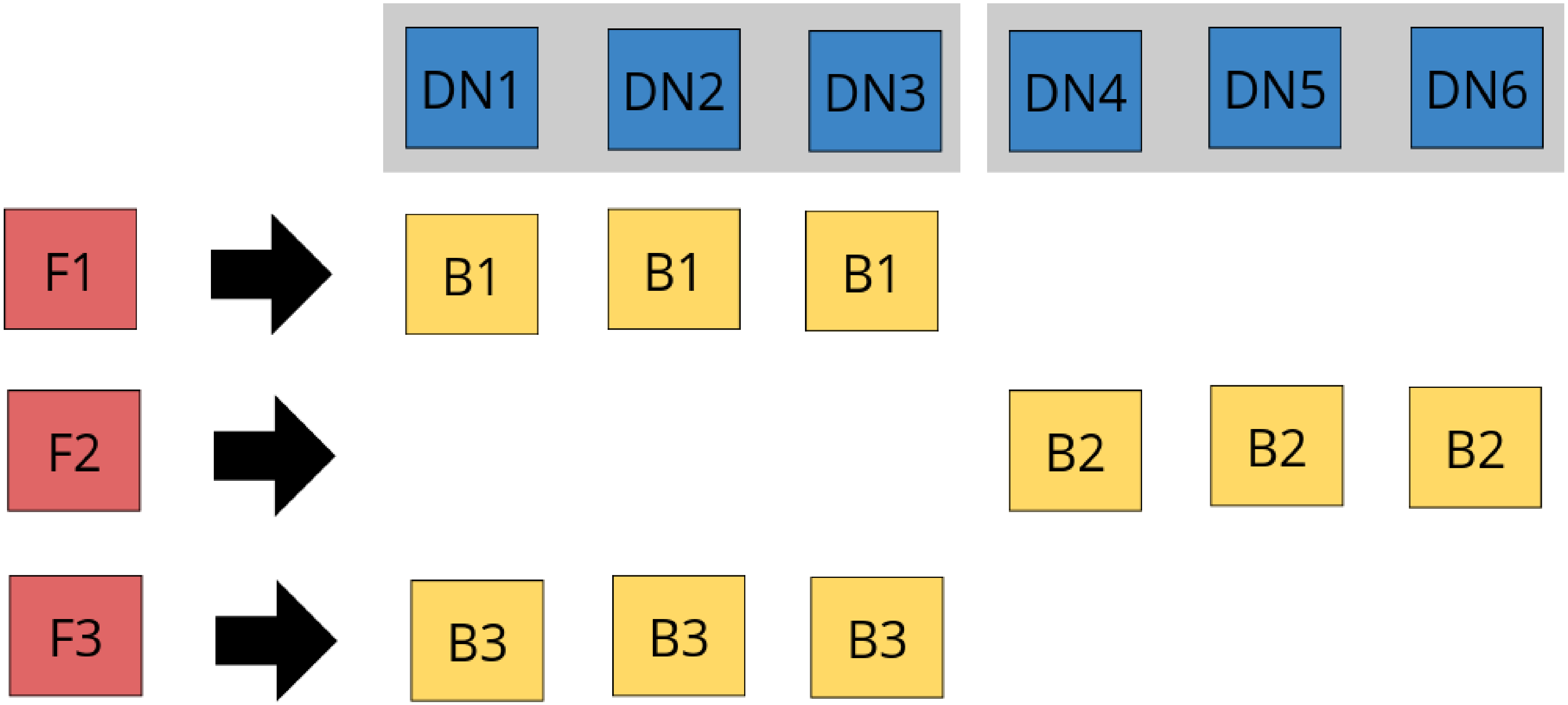
- [1 2 3] → data loss (100 blocks)
- [23 3125 236] → data loss (100 blocks)
- [232 234 126] → data loss (100 blocks)

### Math: 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
  - ...

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

# Let's try sg different


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

# Math: 2 group replication

- 6 datanodes → 2 different 3-node set:
  - [1 2 3] [4 5 6]
- 2000 blocks
  - [1 2 3] → ~1000 blocks
  - [4 5 6] → ~1000 blocks

# Kill 3 datanode randomly

- [3 5 6] → no problem

## Math: 2 group replication

- 6 datanodes → 2 different 3-node set:
  - [1 2 3] [4 5 6]
- 2000 blocks
  - [1 2 3] → ~1000 blocks
  - [4 5 6] → ~1000 blocks

# Kill 3 datanode randomly

- [3 5 6] → no problem
- [2 4 6] → no problem

## Math: 2 group replication

- 6 datanodes → 2 different 3-node set:
  - [1 2 3] [4 5 6]
- 2000 blocks
  - [1 2 3] → ~1000 blocks
  - [4 5 6] → ~1000 blocks

# Kill 3 datanode randomly

- [3 5 6] → no problem
- [2 4 6] → no problem
- [1 2 4] → no problem

## Math: 2 group replication

- 6 datanodes → 2 different 3-node set:
  - [1 2 3] [4 5 6]
- 2000 blocks
  - [1 2 3] → ~1000 blocks
  - [4 5 6] → ~1000 blocks

# Kill 3 datanode randomly

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

## Math: 2 group replication

- 6 datanodes → 2 different 3-node set:
  - [1 2 3] [4 5 6]
- 2000 blocks
  - [1 2 3] → ~1000 blocks
  - [4 5 6] → ~1000 blocks

## 2 groups

- datanodes: 6
- copysets: 2
- changes to dataloss: 2:20
- blocks in sets: ~1000

## random

- datanodes: 6
- copysets: 20
- changes to dataloss: 20:20
- blocks in sets: ~100

## 2 groups

- 10 % chance
- 50 % loss

## random

- 100 % chance
- 10 % loss

# Any problem?

# Random replication: recovery

- If only ONE node is failed (1)
- [(1) 2 3] copy blocks from ← 2 3
- [(1) 2 4] copy blocks from ← 2 4
- [(1) 2 5] copy blocks from ← 2 5
- ...

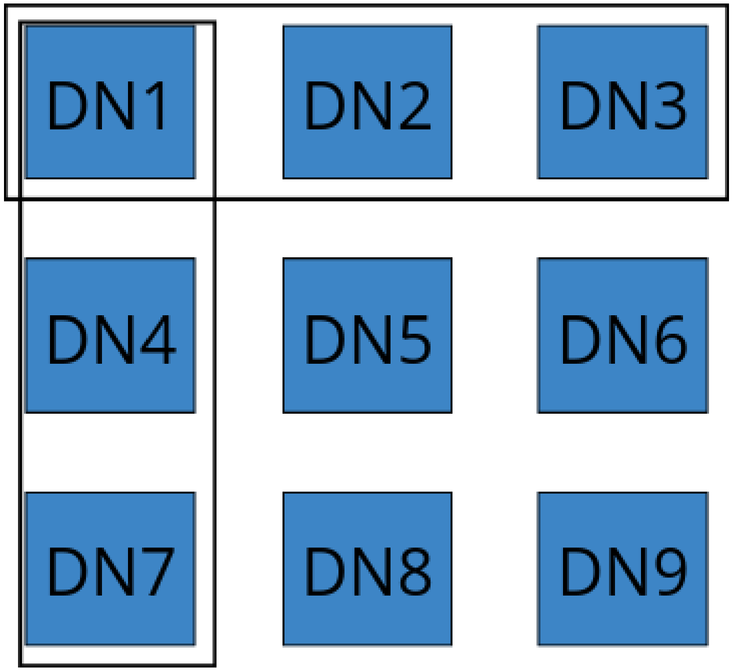
# Group replication: recovery

- If only ONE node is failed (1)
- [(1) 2 3] copy blocks from ← 2 3
- [4 5 6] → HEALTHY

Next try…

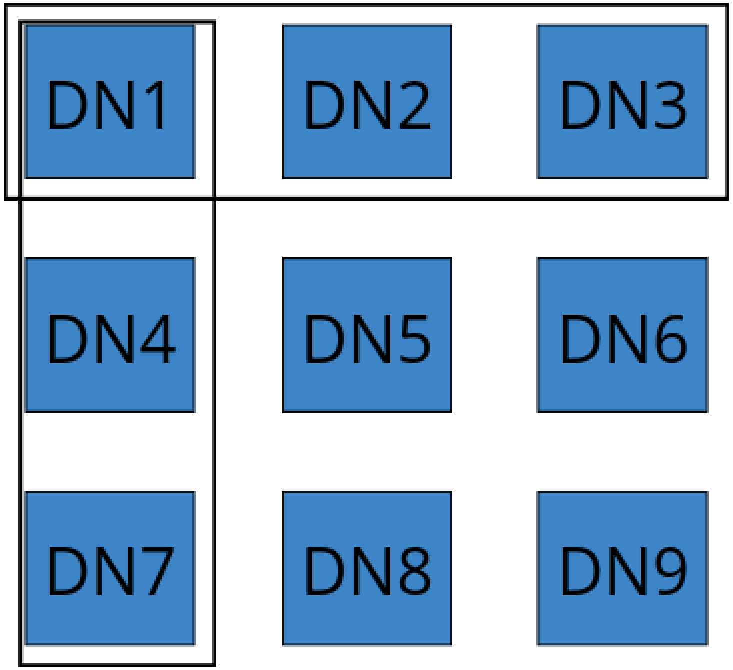
# Copysets

- 6 groups (9 nodes!):
  - [1 2 3]
  - [4 5 6]
  - [7 8 9]
  - [1 4 7]
  - [2 5 8]
  - [3 6 9]



# Copysets

- Chance to loose data:
  - 6:84 (9 node): 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 |

S3 protocol

Hadoop FS

CSI

# Apache Hadoop Ozone

hadoop.apache.org/ozone

THE APACHE SOFTWARE FOUNDATION

# Elek Márton

- Cloudera
  - Pricipal software engineer
- elek@apache.org
- twitter.com/anzix
- Ozone:
  - Ozone Explained (Youtube)
- Kubernetes + Apache Bigdata:
  - **github.com/elek**/flekszible
  - flokkr.github.io