Losing Data in a Safe Way

Advanced Replication Strategies in Apache Hadoop Ozone

DATA SCIENCE

RATIS

RAFT library for Java

Apache Hadoop Ozone

Marton Elek
elek@apache.org
@anzix

https://github.com/elek/flekszible
https://flokkr.github.io

DATA
SCIENCE

hadoop

Storage in the world

hadoop-hdfs

Storage in the hadoop world

hadoop-hdfs

hadoop-aws/aliyun/azure

Storage in the hadoop world

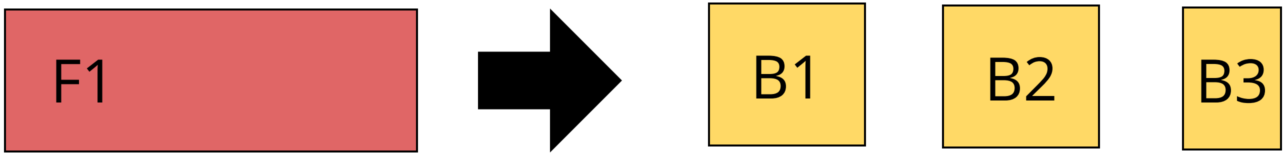
hadoop-hdfs

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

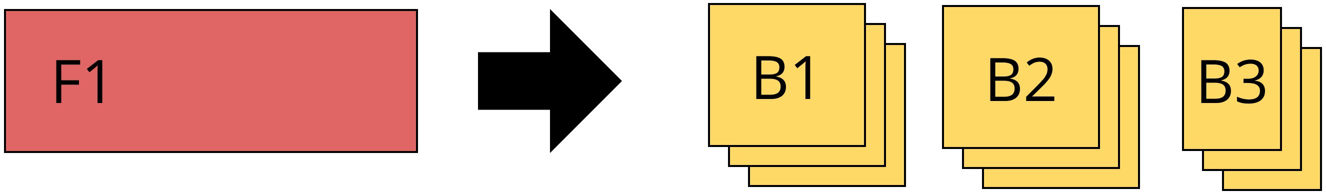
hadoop-aws/aliyun/azure

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

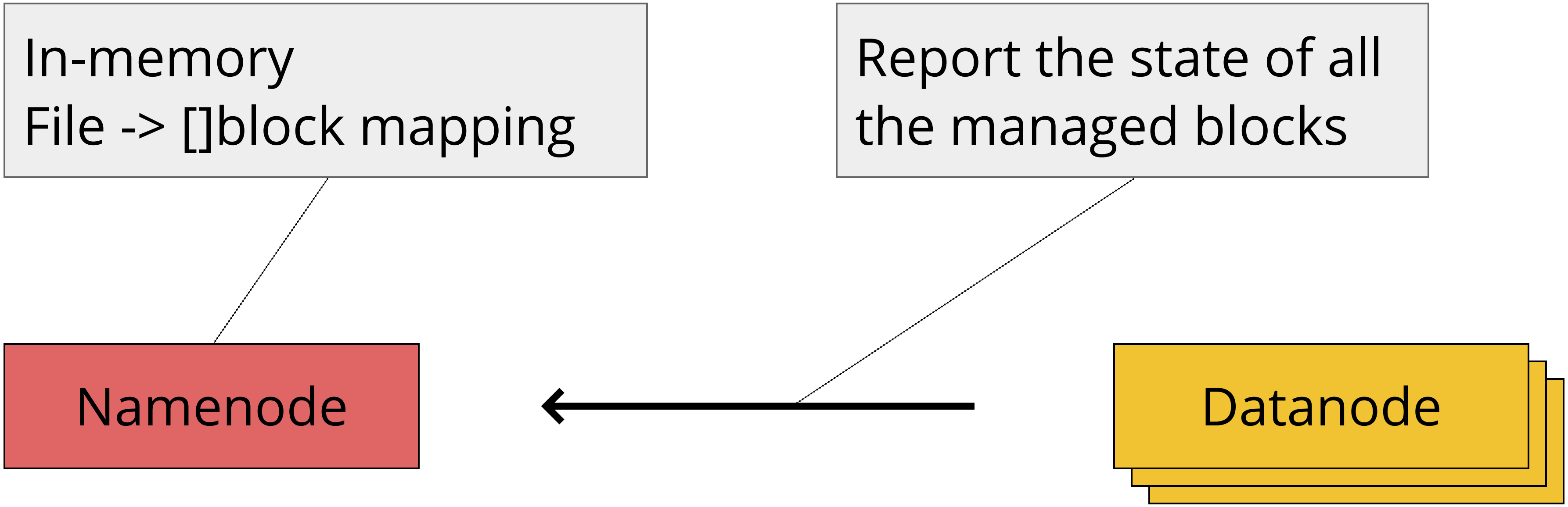
Replication: split the files



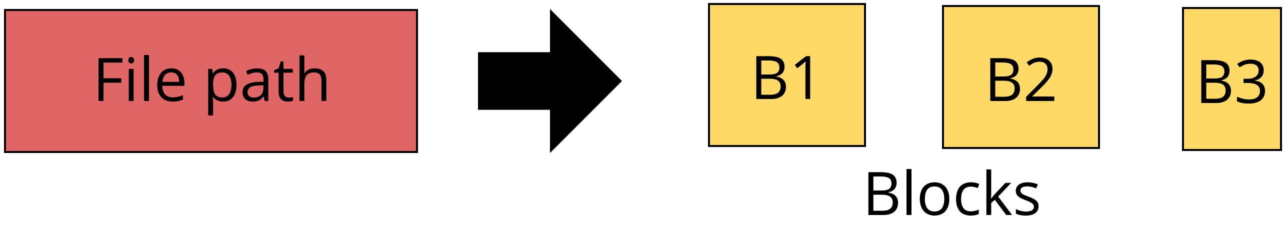
Replication: split the files



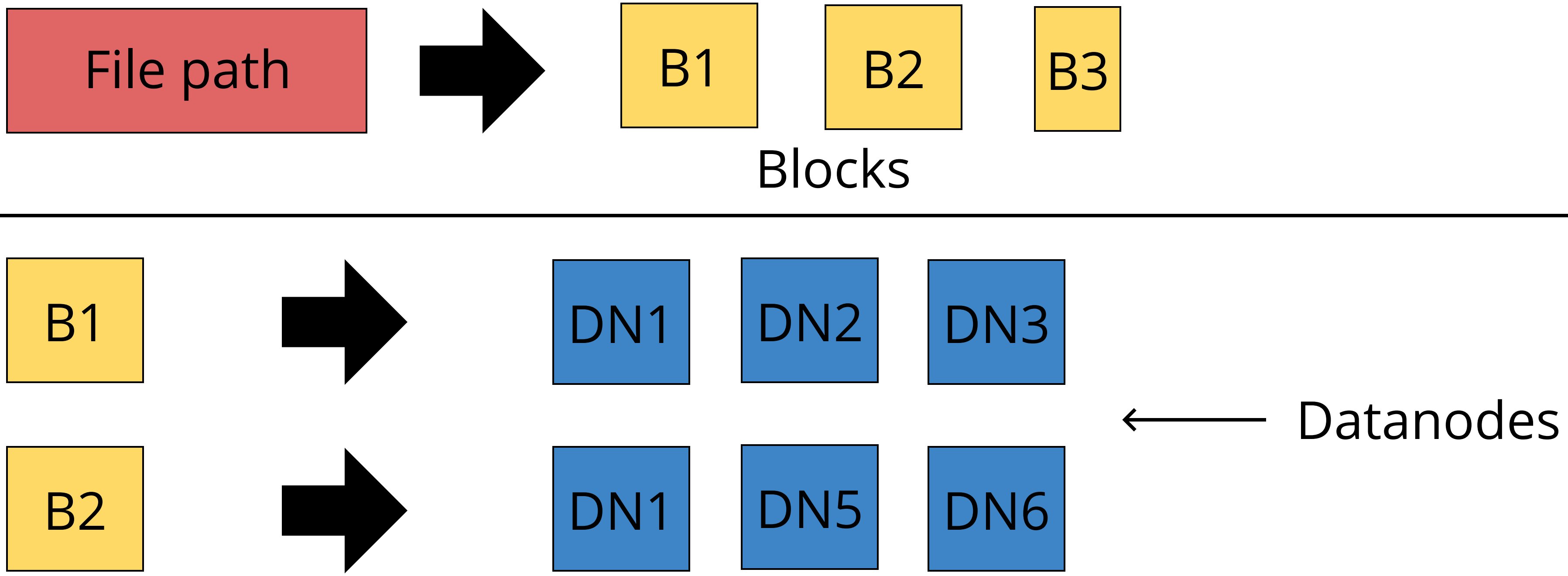
Replication: split the files

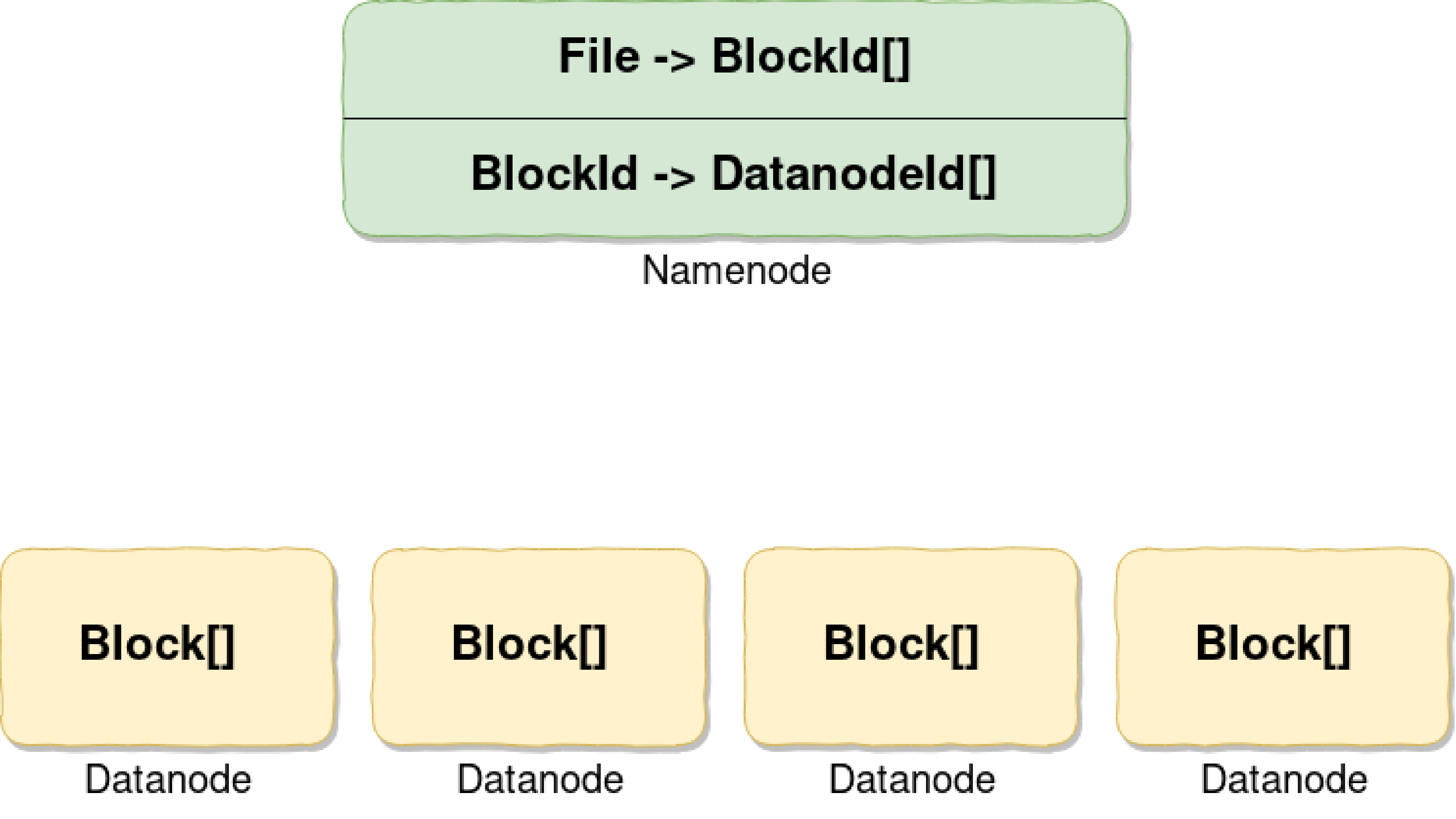


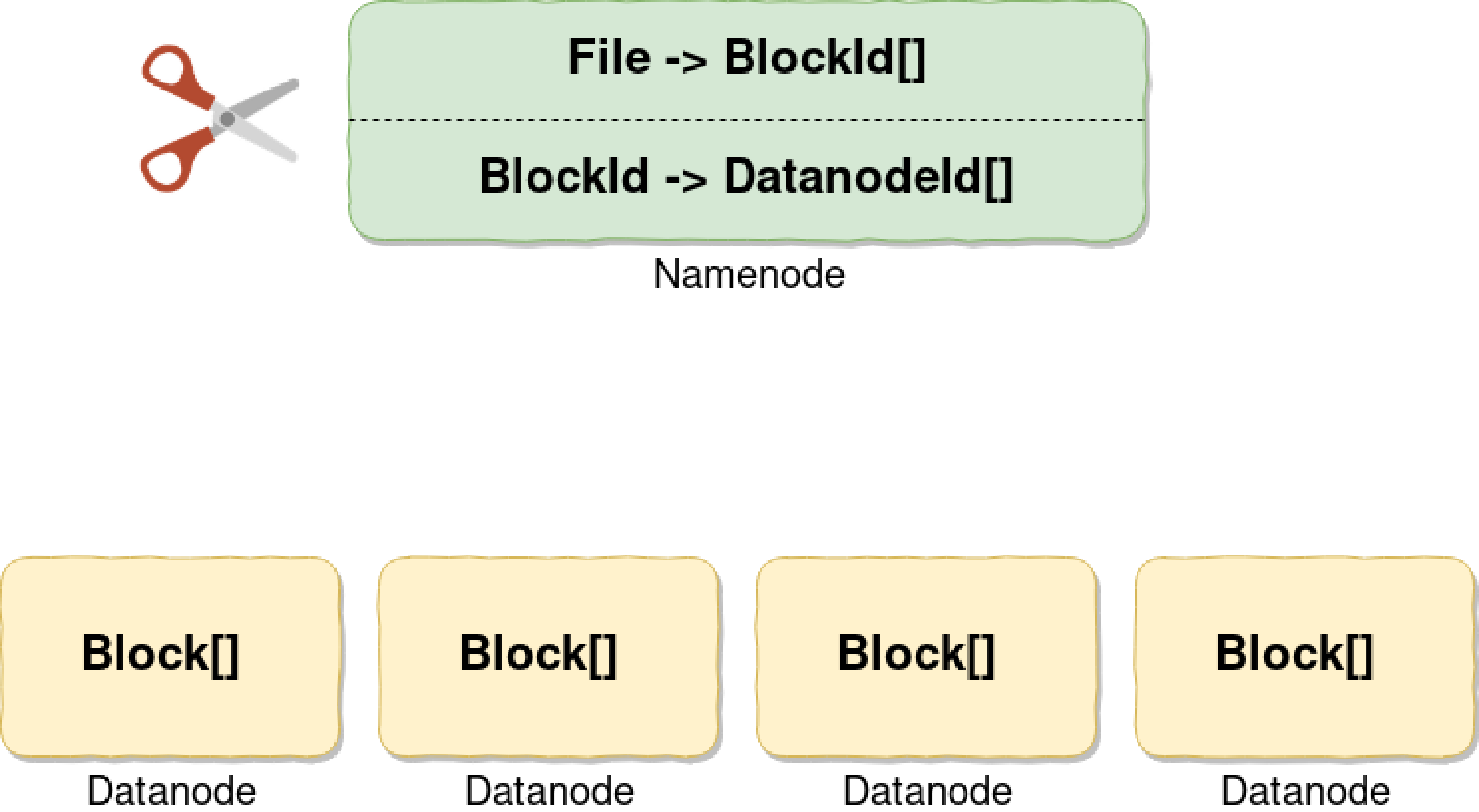
Inside HDFS Namenode

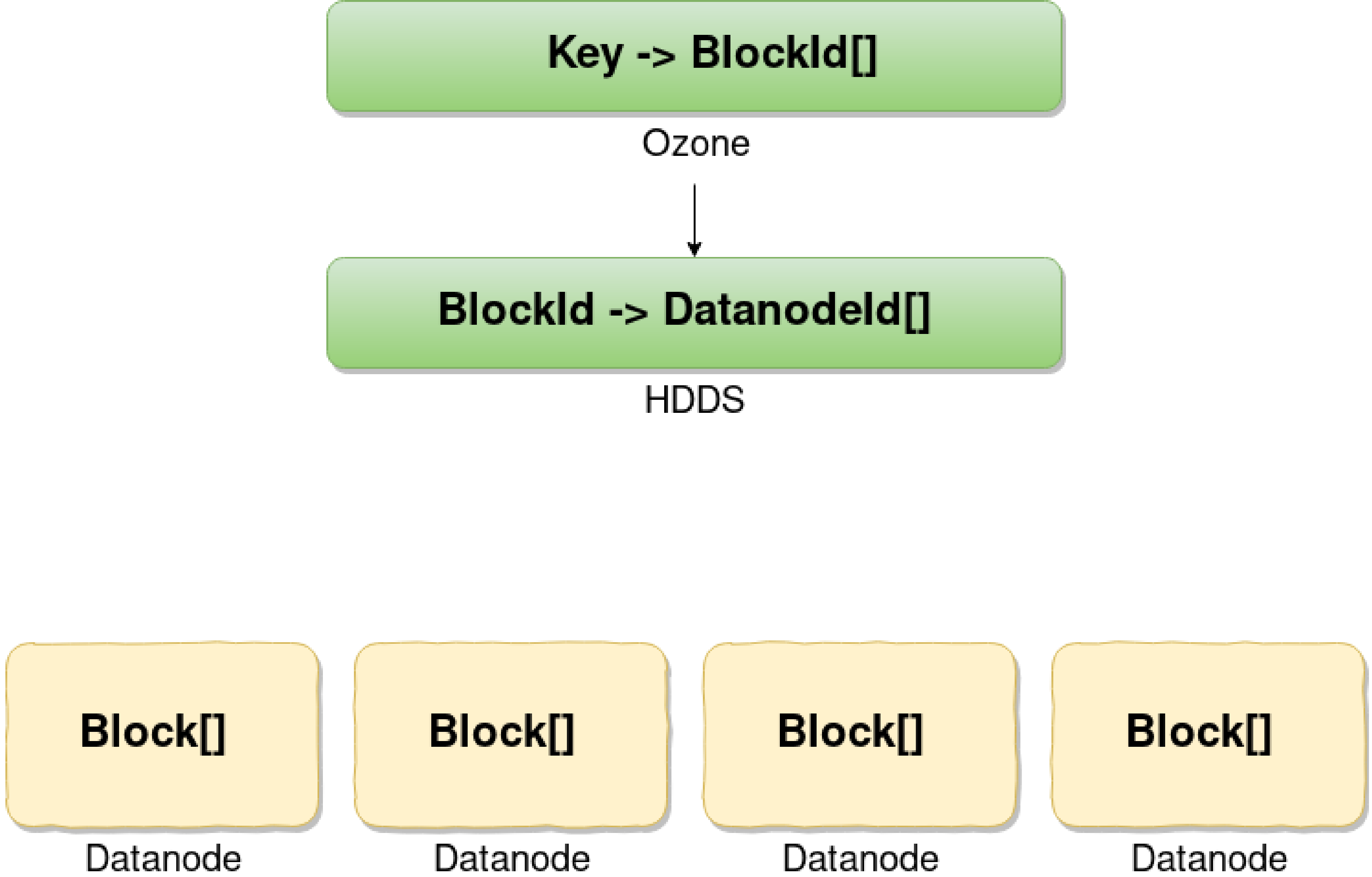


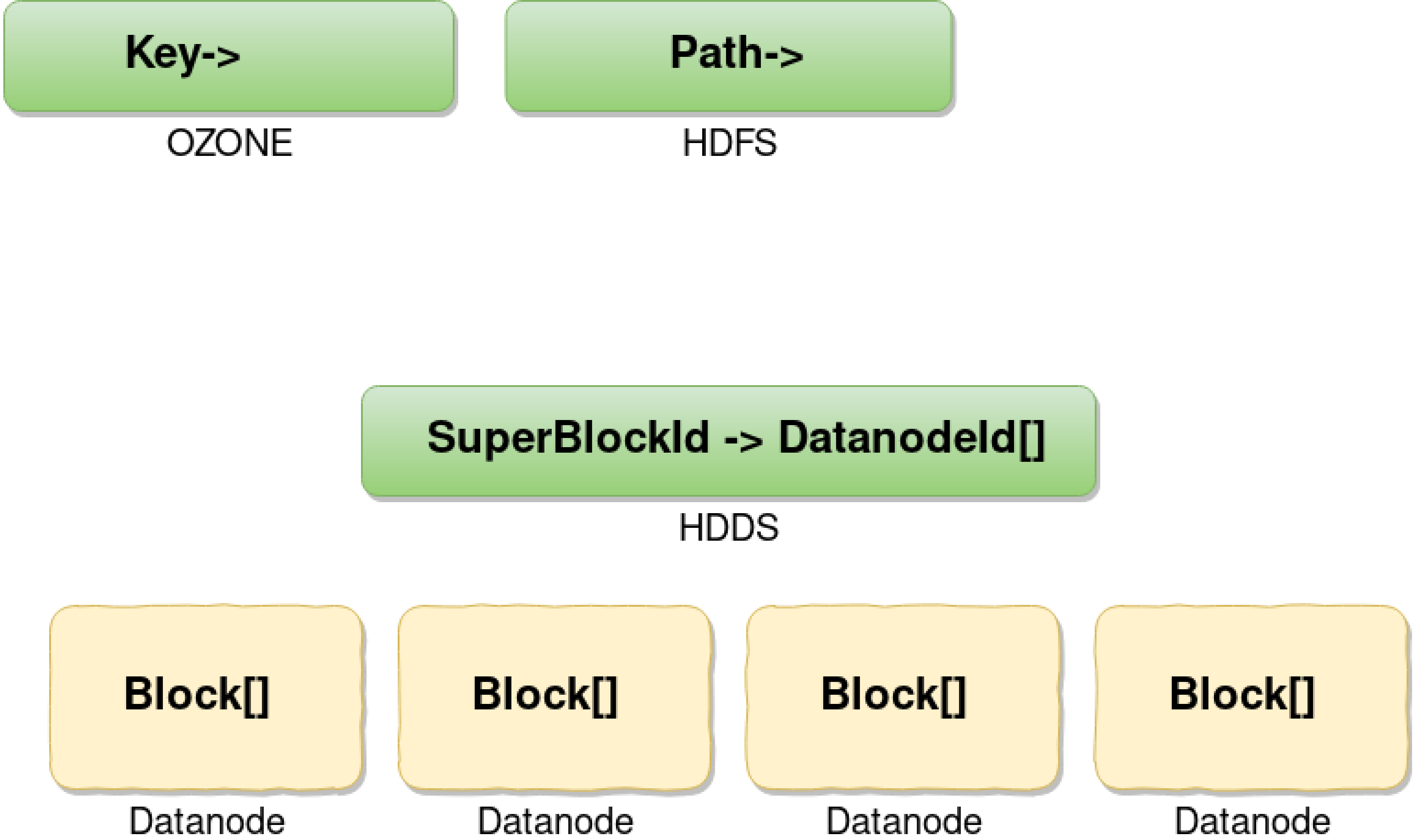
Inside HDFS Namenode

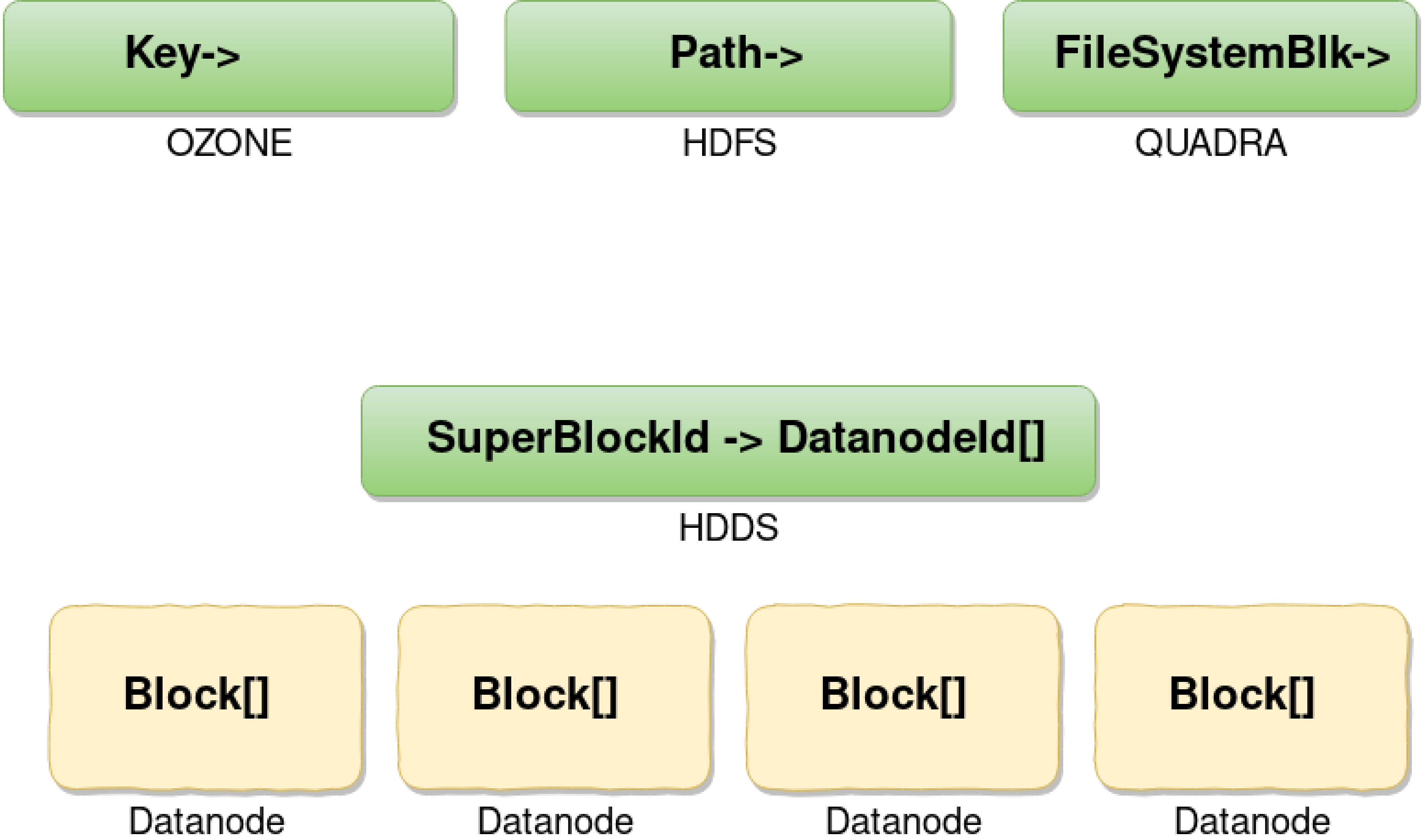


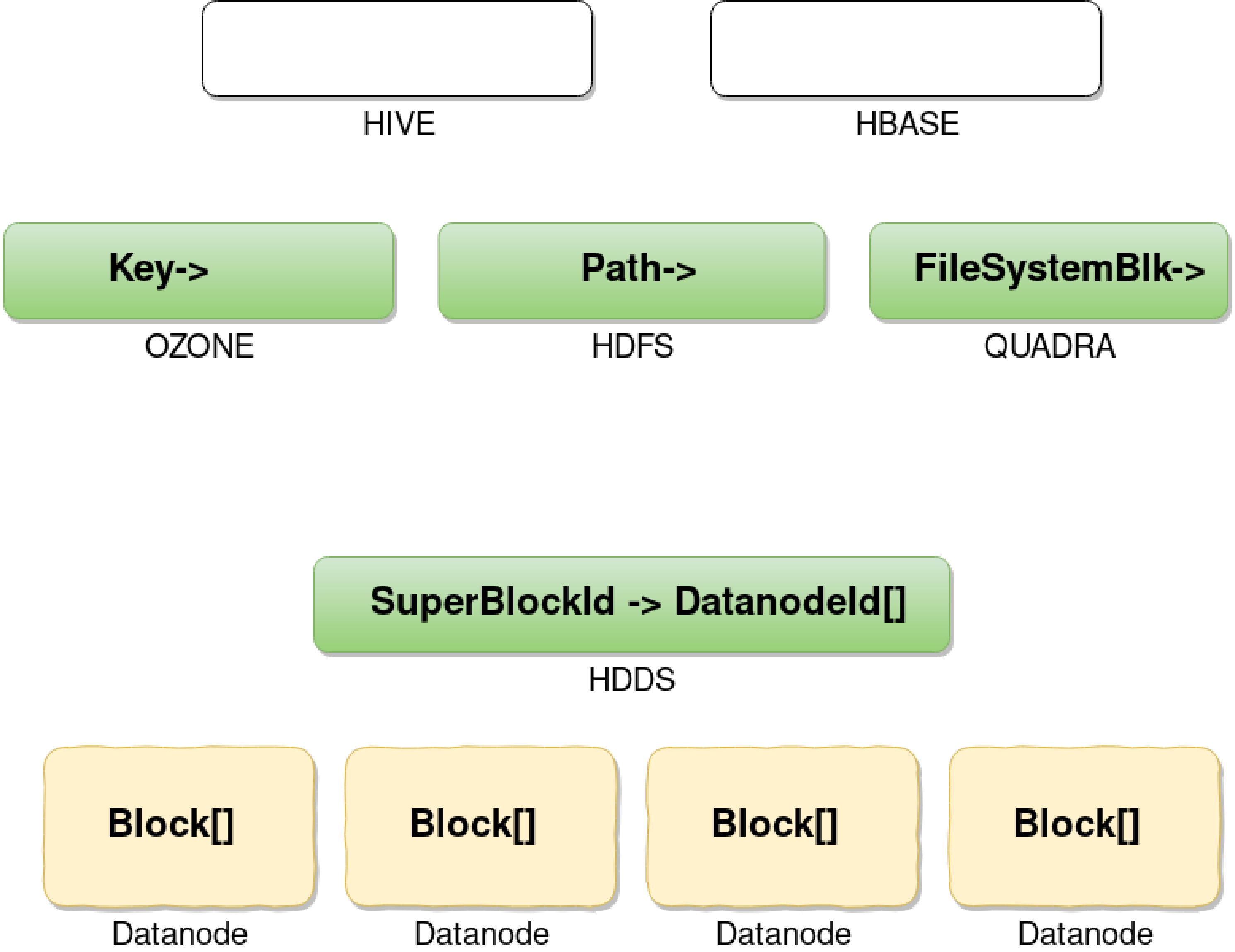

File -> BlockId[]
BlockId -> DatanodeId[]
Namenode
Block[]
Block[]
Block[]
Block[]
Datanode
Datanode
Datanode
Datanode


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


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


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

- Consistent

- Fast

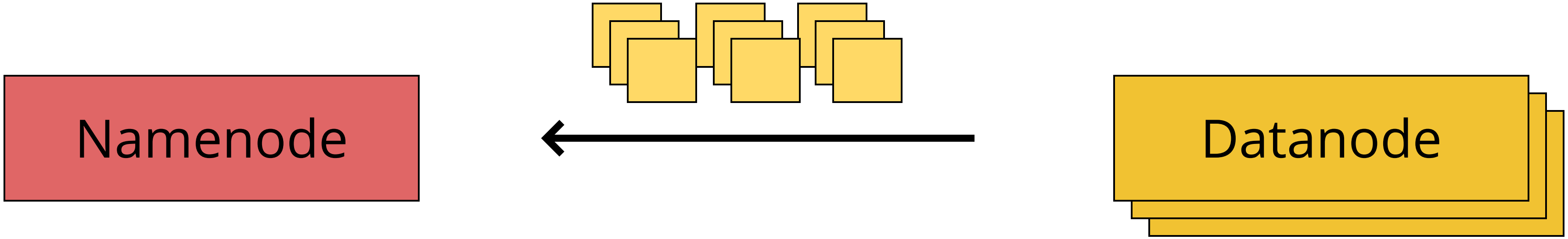
- Cloud native

- Easy to use

Bonus: reporting superblocks

Replicate blocks in bigger groups:

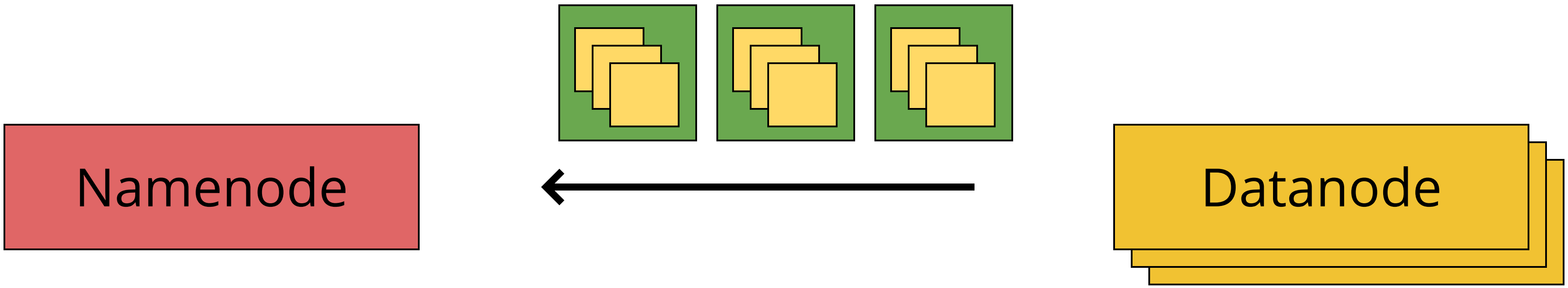
- Can handle 10 billions of blocks



Bonus: reporting superblocks

Replicate blocks in bigger groups:

- Can handle 10 billions of blocks



Open containers

- Replicated with RAFT
  - Apache Ratis
- If the size is < 5Gb
- If everything is fine

Closed Containers

- Full containers
- Immutable data!
- Can be merged after deleting data

S3 protocol

Hadoop FS

CSI

Apache Hadoop Ozone

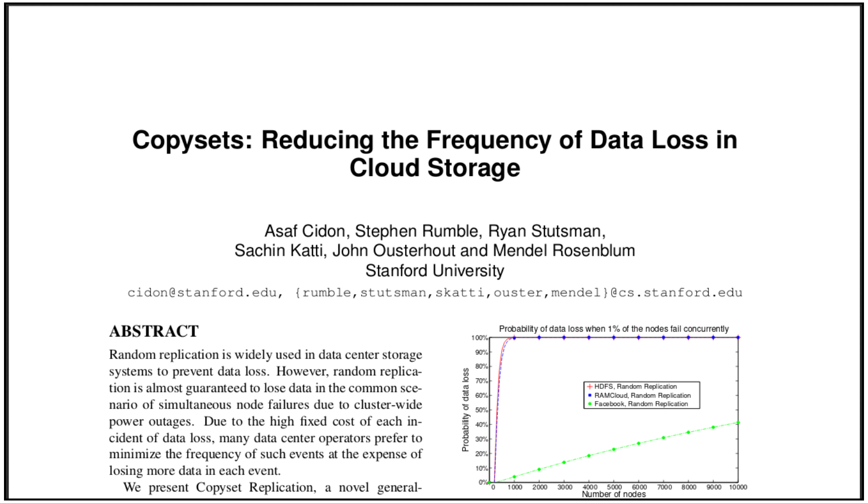
hadoop.apache.org/ozone

Roadmap

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

DATA SCIENCE

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

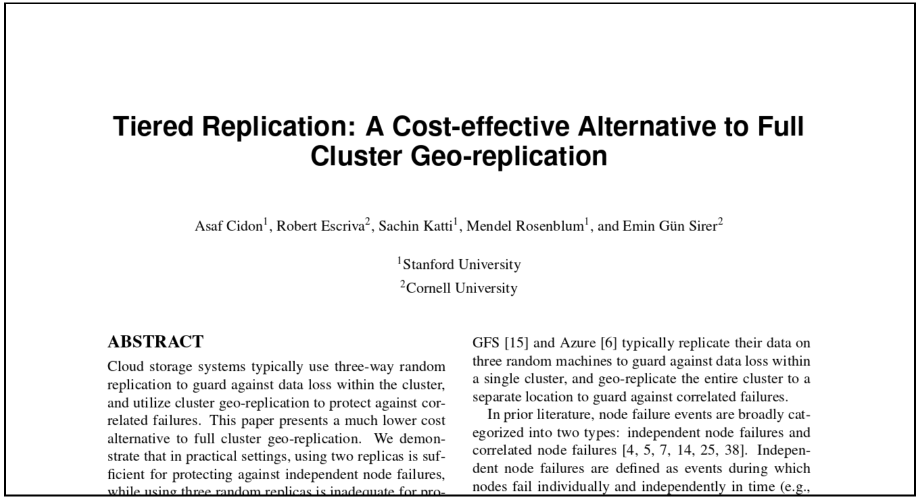
ABSTRACT

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

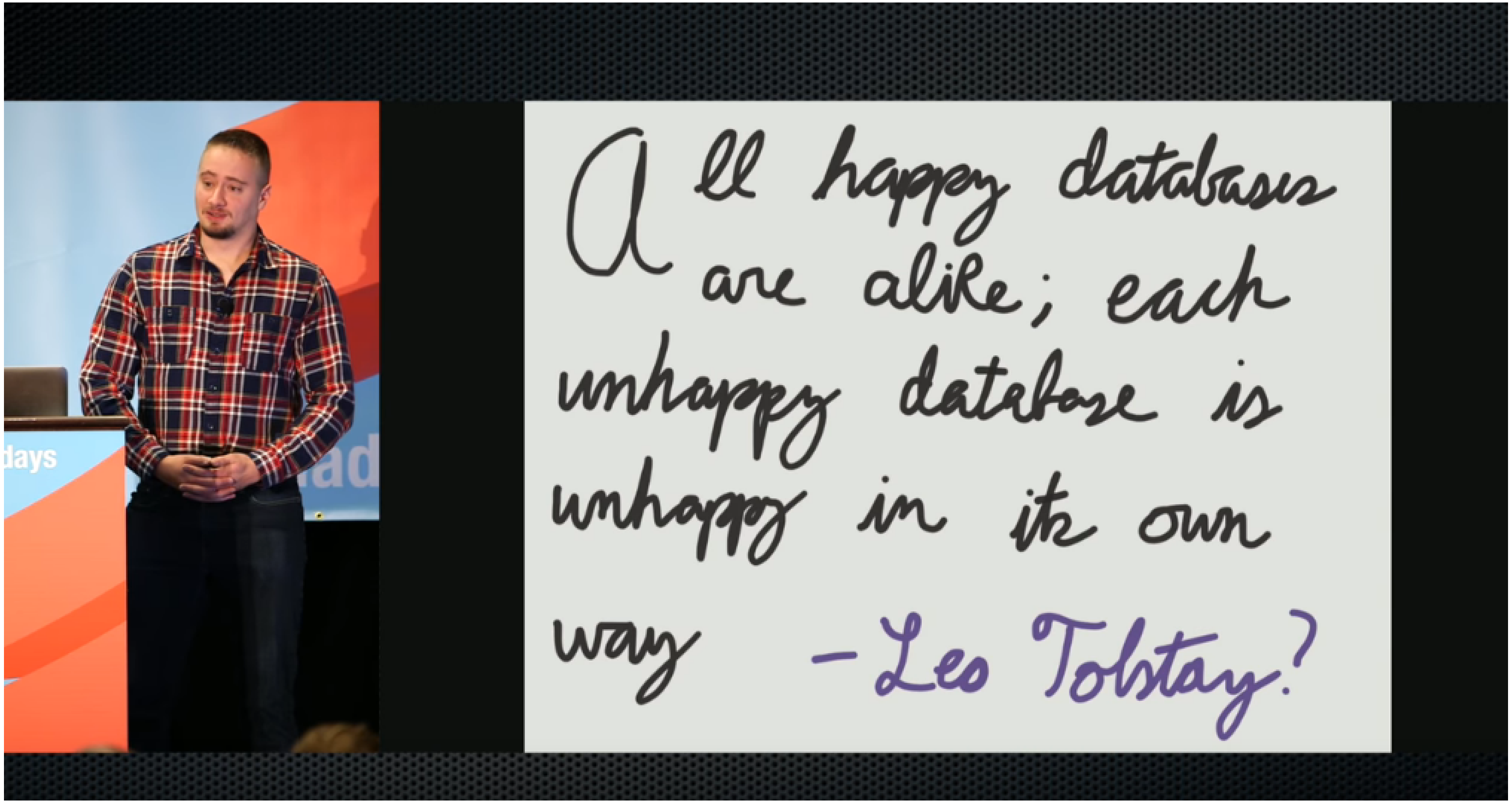
ABSTRACT

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

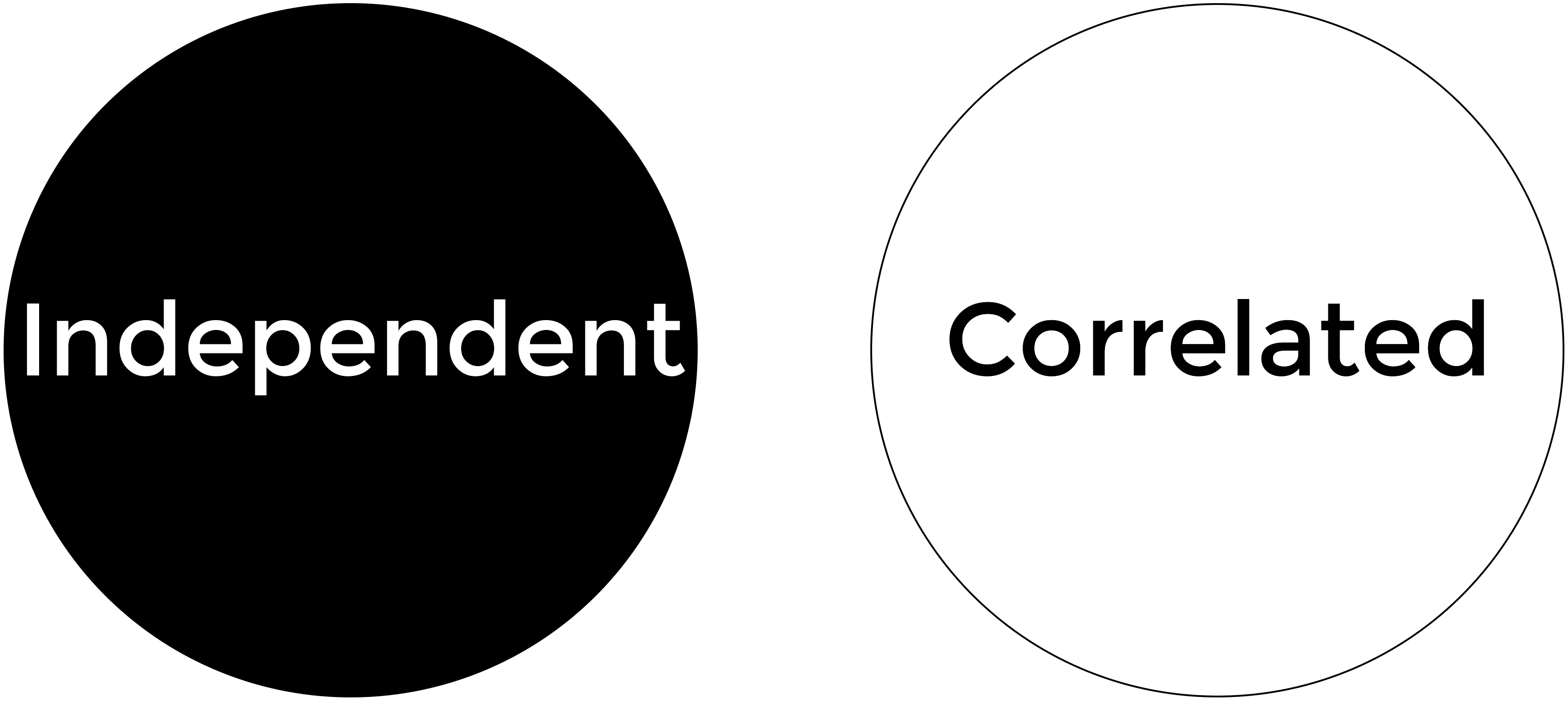


Kyle Kingsbury: Anna Concurrenina #jepsen

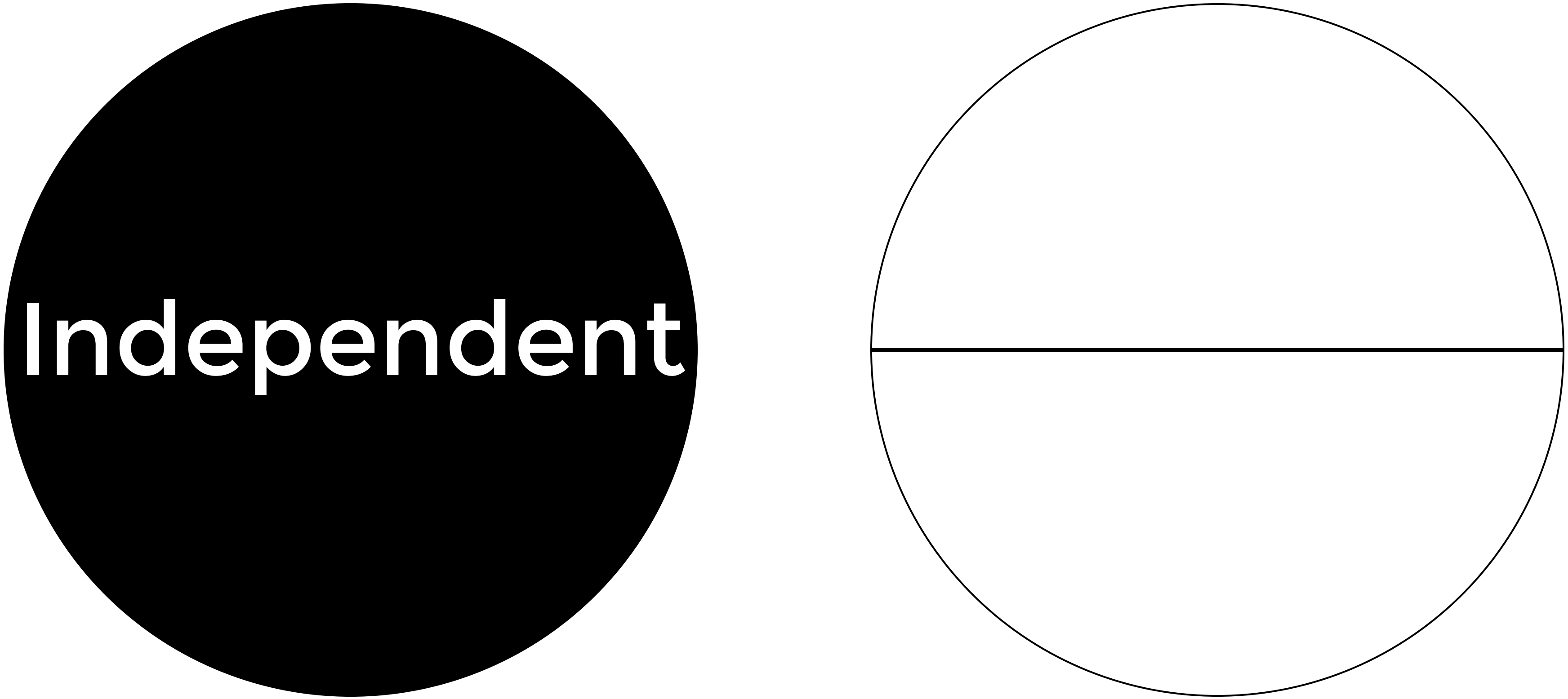
https://www.youtube.com/watch?v=eSaFVX4izsQ

What could go wrong?

Node failures

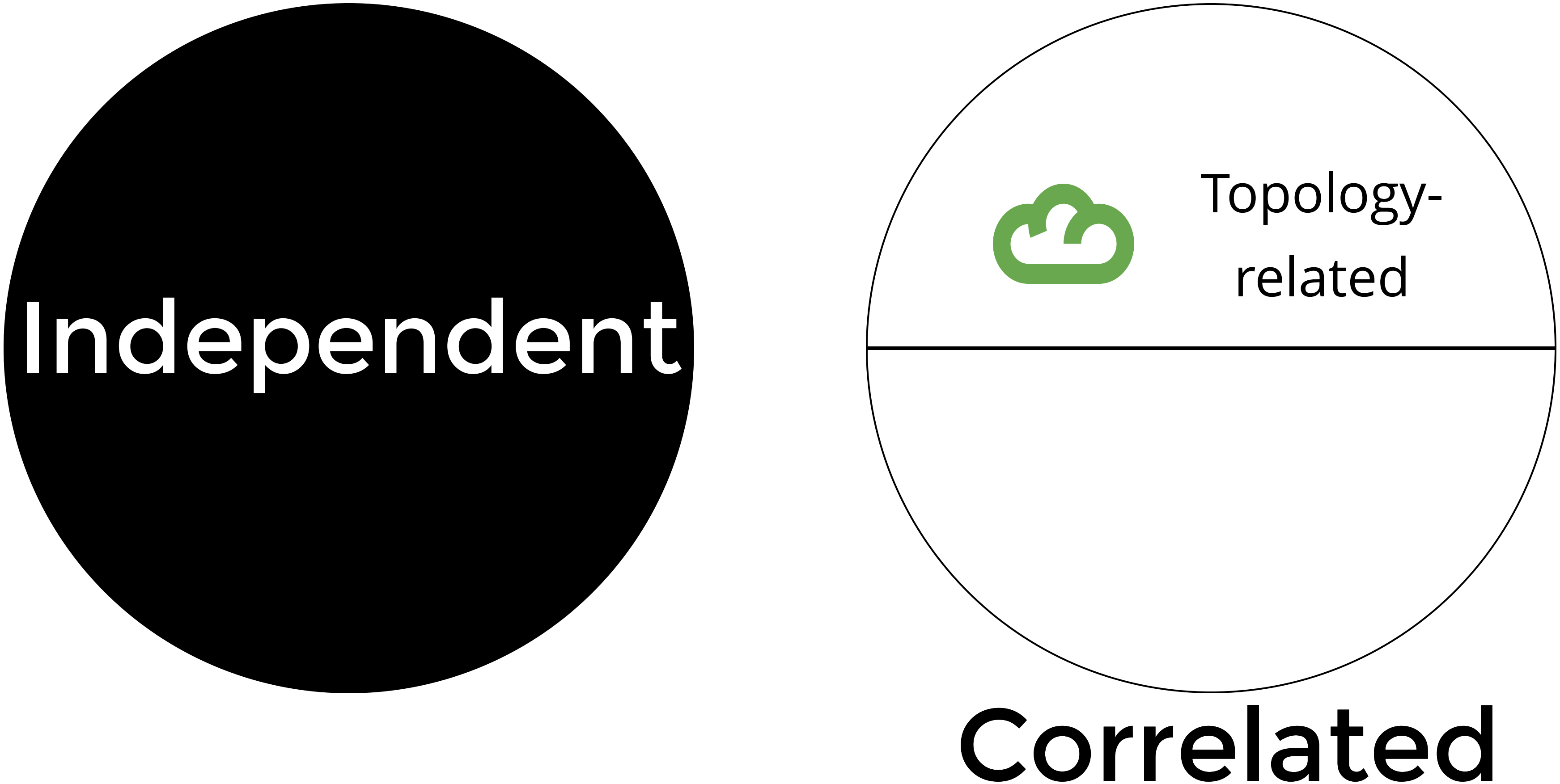


Node failures

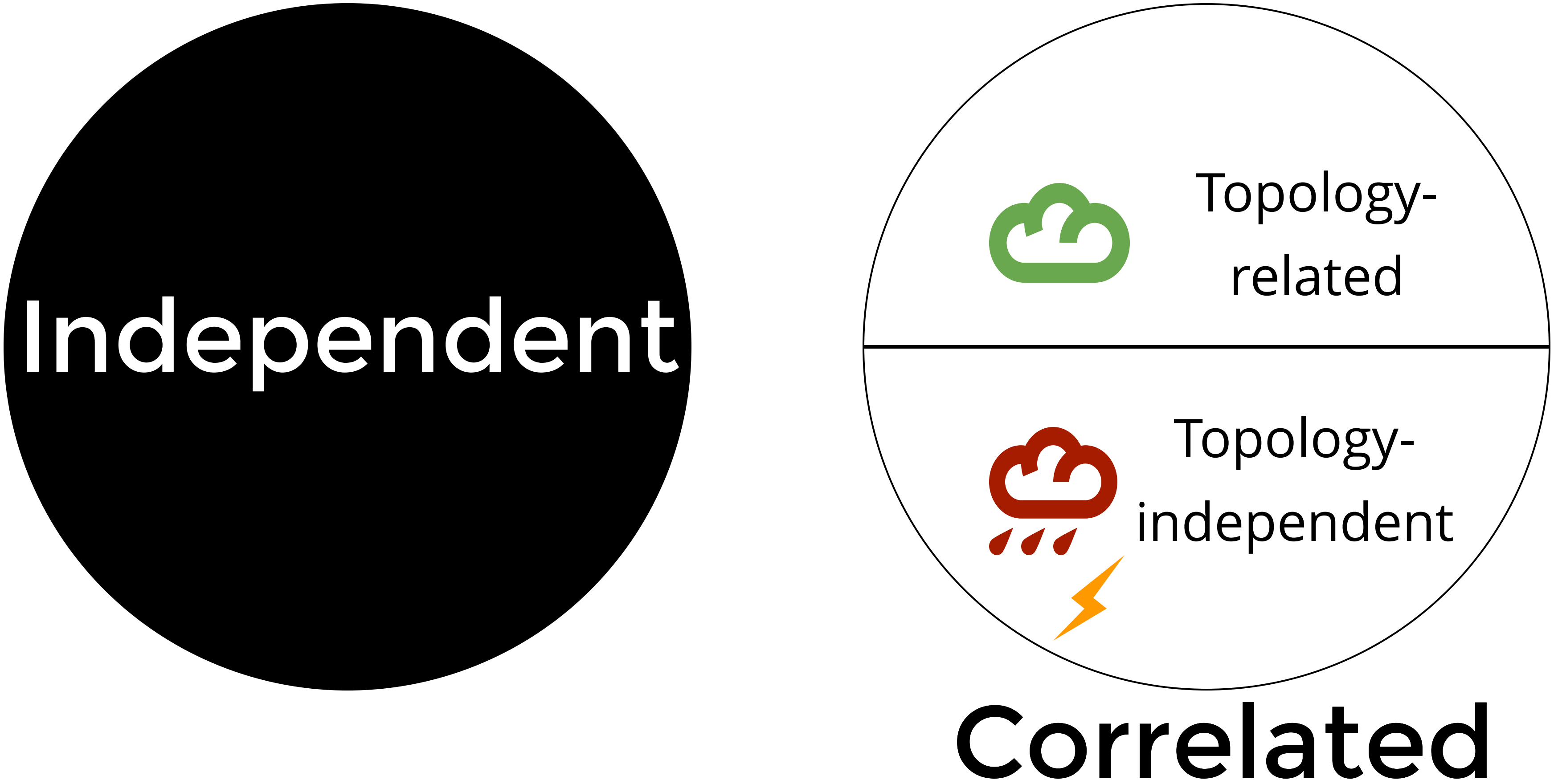


Correlated

Node failures



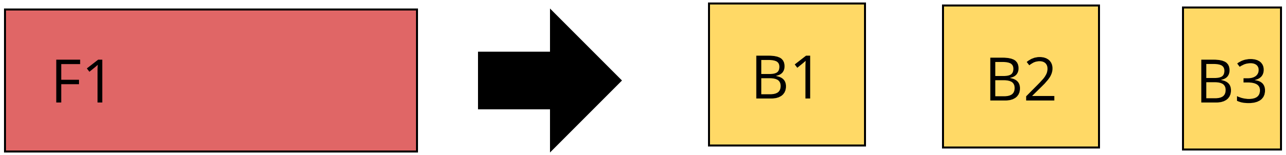
Node failures



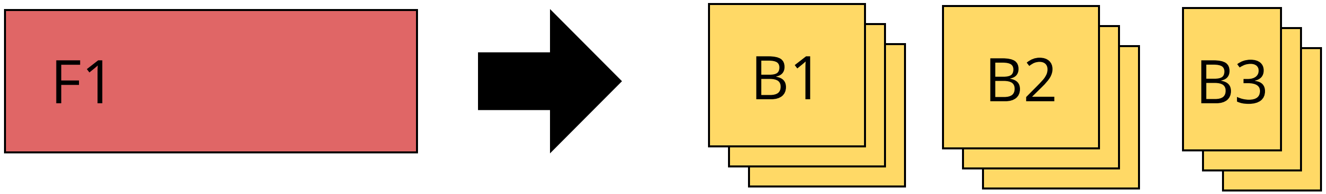
Topology-independent

correlated failures

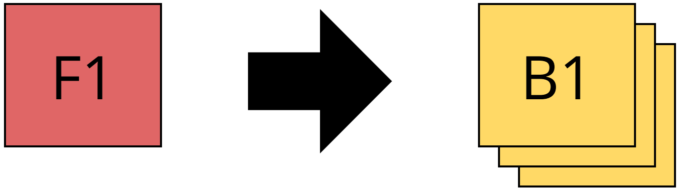
Replication: split the files



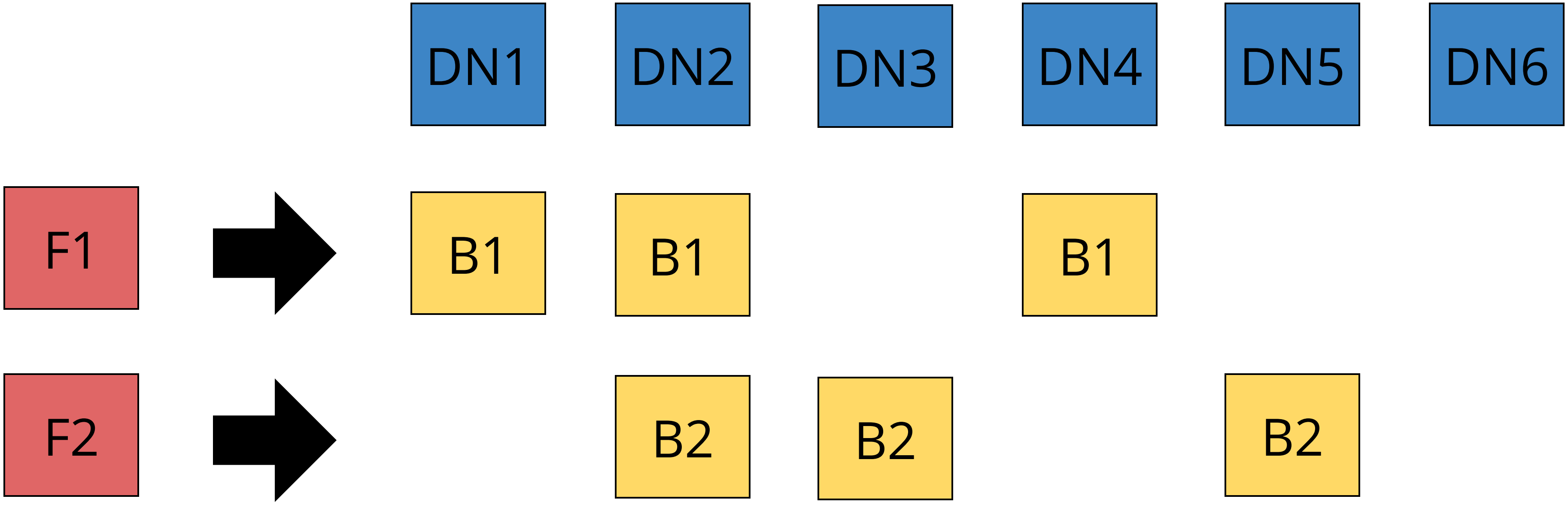
Replication: split the files



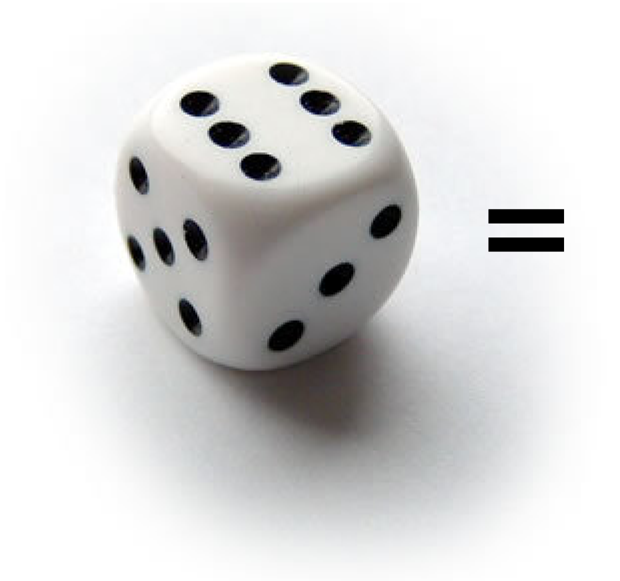
Replication: split the files



Replication

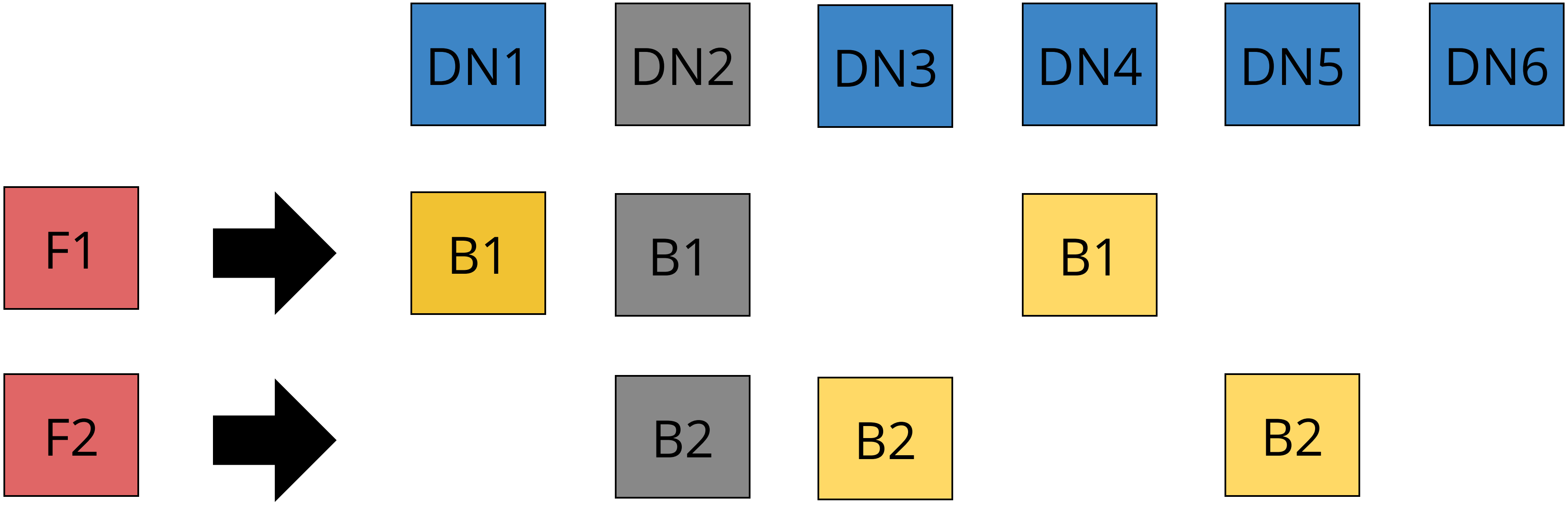


Random datanode failure



= #2

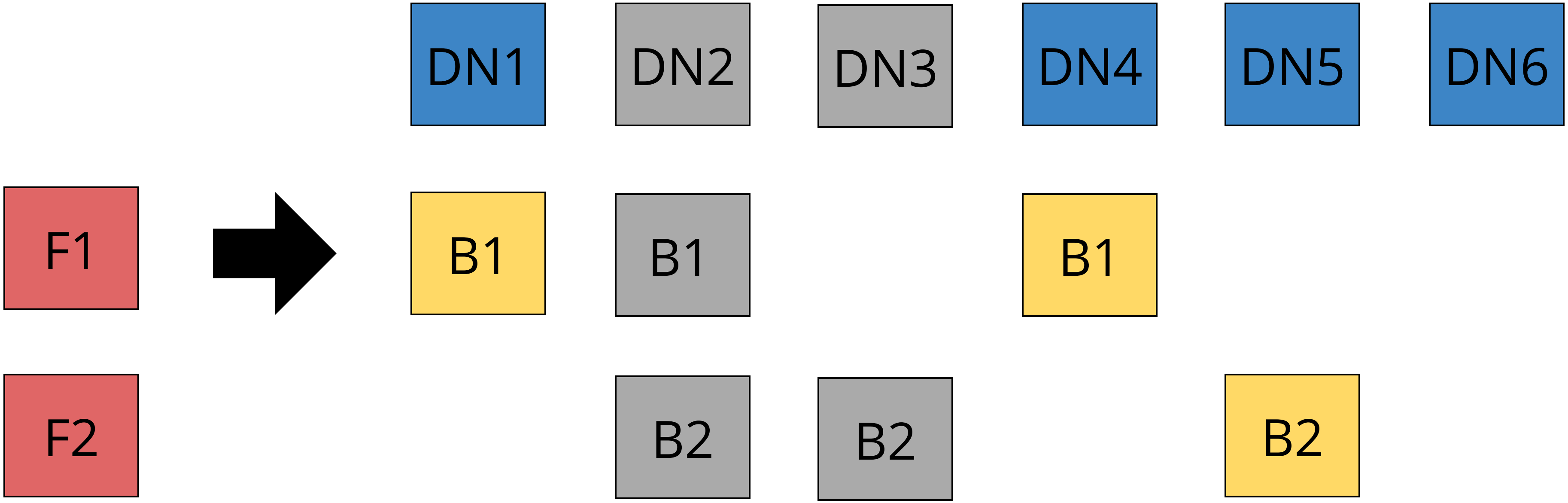
Replication



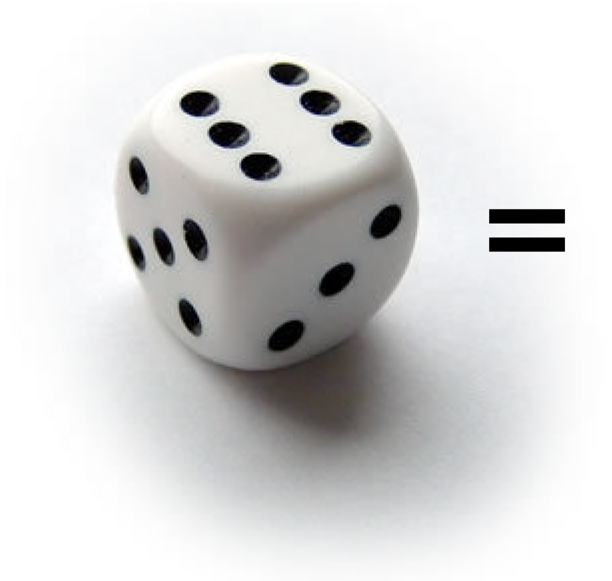
Random datanode failure

= #3

Replication: split the files

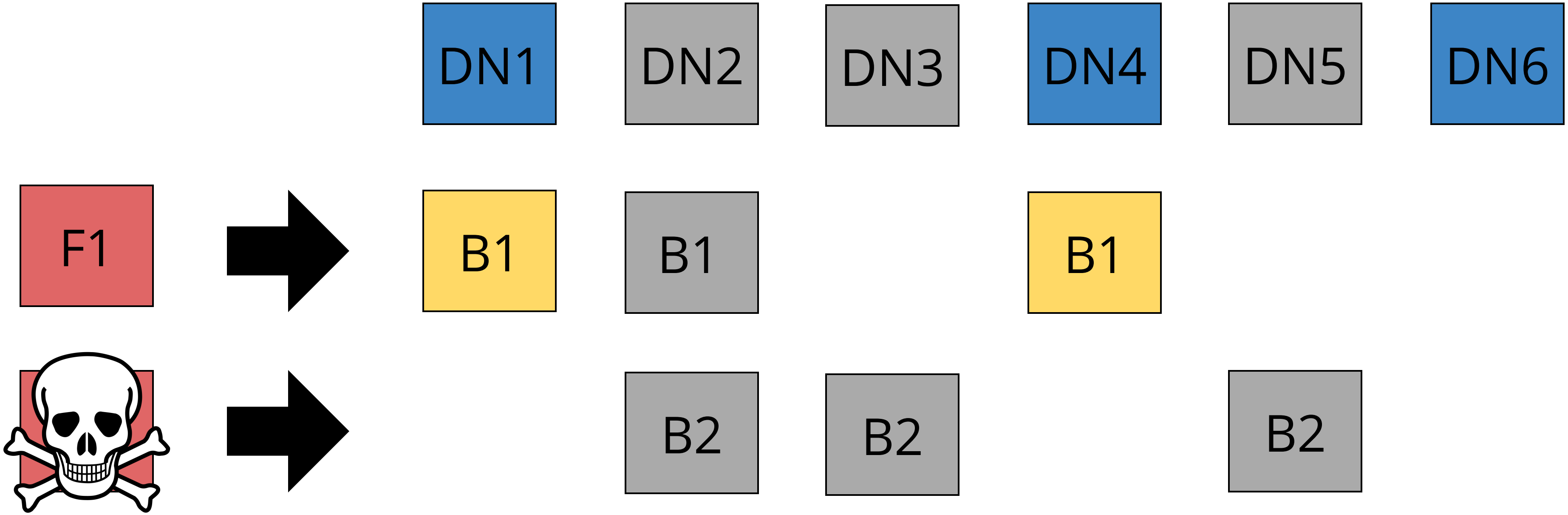


Random datanode failure



= #5

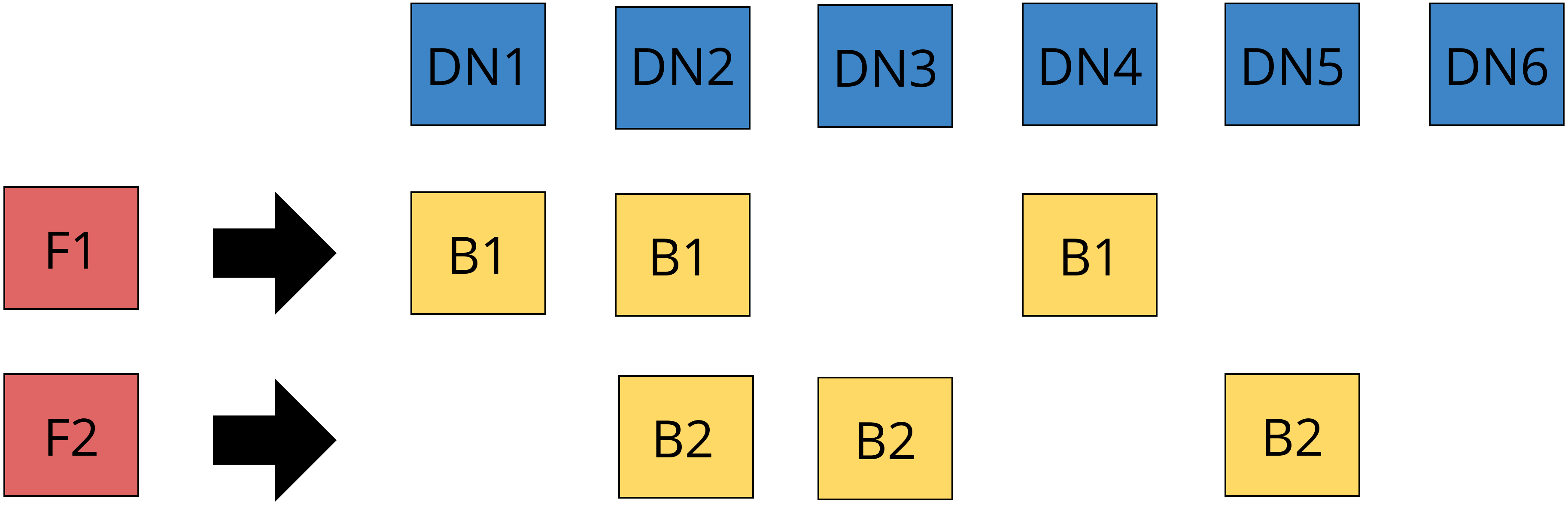
Replication: split the files

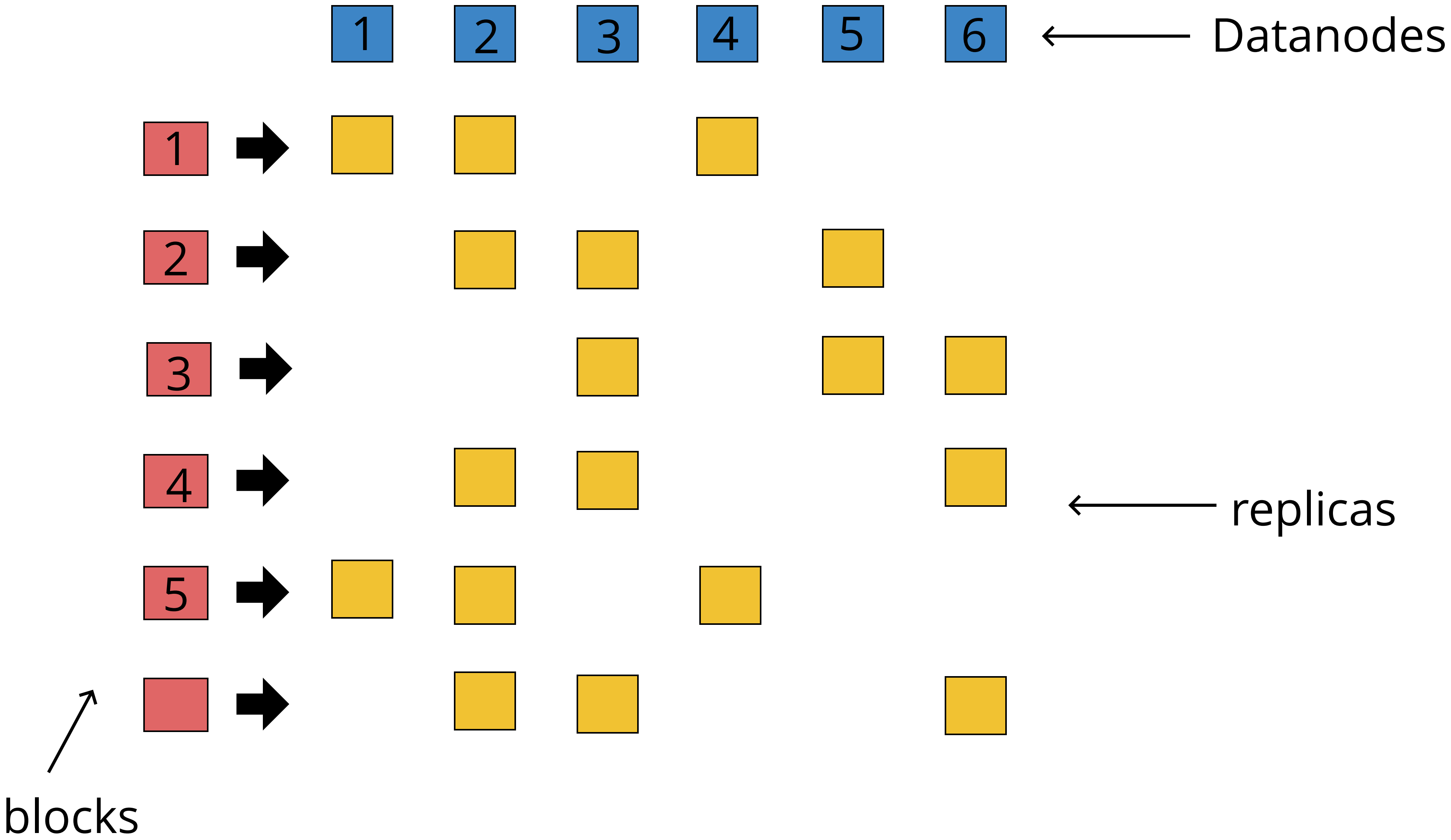


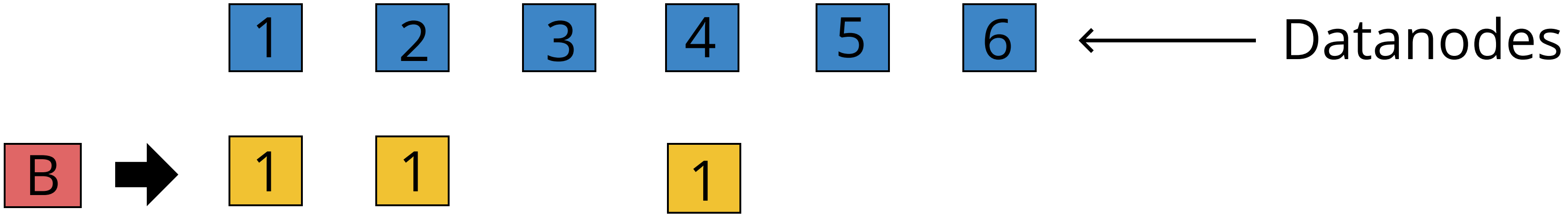
failure of
3 nodes

data loss
with 100% prob

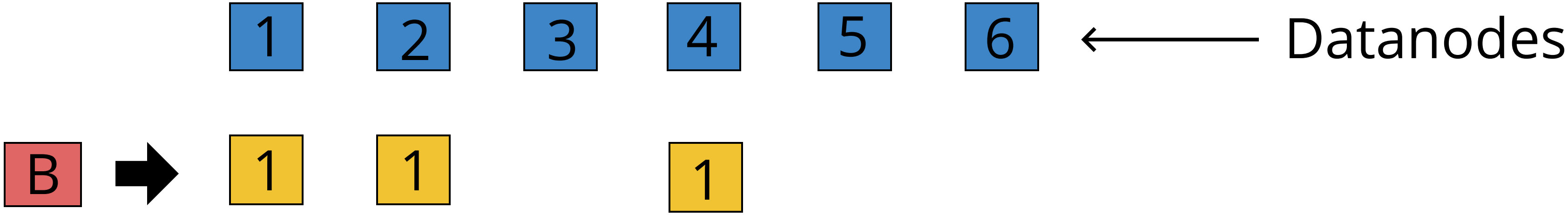
Replication




1
2
3
4
5
6
Datanodes
1
2
3
4
5
replicas
blocks



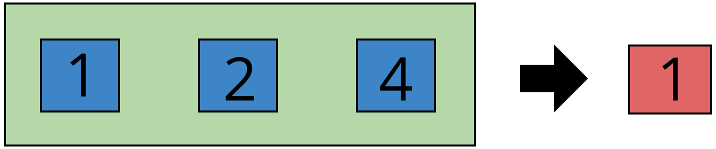
Block B is replaced to datanode: 1,2,3

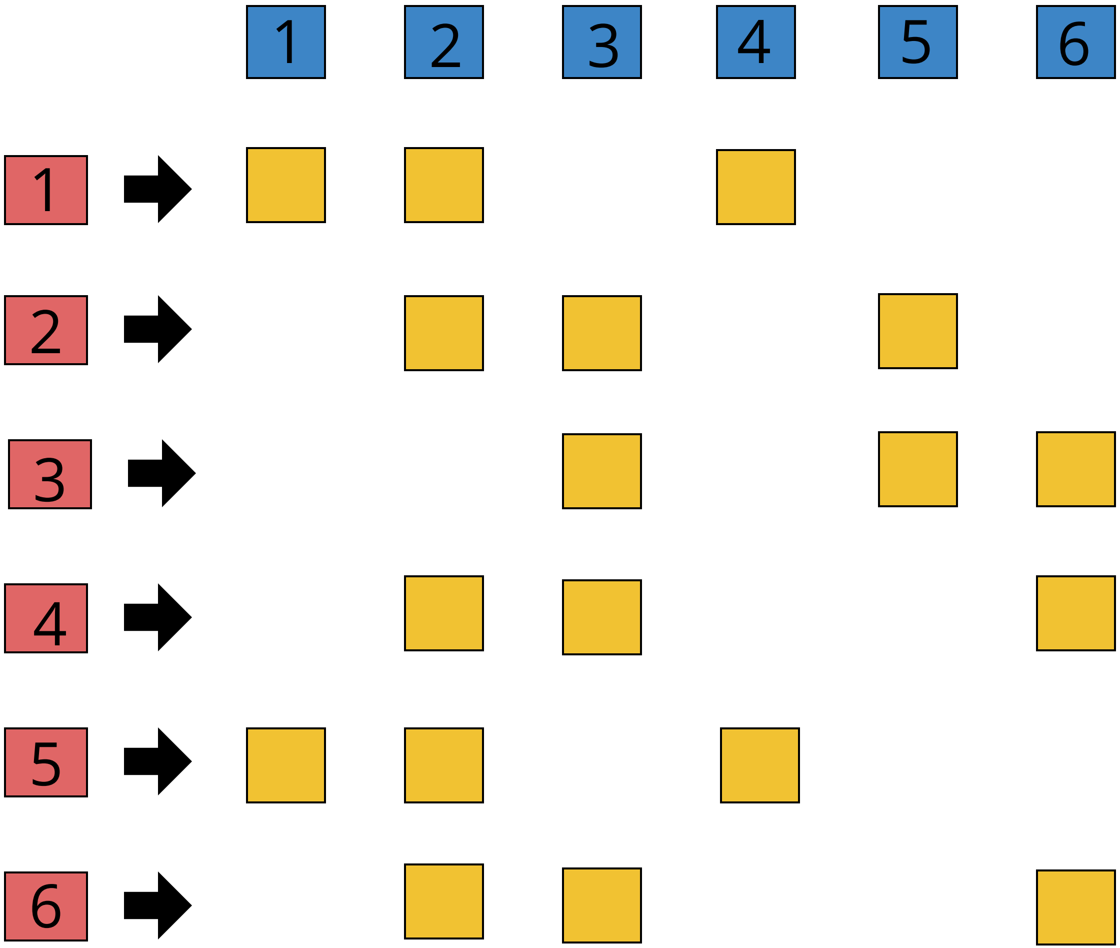
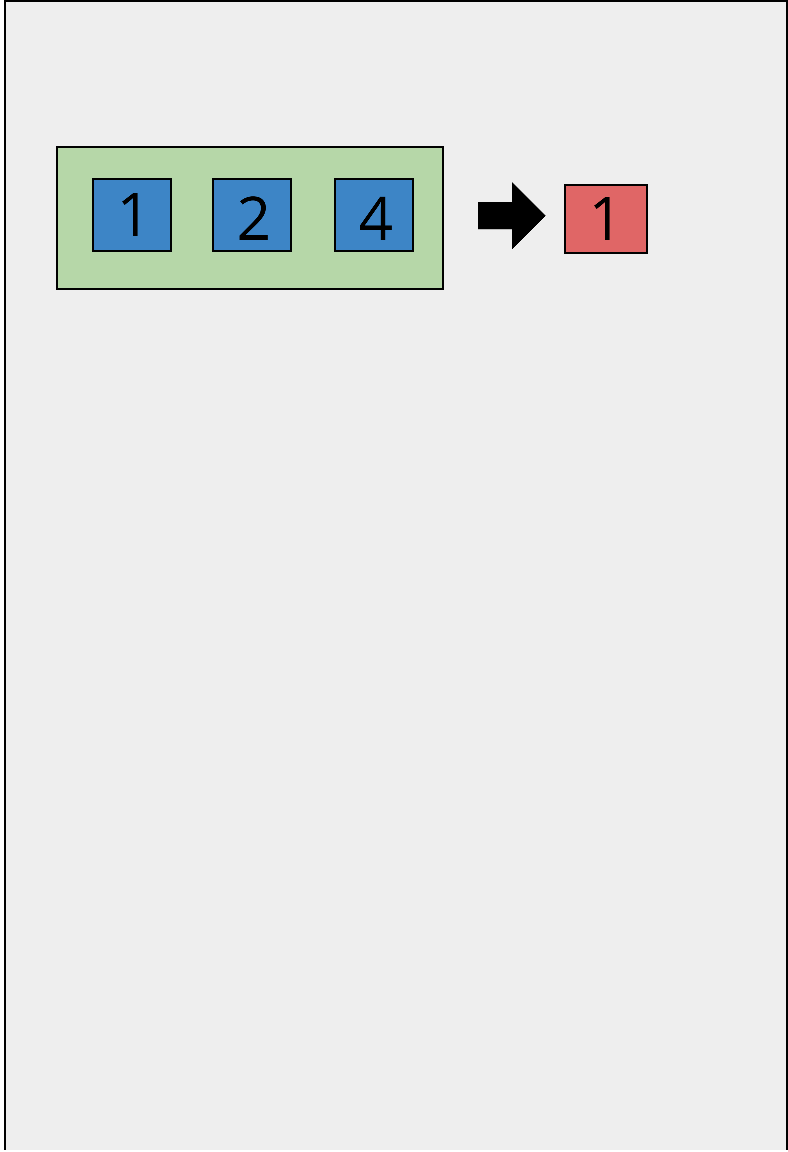


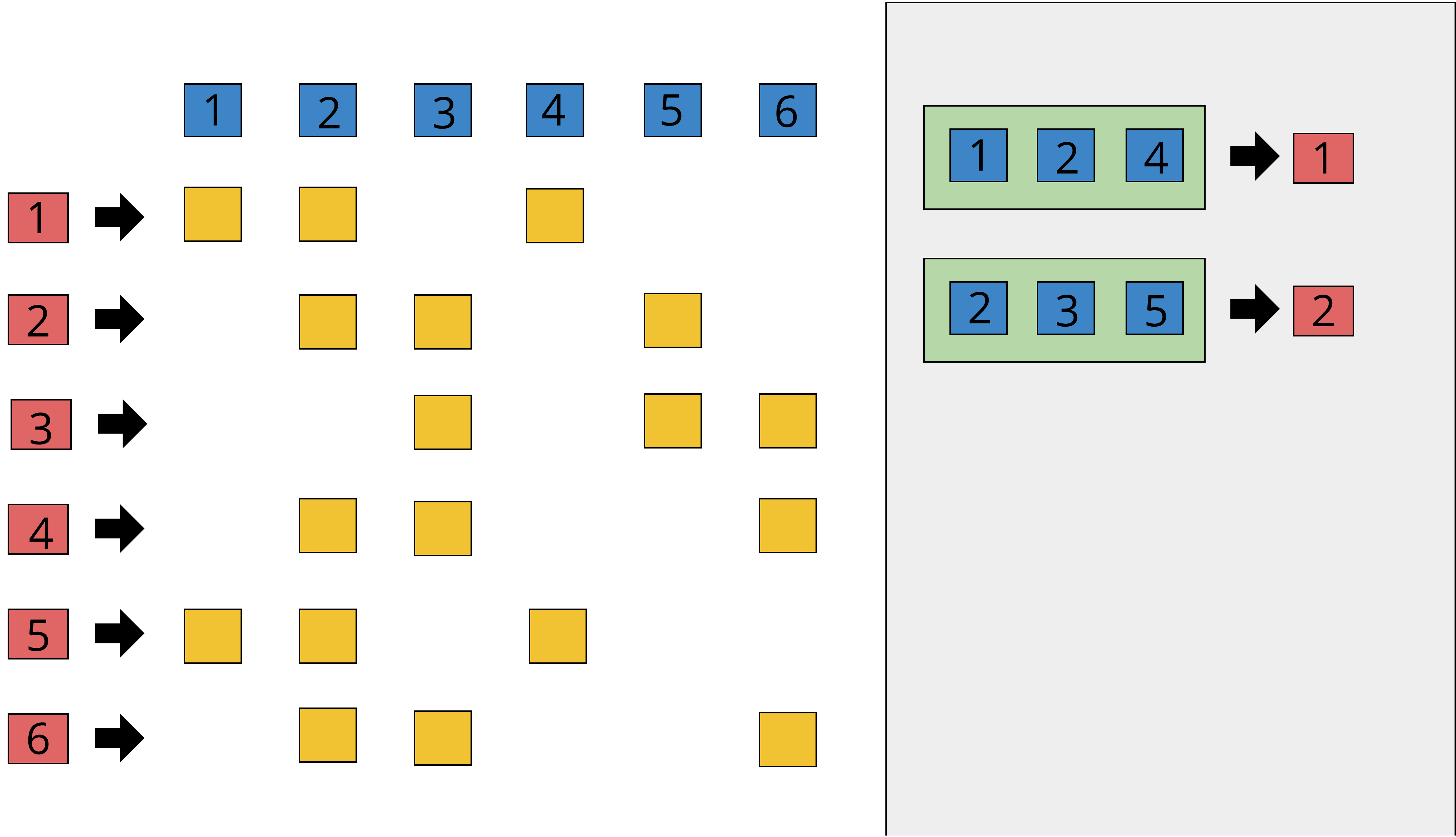
Block B is replaced to datanode: 1,2,3

ReplicaSet: Set of datanodes (3)

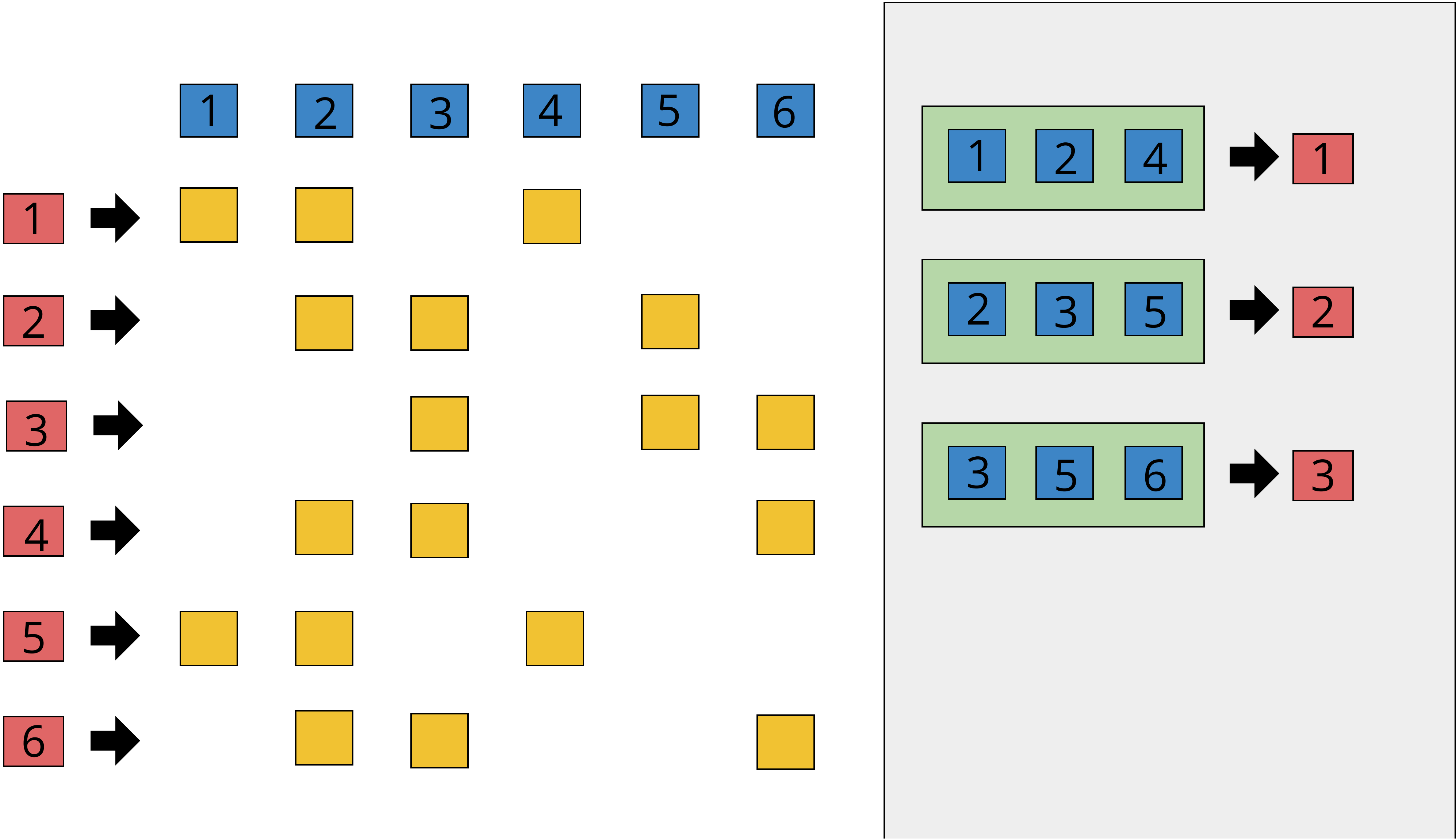
ReplicaSet (1,2,4) manages replicas of B block

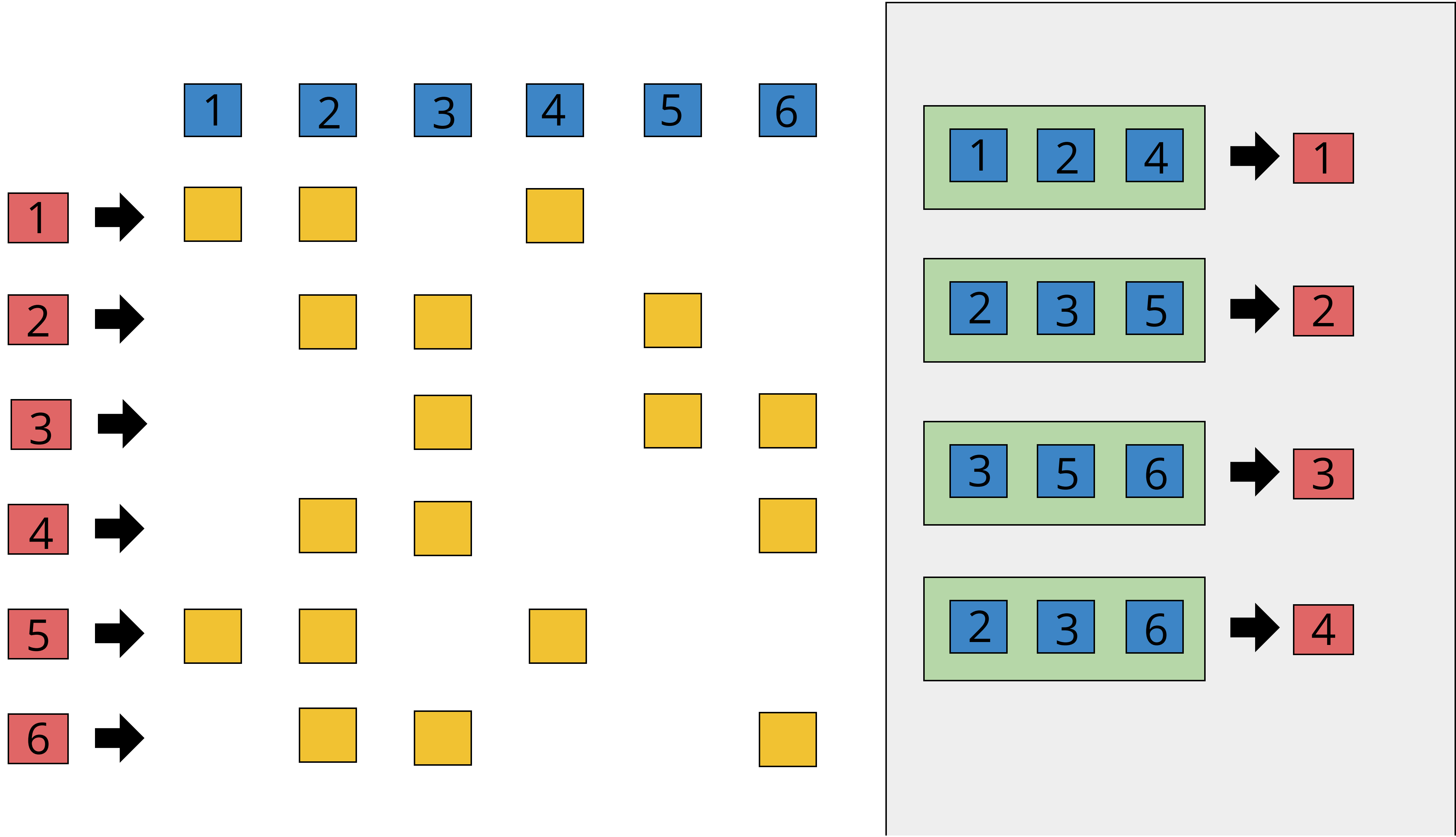


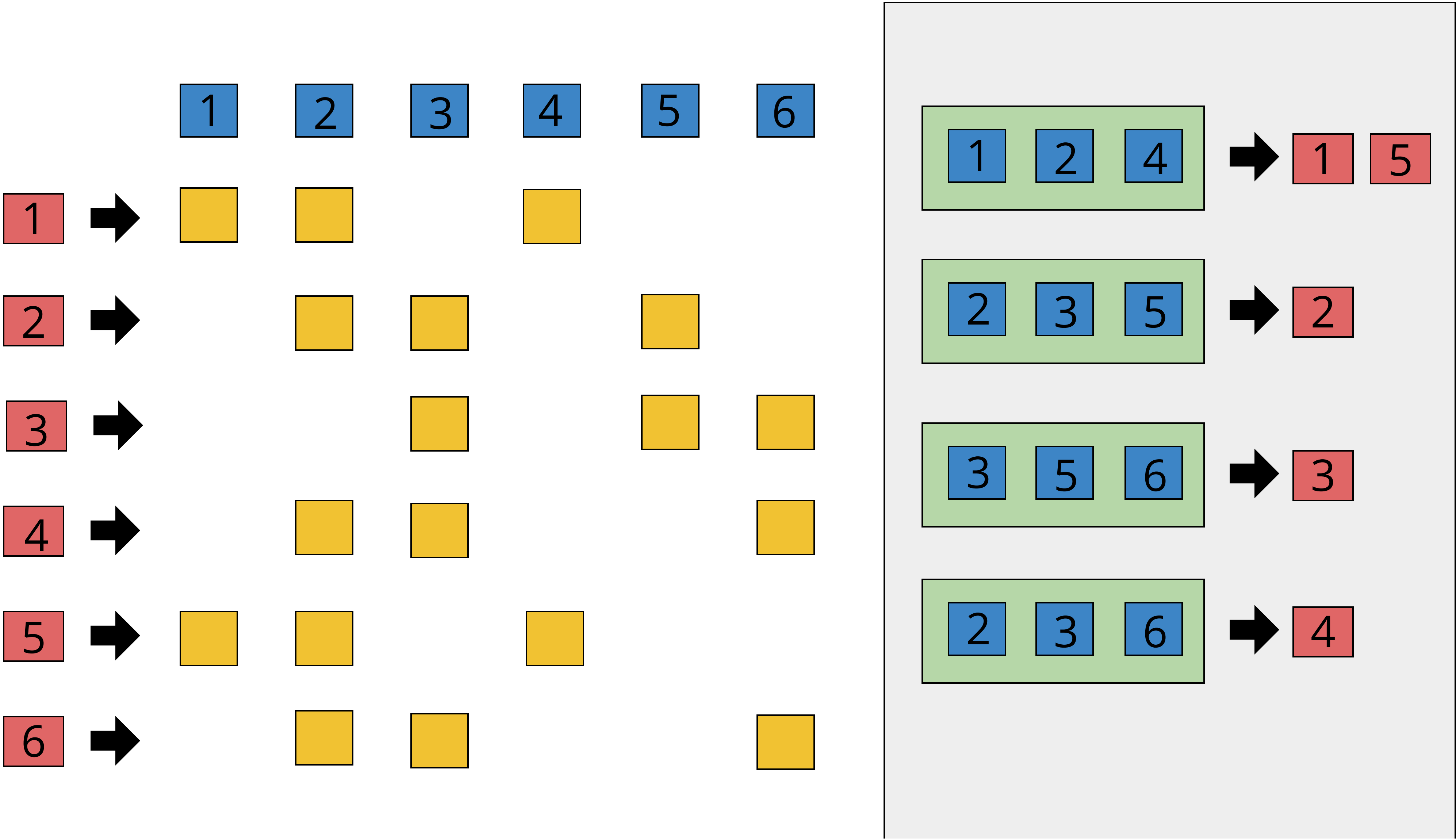

1
2
3
4
5
6
1
2
3
4
5
6

1
2
4
1

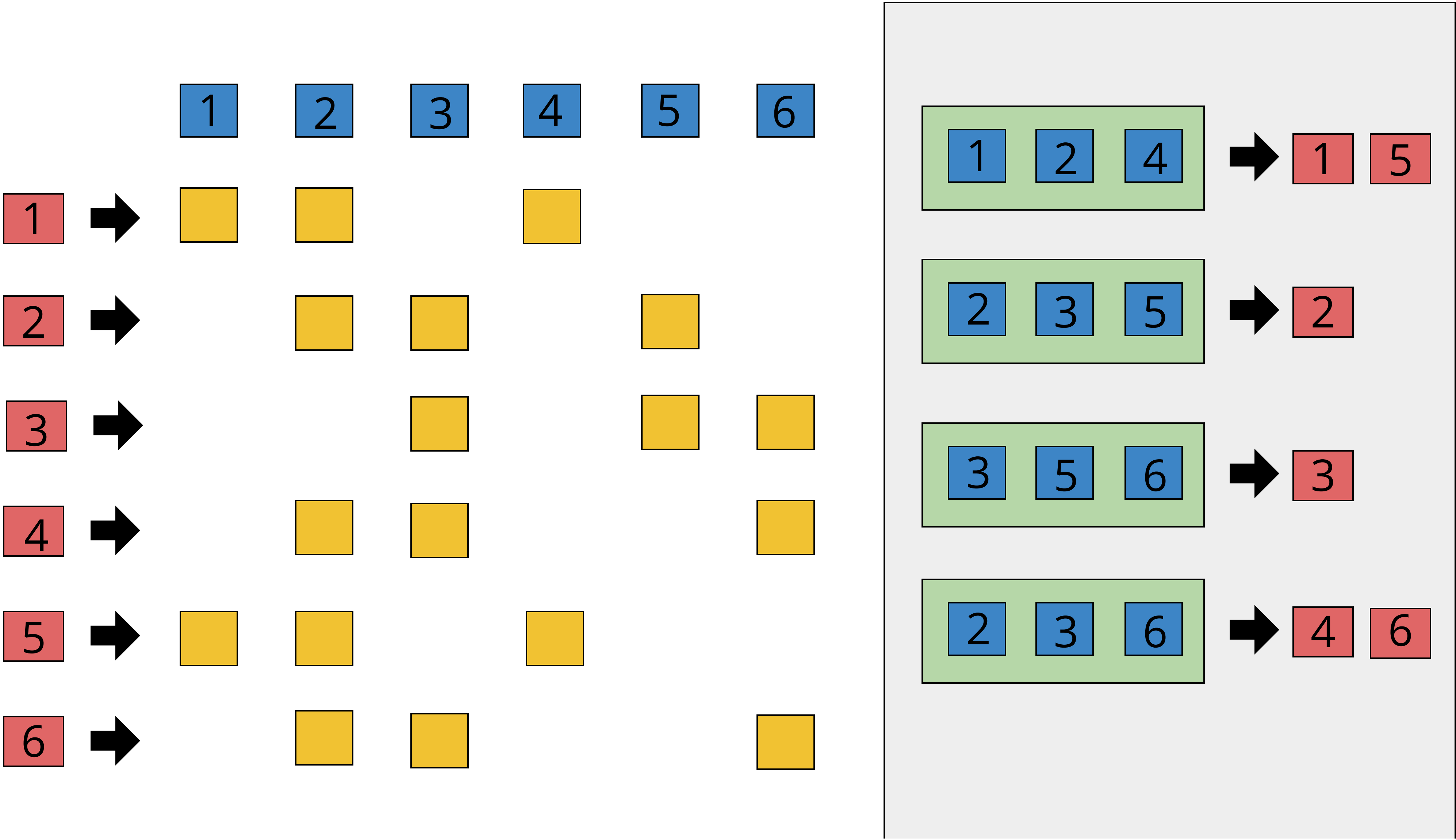


1
2
3
4
5
6
1
2
3
4
5
6
1
2
4
1
2
3
5
2

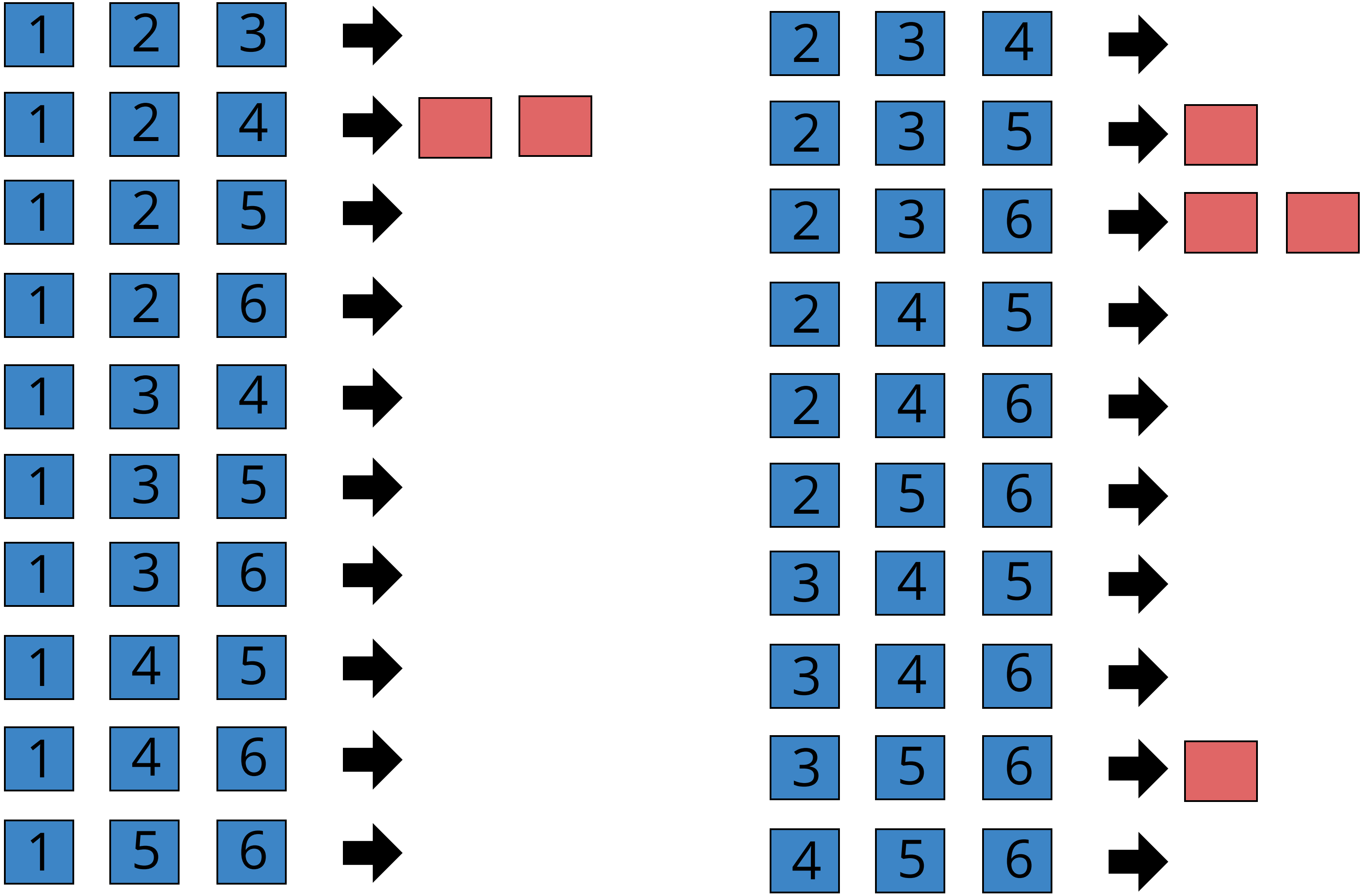

1 2 3 4 5 6
1
2
3
4
5
6
1 2 4
1
2 3 5
2
3 5 6
3

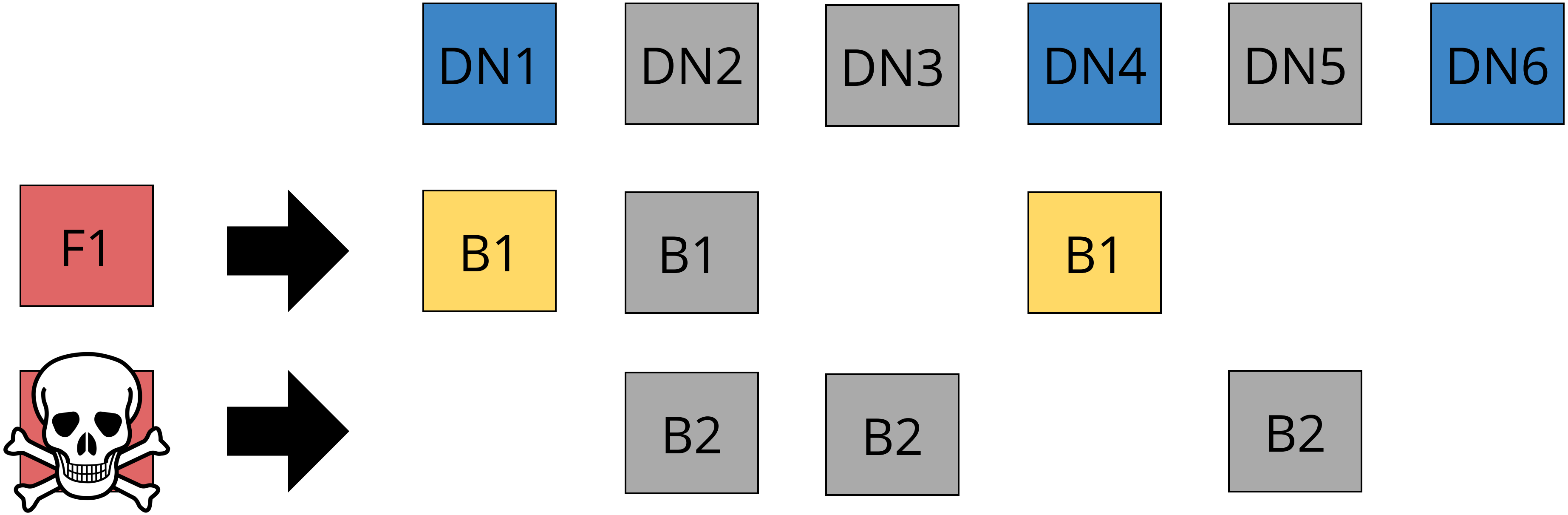

1 2 3 4 5 6
1
2
3
4
5
6
1 2 4
1
2 3 5
2
3 5 6
3
2 3 6
4

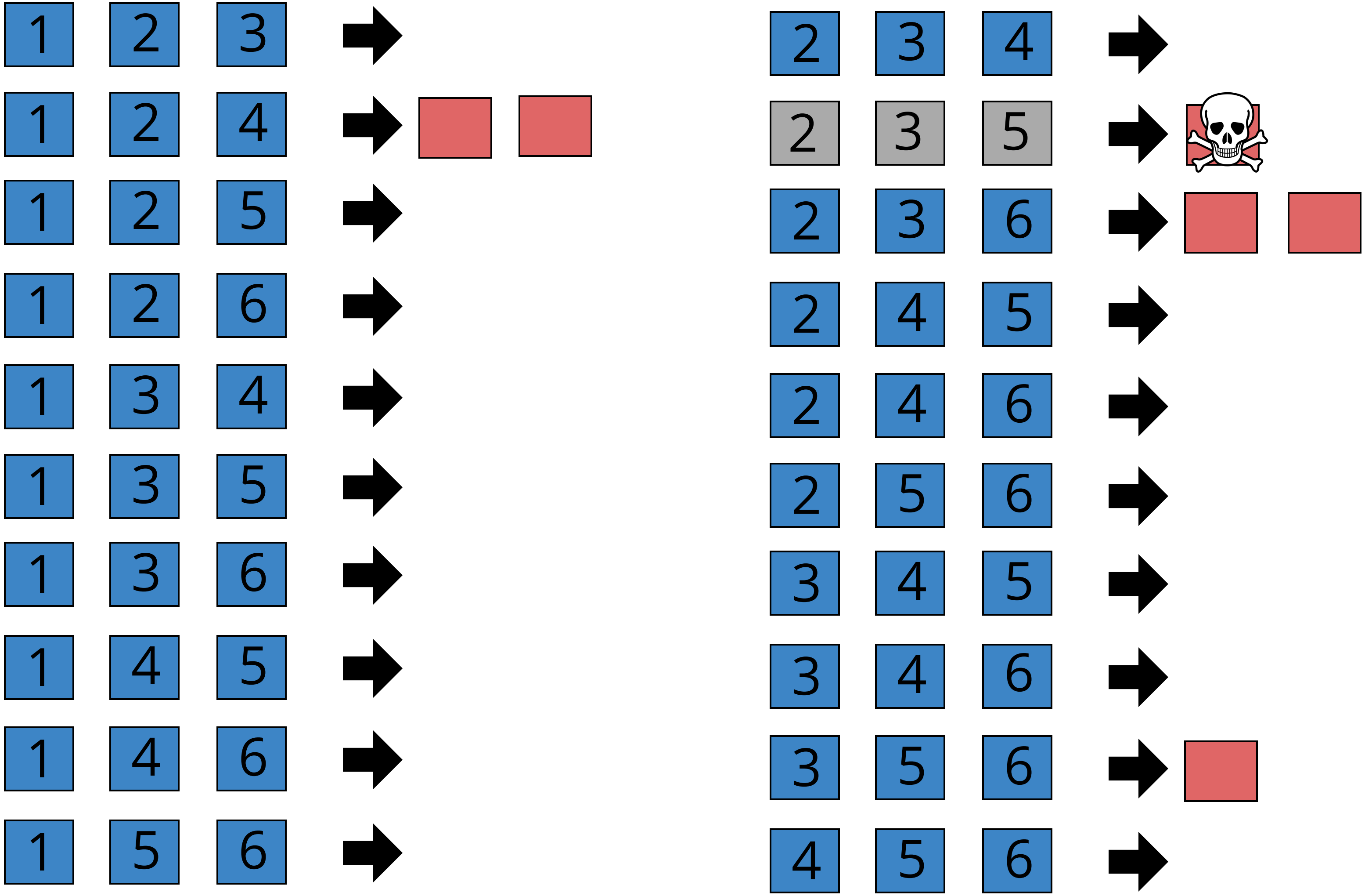

1 2 3 4 5 6
1
2
3
4
5
6
1 2 4
1 5
2 3 5
2
3 5 6
3
2 3 6
4



1
2
3
4
5
6
1
2
3
4
5
6
1 2 4
1 5
2 3 5
2
3 5 6
3
2 3 6
4 6


1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
1 3 6
1 4 5
1 4 6
1 5 6
2 3 4
2 3 5
2 3 6
2 4 5
2 4 6
2 5 6
3 4 5
3 4 6
3 5 6
4 5 6


DN1
DN2
DN3
DN4
DN5
DN6
F1
B1
B1
B1
B2
B2
B2

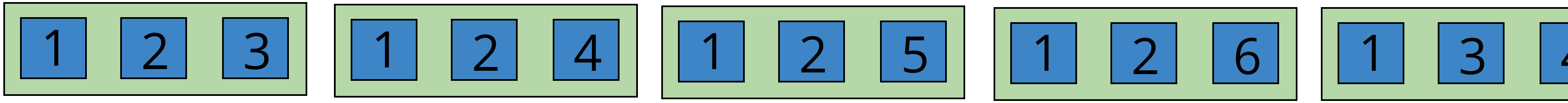

1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
1 3 6
1 4 5
1 4 6
1 5 6
2 3 4
2 3 5
2 3 6
2 4 5
2 4 6
2 5 6
3 4 5
3 4 6
3 5 6
4 5 6

failure of
3 independent
nodes

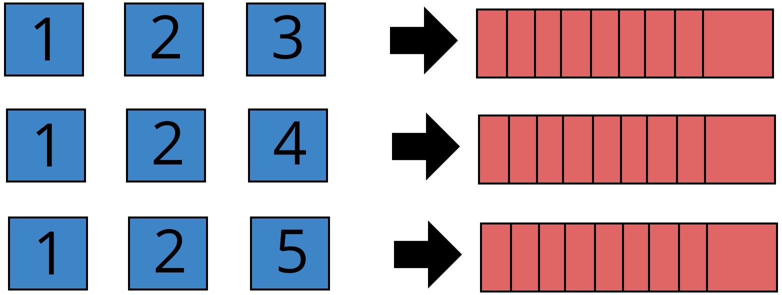
Kill one
Replicaset

Random replication

- 6 datanodes --> 20 different 3-node set: $\binom{6}{3} = 20$



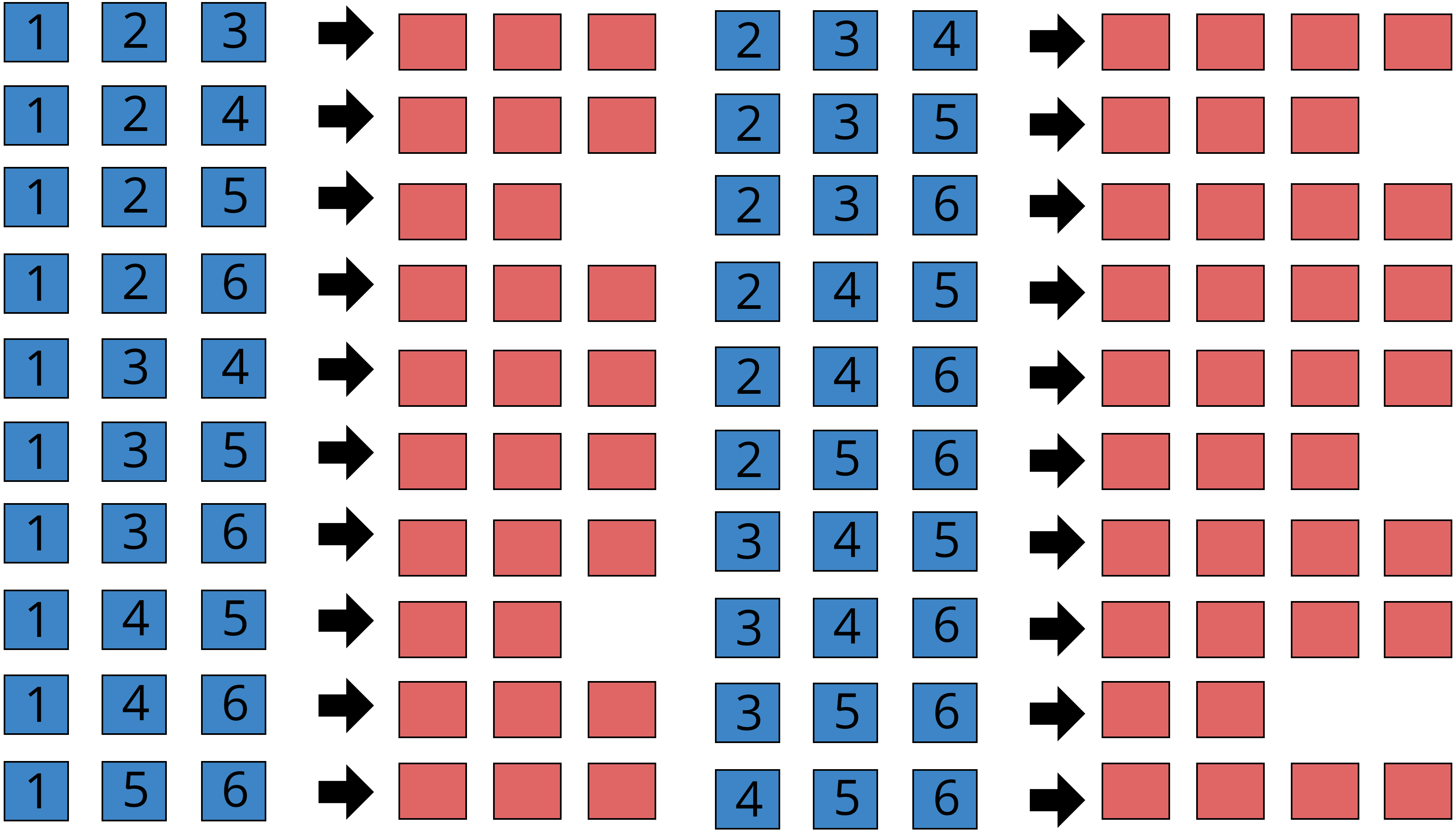
- 2000 blocks (20 * 100)

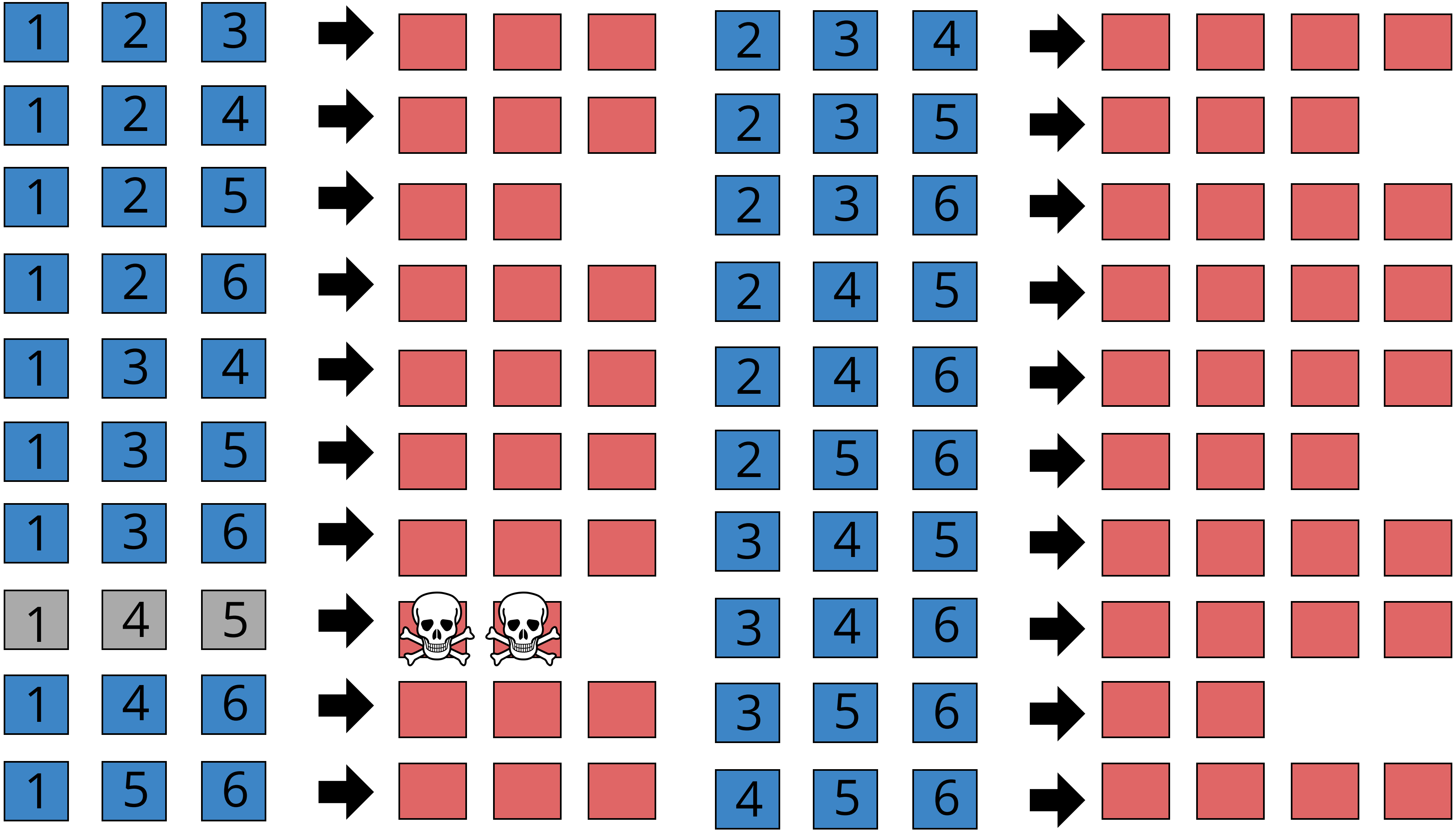


~100 blocks

~100 blocks

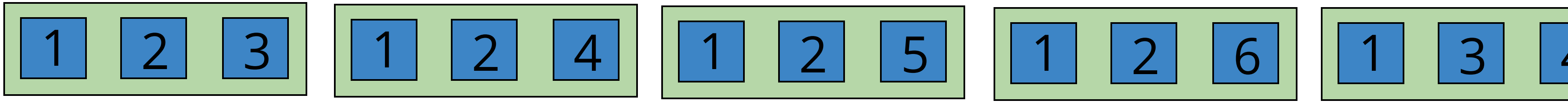
~100 blocks


1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
1 3 6
1 4 5
1 4 6
1 5 6
2 3 4
2 3 5
2 3 6
2 4 5
2 4 6
2 5 6
3 4 5
3 4 6
3 5 6
4 5 6

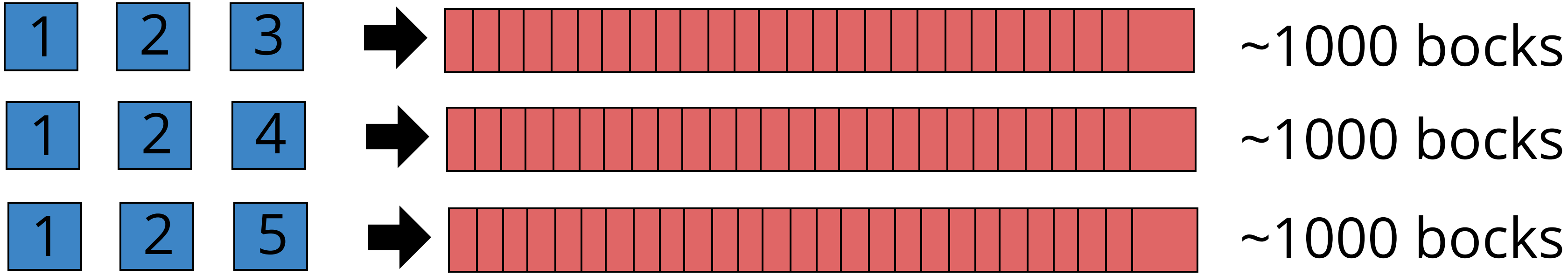

1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
1 3 6
1 4 5
1 4 6
1 5 6
2 3 4
2 3 5
2 3 6
2 4 5
2 4 6
2 5 6
3 4 5
3 4 6
3 5 6
4 5 6

Scale it up? Doesn't help

- 600 datanodes --> $\binom{600}{3} = 35820200$ different 3-node replicaset

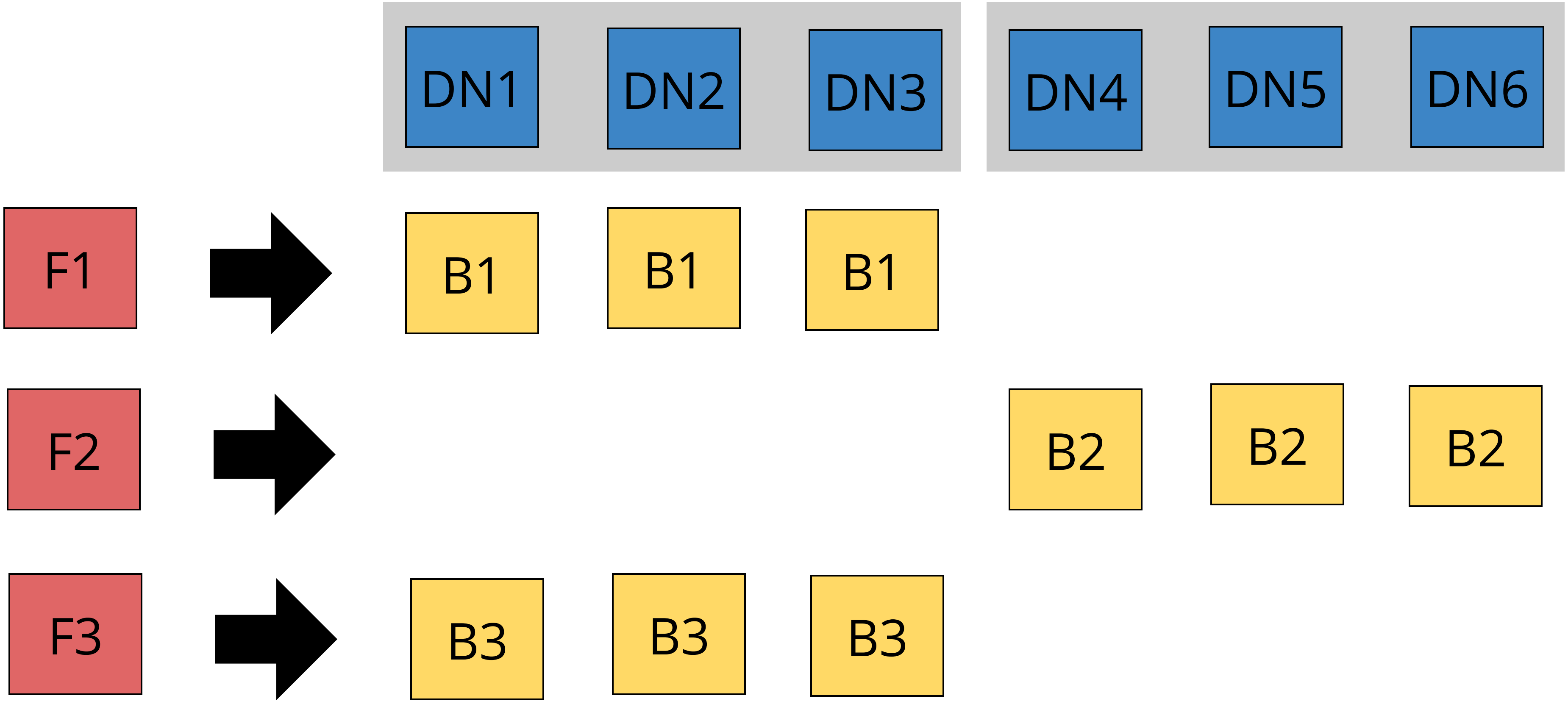


- 35 000 000 000 blocks (1000 * 35 000 000)

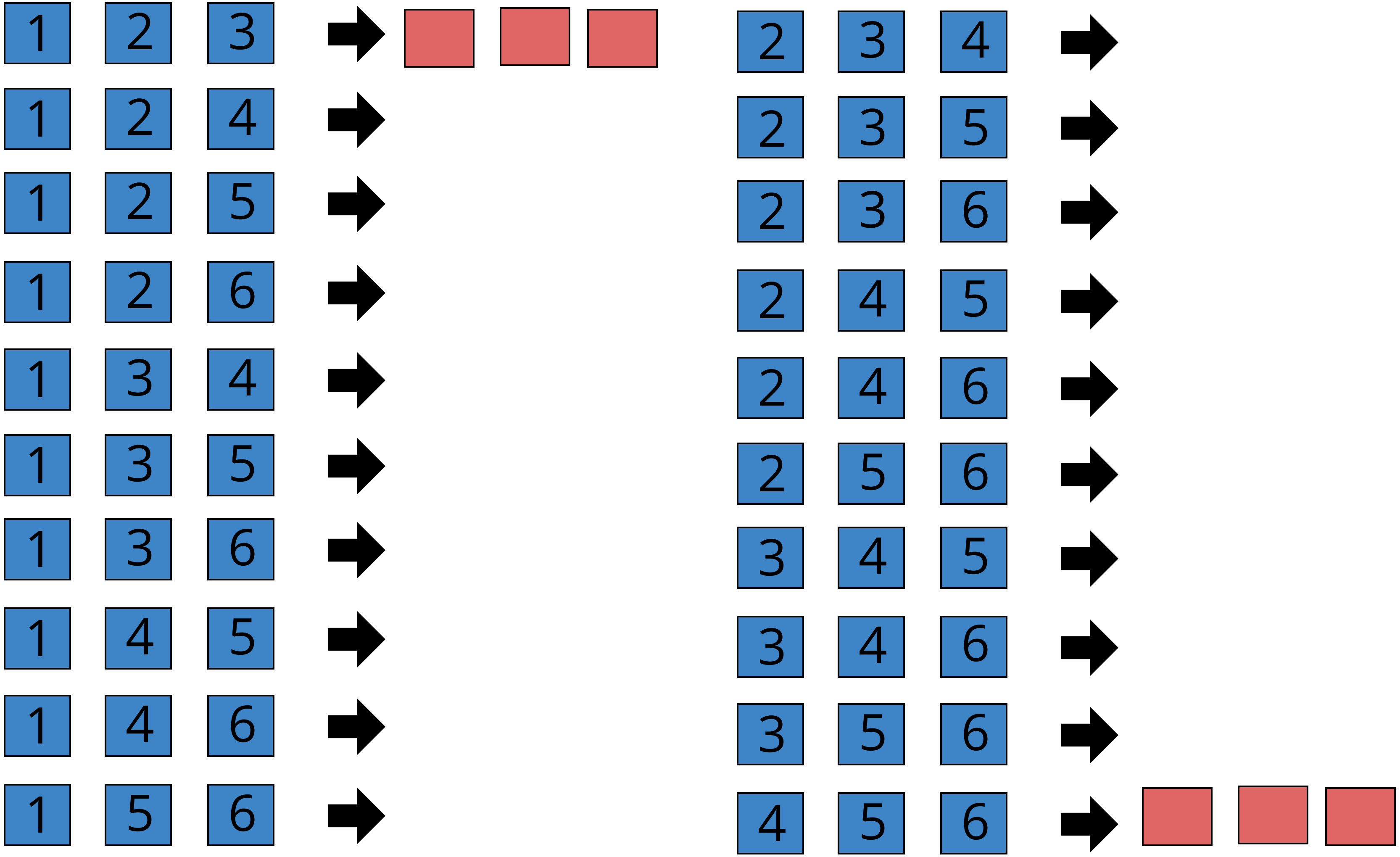


?

How can we do it better?



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


1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
1 3 6
1 4 5
1 4 6
1 5 6
2 3 4
2 3 5
2 3 6
2 4 5
2 4 6
2 5 6
3 4 5
3 4 6
3 5 6
4 5 6

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

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

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

2 groups

10% chance

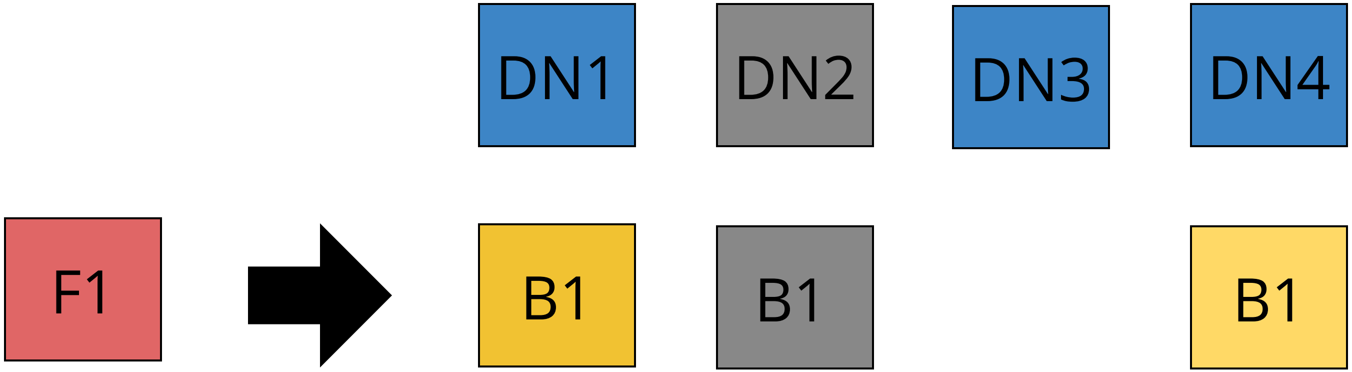
50% loss

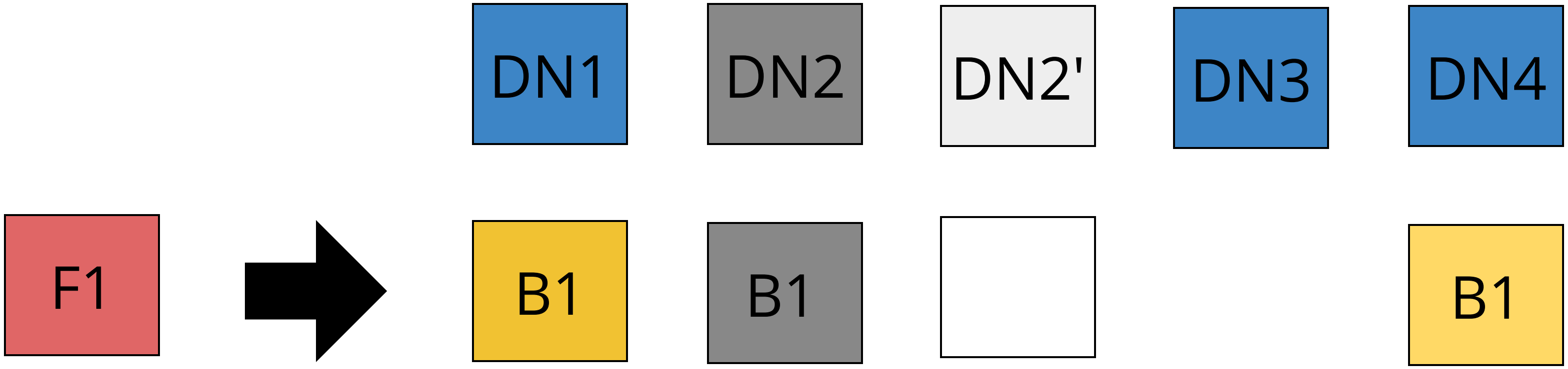
Random

100% chance

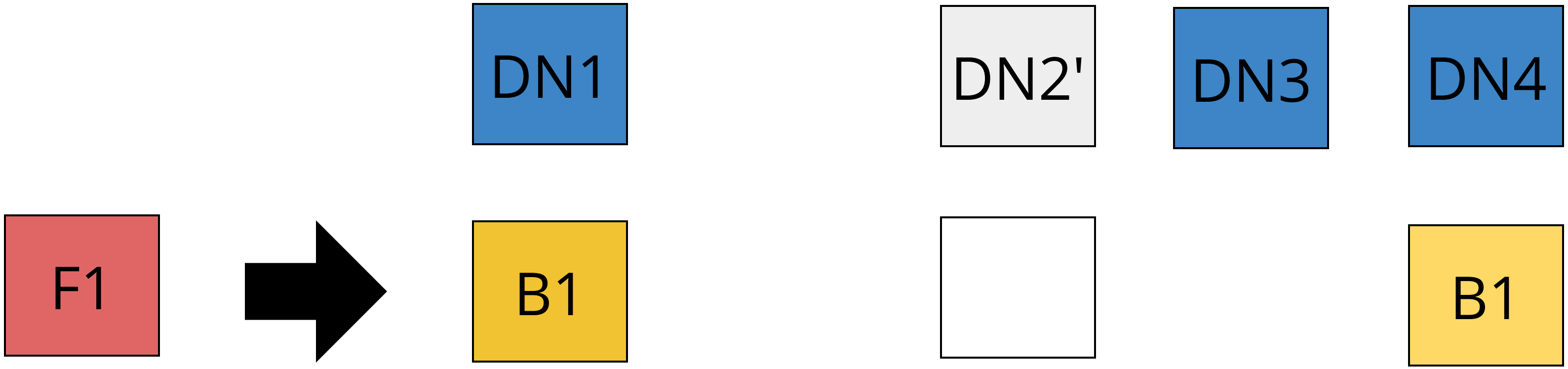
10% loss

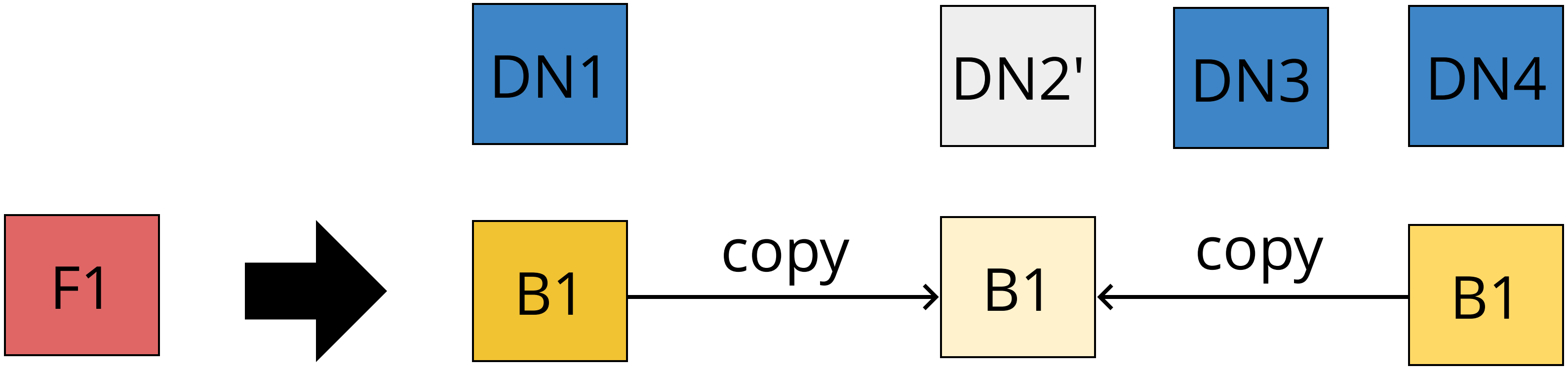
Problem?

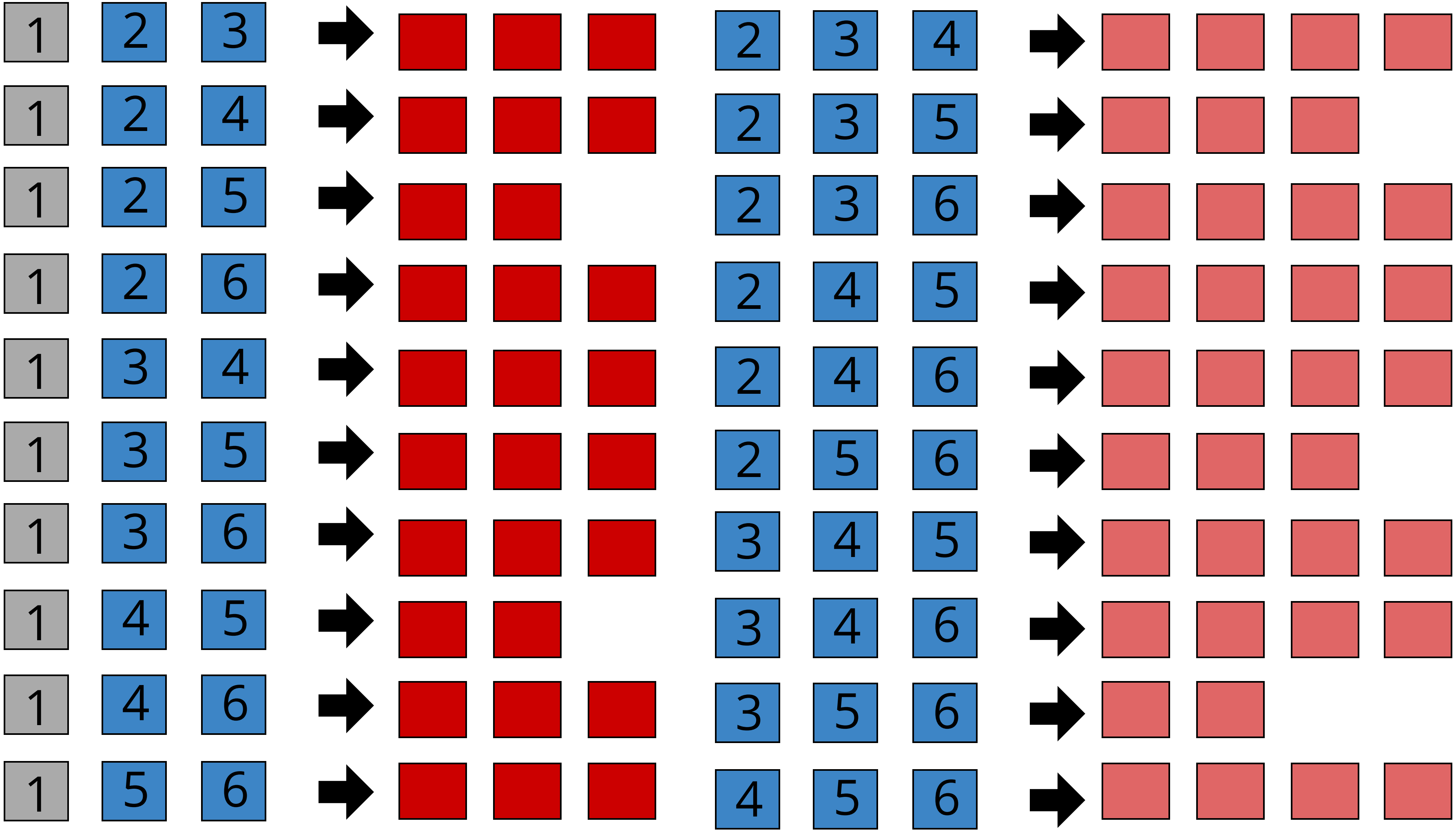

DN1
DN2
DN3
DN4
F1
B1
B1
B1

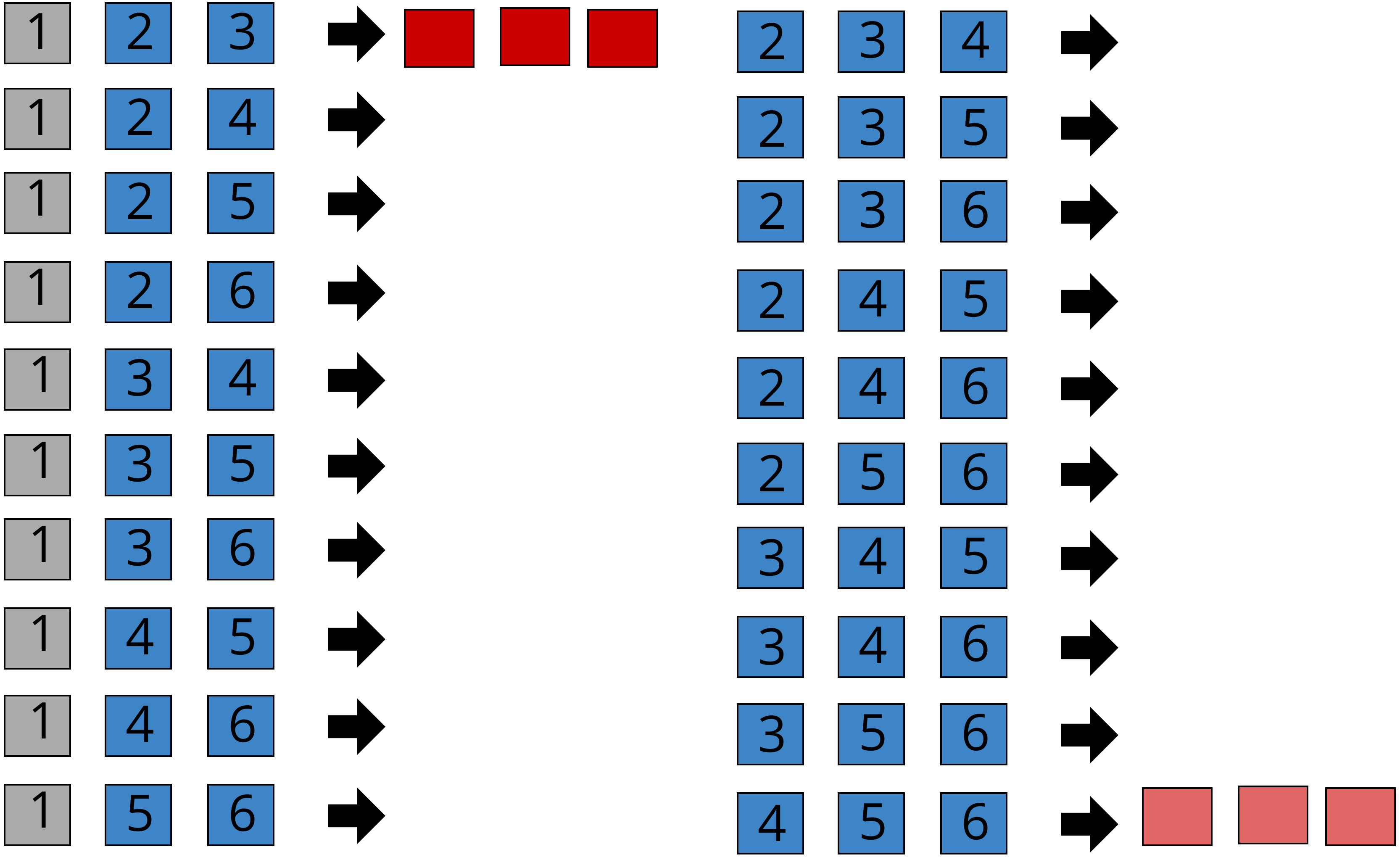


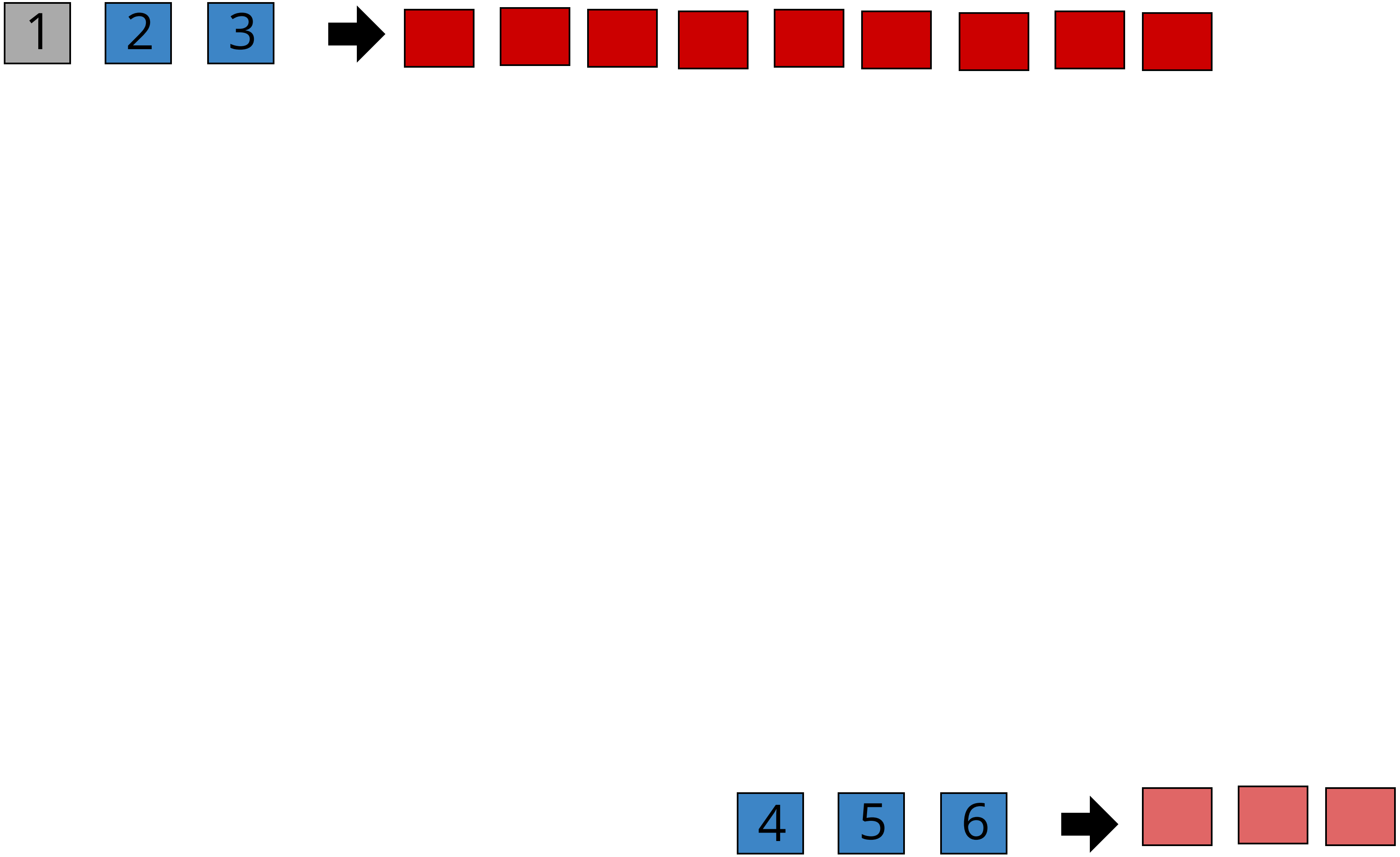
DN1
DN2
DN2'
DN3
DN4
F1
B1
B1
B1


DN1
DN2'
DN3
DN4
F1
B1
B1

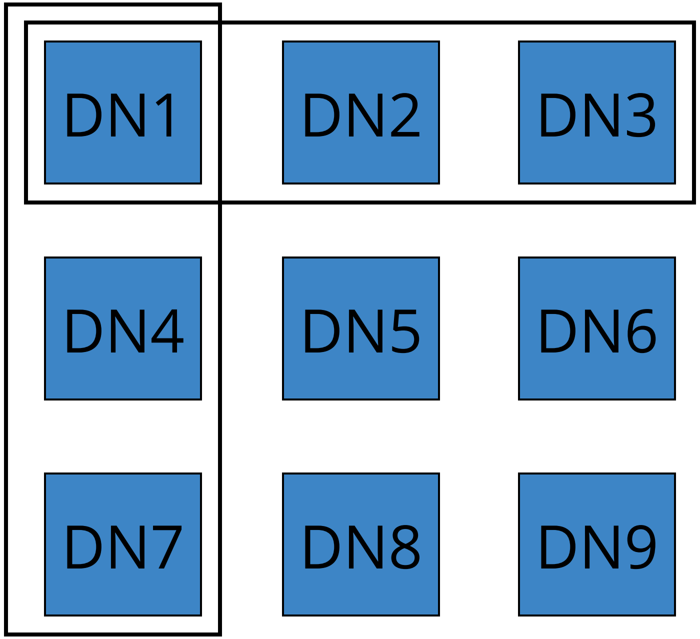

DN1
DN2'
DN3
DN4
F1
B1
copy
B1
copy
B1


1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
1 3 6
1 4 5
1 4 6
1 5 6
2 3 4
2 3 5
2 3 6
2 4 5
2 4 6
2 5 6
3 4 5
3 4 6
3 5 6
4 5 6


1 2 3
1 2 4
1 2 5
1 2 6
1 3 4
1 3 5
1 3 6
1 4 5
1 4 6
1 5 6
2 3 4
2 3 5
2 3 6
2 4 5
2 4 6
2 5 6
3 4 5
3 4 6
3 5 6
4 5 6


1
2
3
4
5
6

CopySets


DN1
DN2
DN3
DN4
DN5
DN6
DN7
DN8
DN9

 # replicasets:

6 groups: 3 col, 3 row

3 random node failures:

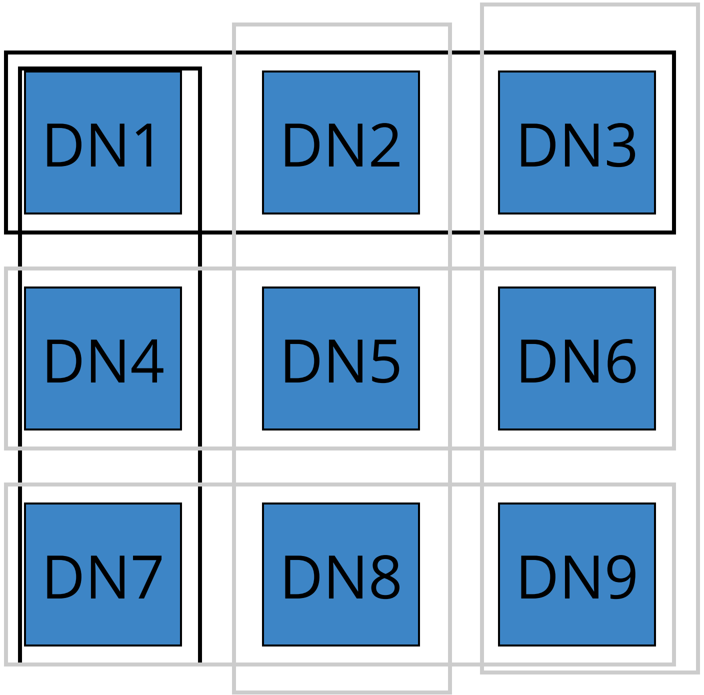
possible in **84 ways**

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 ❤ small files)
- Easy to use and run in containerized environments

Q&A

S3 protocol

Hadoop FS

CSI

Apache Hadoop Ozone

THE APACHE SOFTWARE FOUNDATION

hadoop.apache.org/ozone

Note: in-place upgrade

- Upgrade path from existing HDFS
- **without (or minimal) Data Move**
- The math is the same
  - Grouping block in the **same** replicaset
  - Report them in groups (containers)
  - Better: Less replicasets with more blocks in each
- Smart preprocessing/balancing may be required