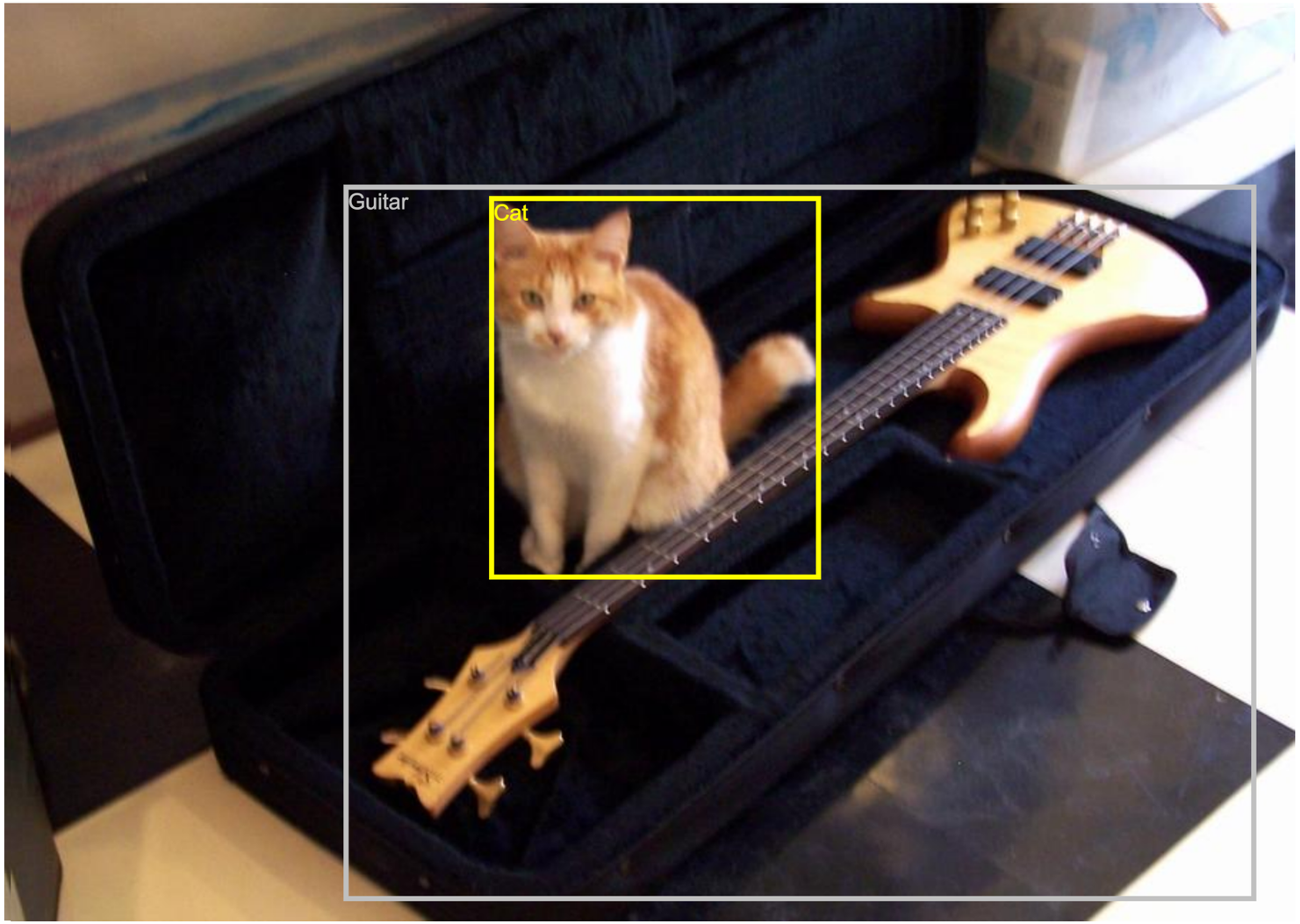
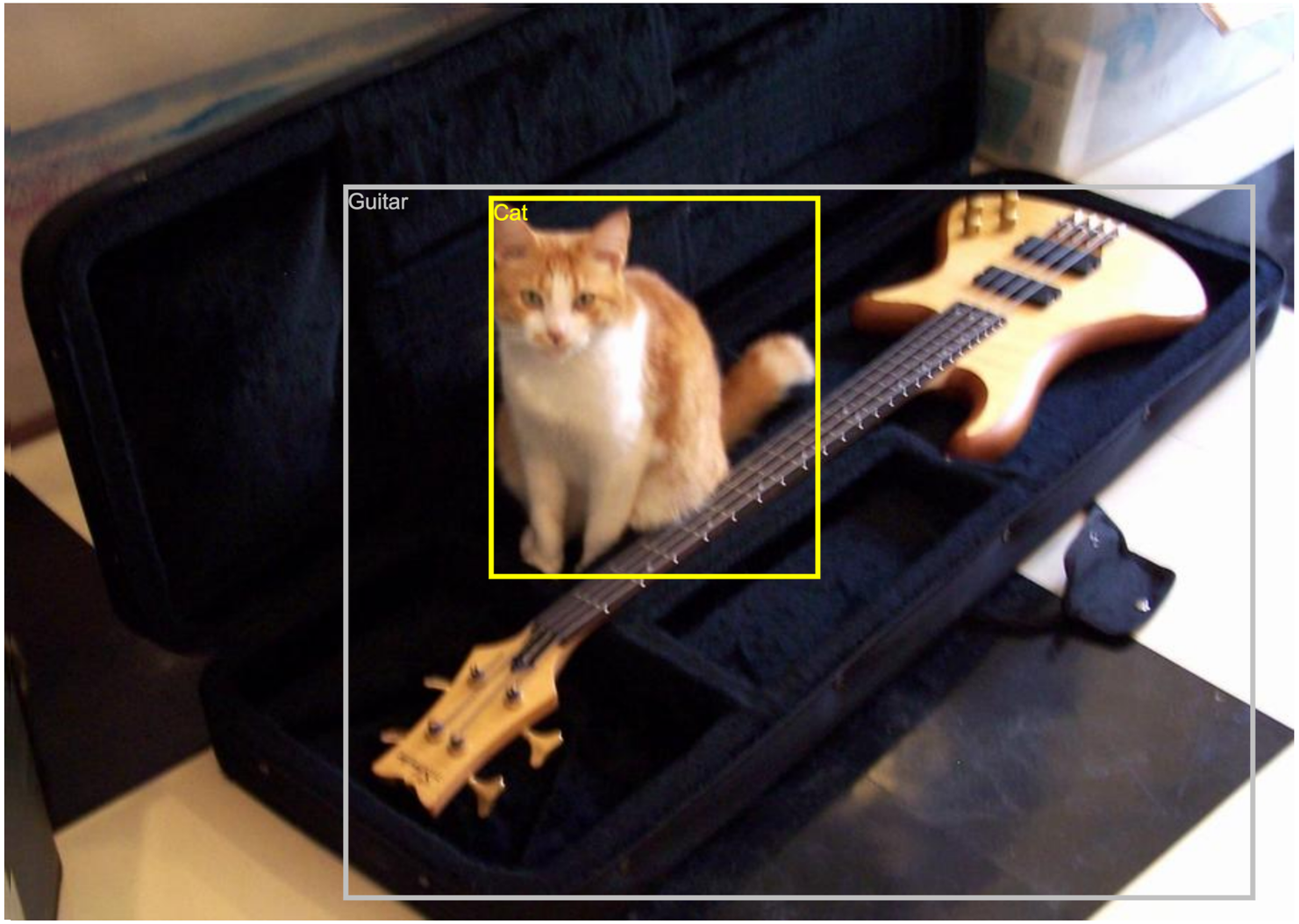
HopsFS & ePipe

Mahmoud Ismail
KTH

Distributed Computing and Analytics Workshop, September 26th 2018

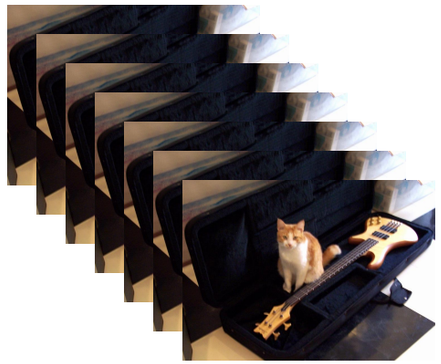

Guitar
Cat



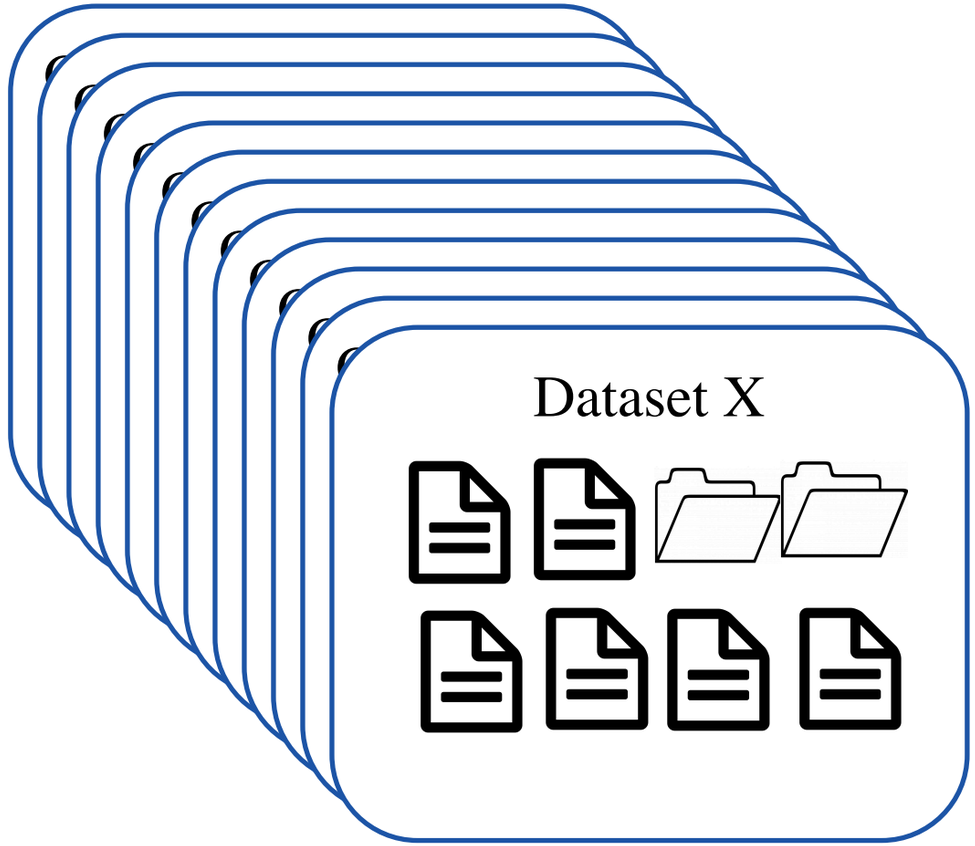
Problem: Data layer to store millions of these images and their annotations

At Scale

Open Images Dataset



At Scale

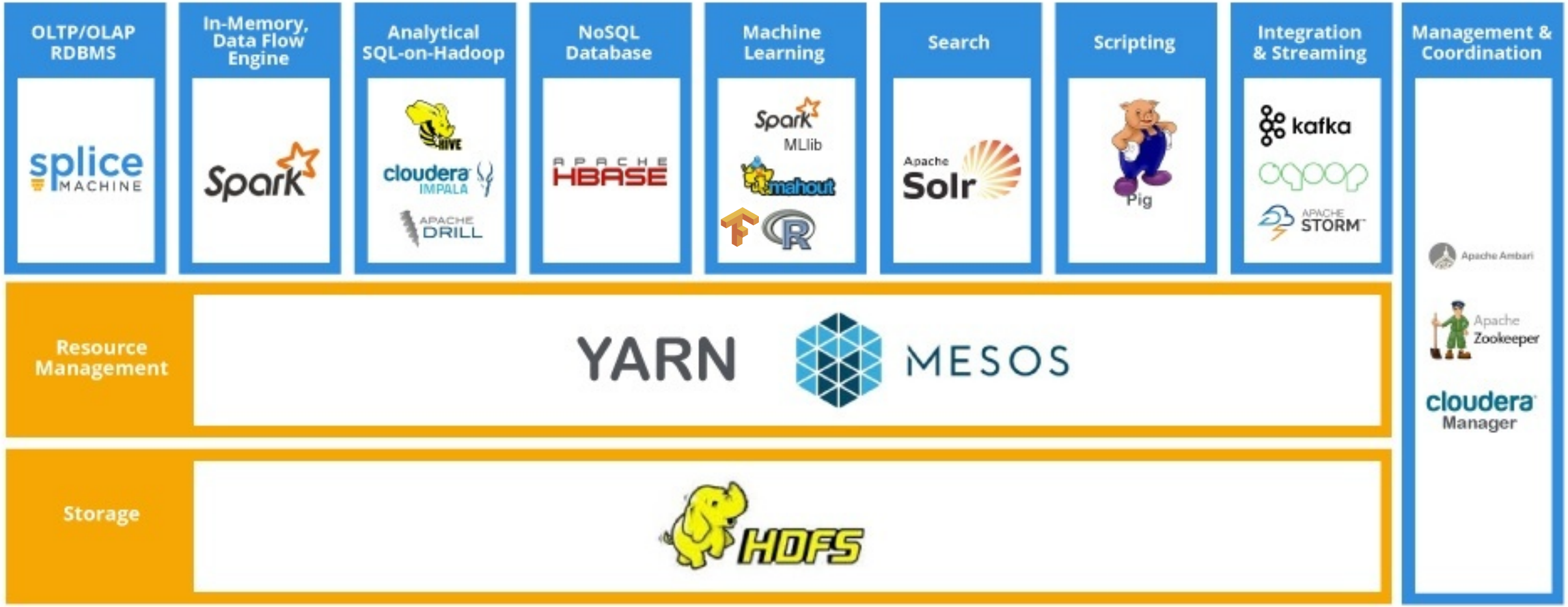


Requirements

- Reading/Writing millions of images with high throughput
- Attaching annotations to each image, and then searching using these annotations

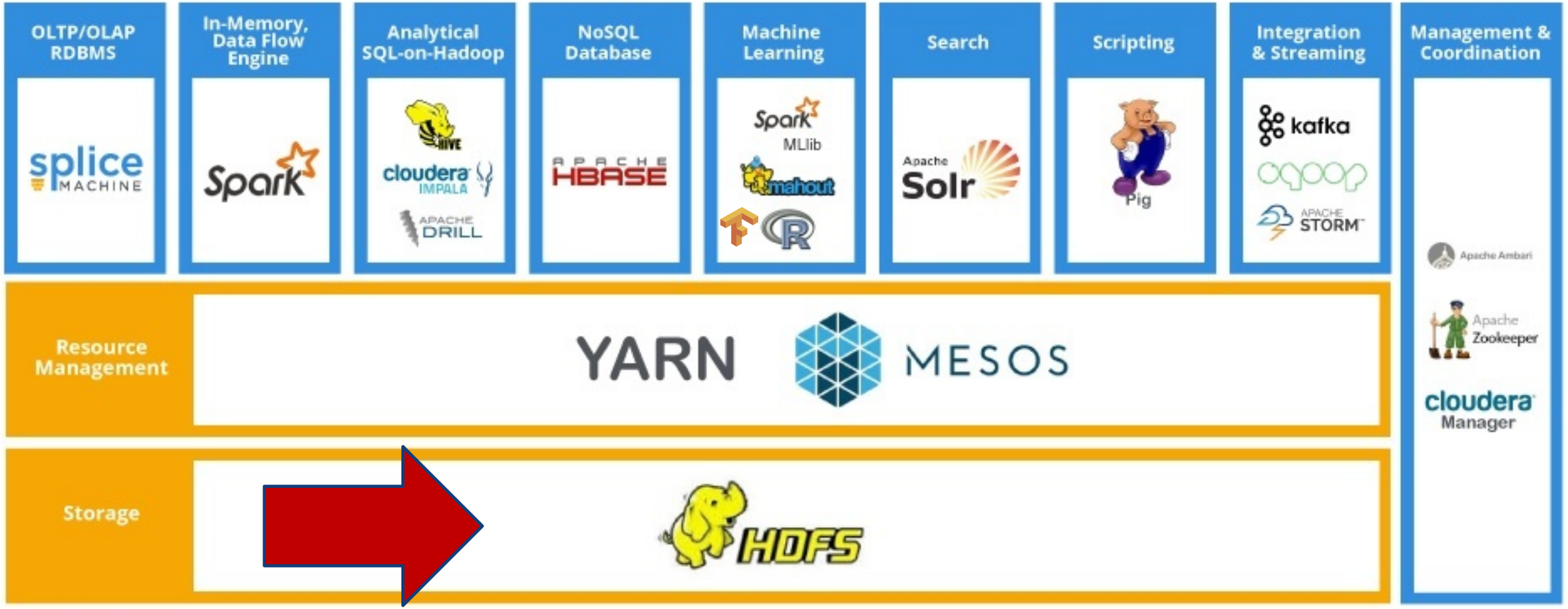
HDFS

HDFS



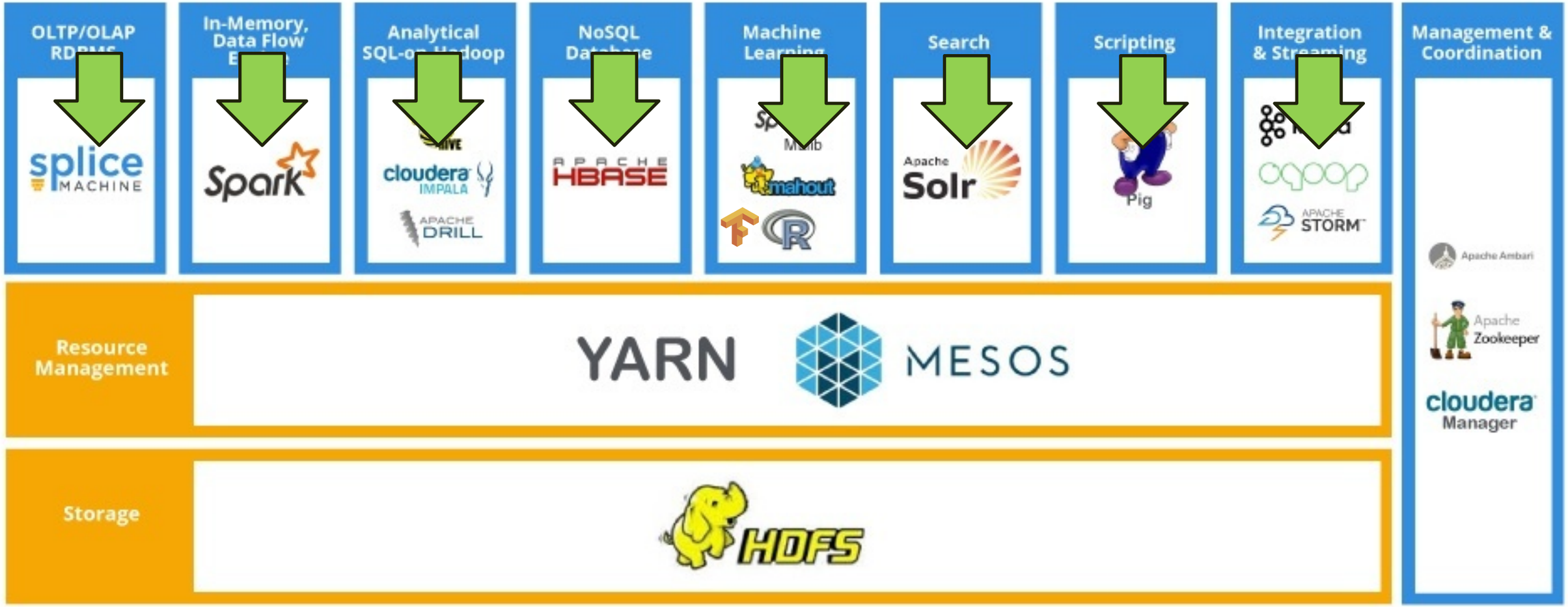
Hadoop Software Stack

HDFS



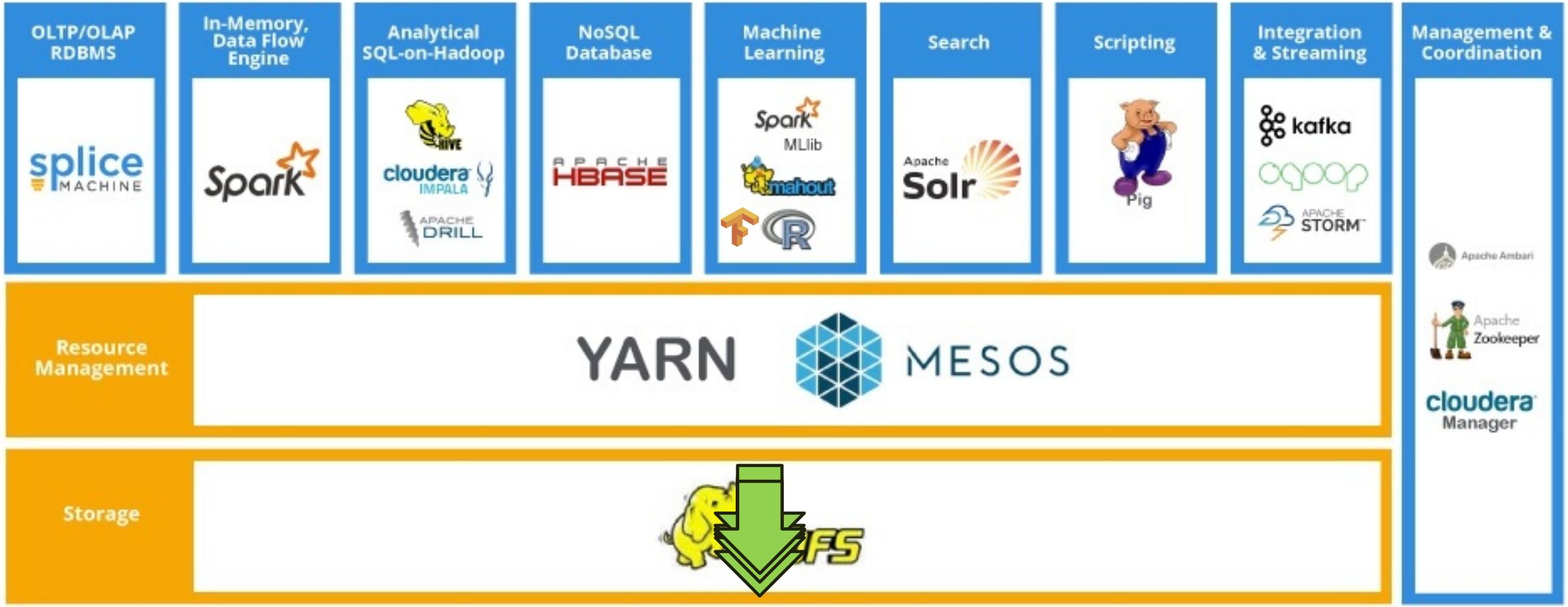
Hadoop Software Stack

HDFS



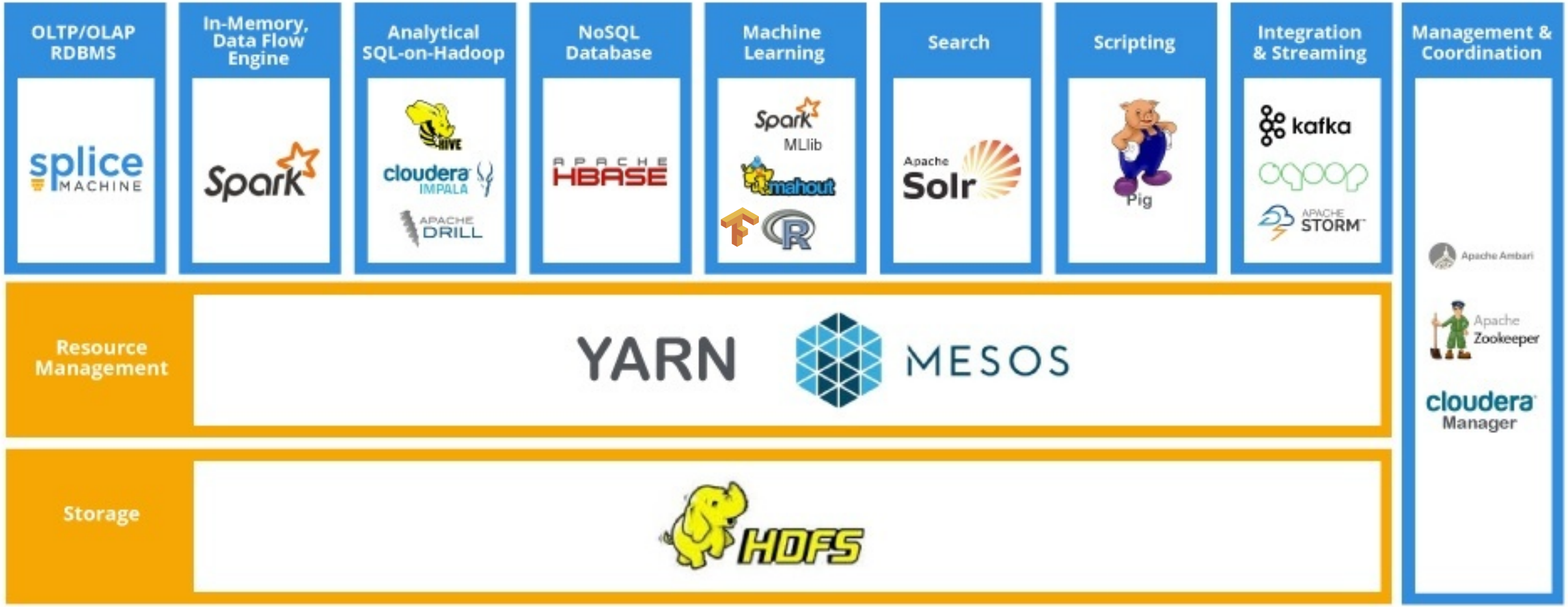
Hadoop Software Stack

HDFS



Hadoop Software Stack

HDFS



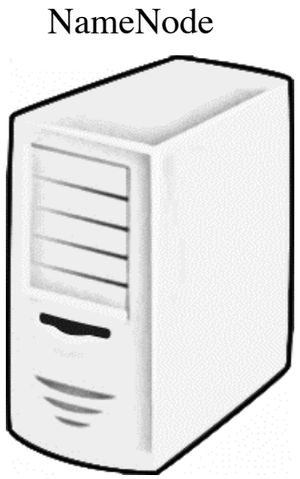
Hadoop Software Stack

HDFS Architecture

HDFS Architecture

DataNodes

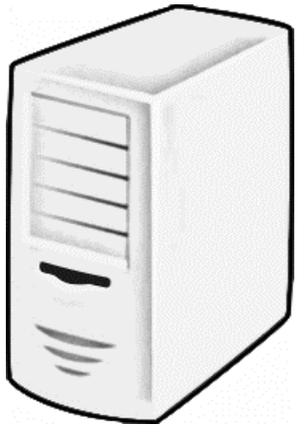
HDFS Architecture



HDFS Architecture

HDFS Client

NameNode



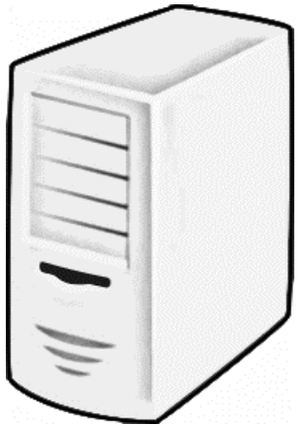
DataNodes

HDFS Architecture

HDFS Client

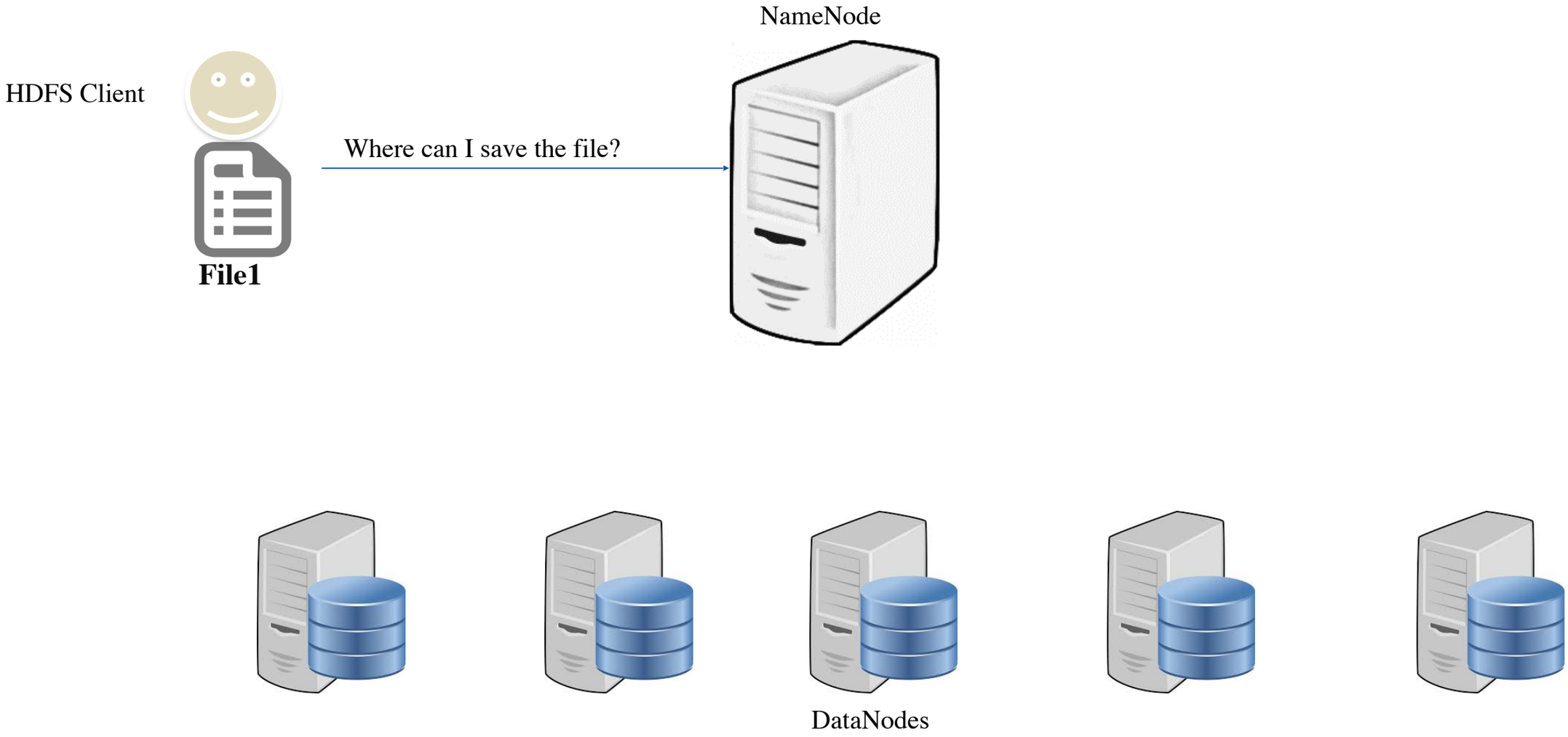
File1

NameNode

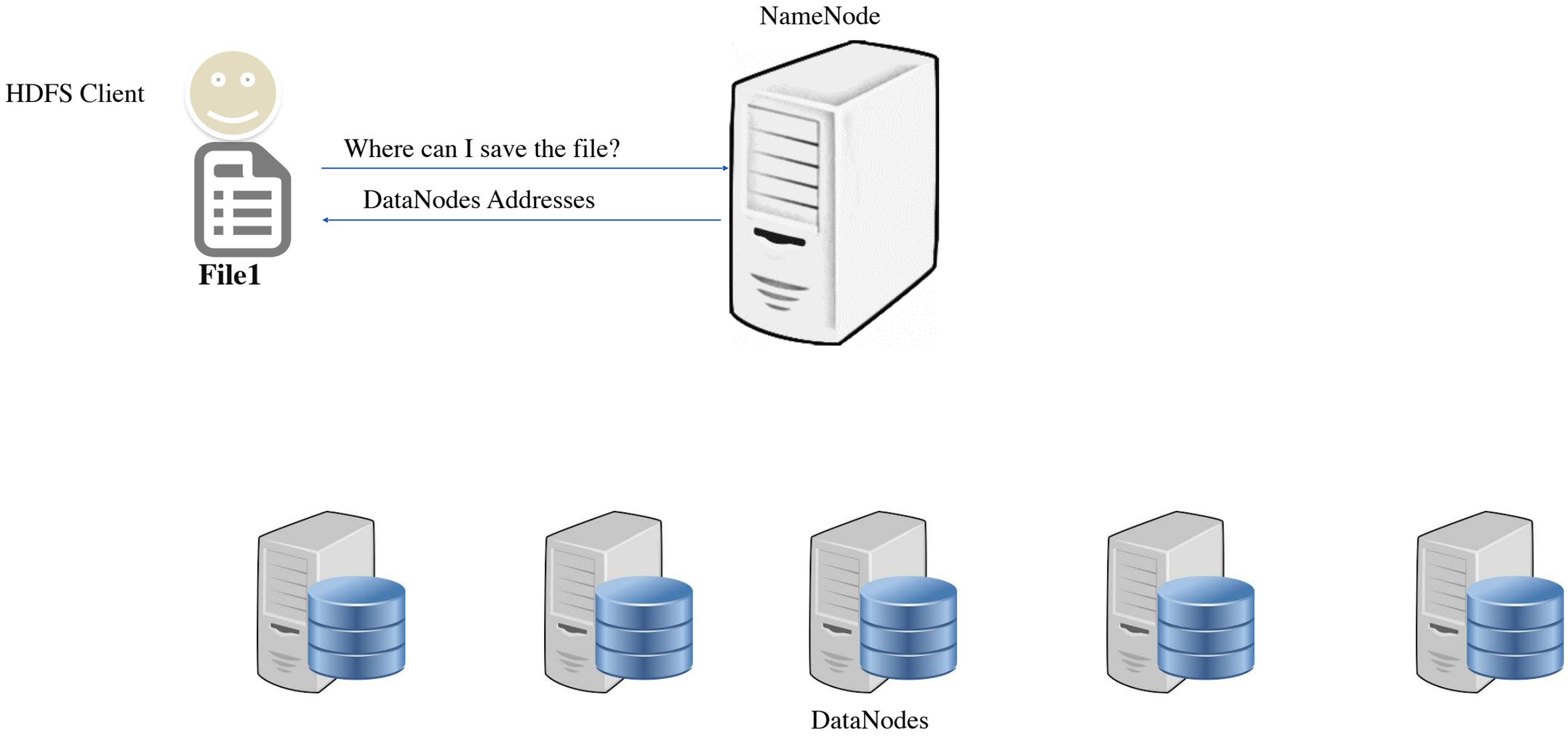


DataNodes

HDFS Architecture



HDFS Architecture

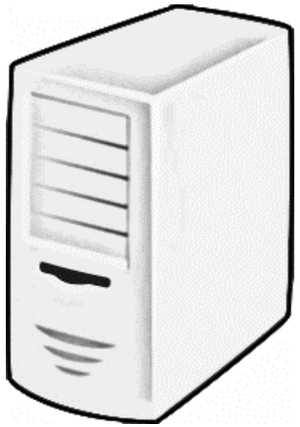


HDFS Architecture

HDFS Client

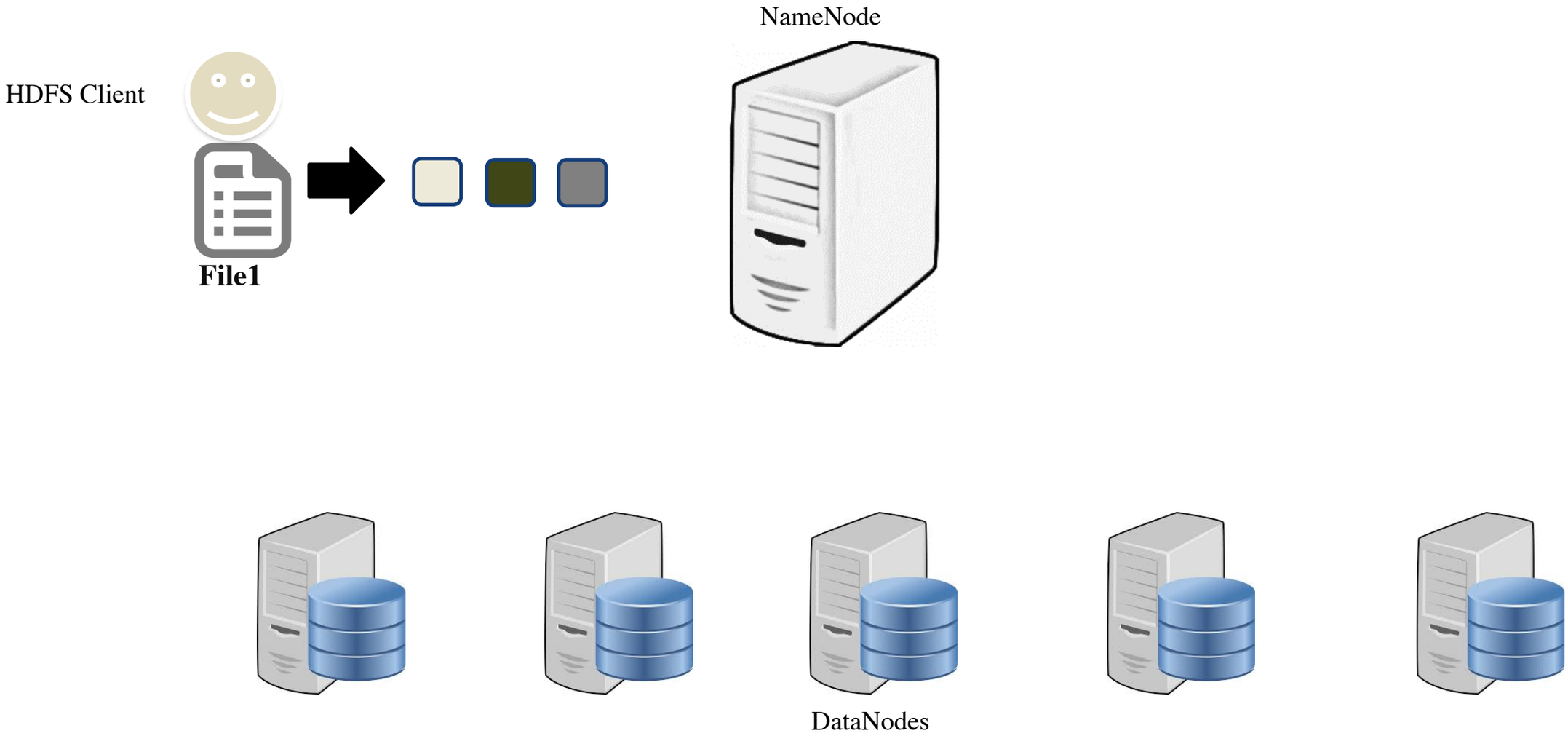
File1

NameNode



DataNodes

HDFS Architecture

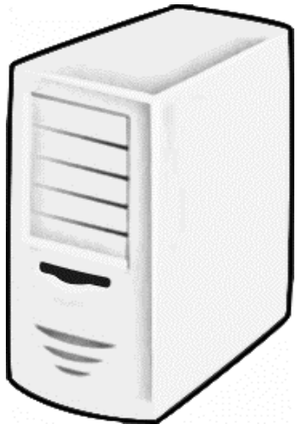


HDFS Architecture

HDFS Client

File1

NameNode



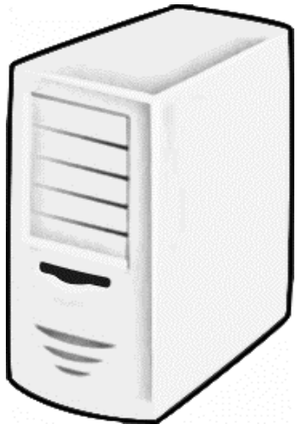
DataNodes

HDFS Architecture

HDFS Client

File1

NameNode



File System Metadata

File	Blocks Mappings

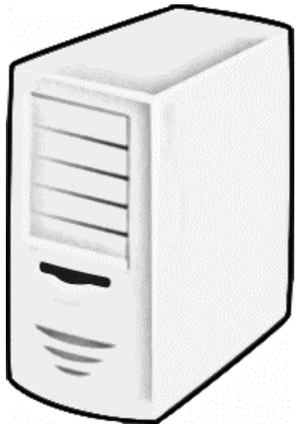




DataNodes

HDFS Architecture

NameNode



File System Metadata

File	Blocks Mappings
File1	Blk1 → DN1, Blk2 → DN5, Blk3 → DN3

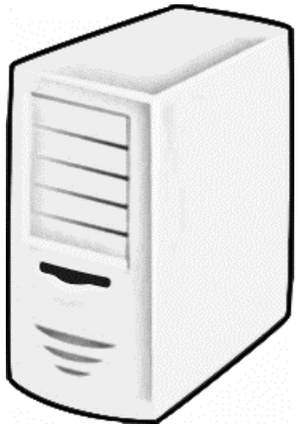




DataNodes

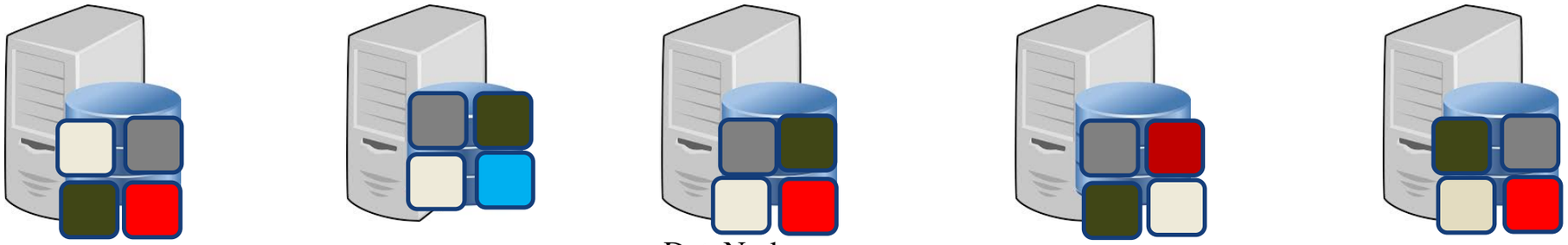
HDFS Architecture

NameNode



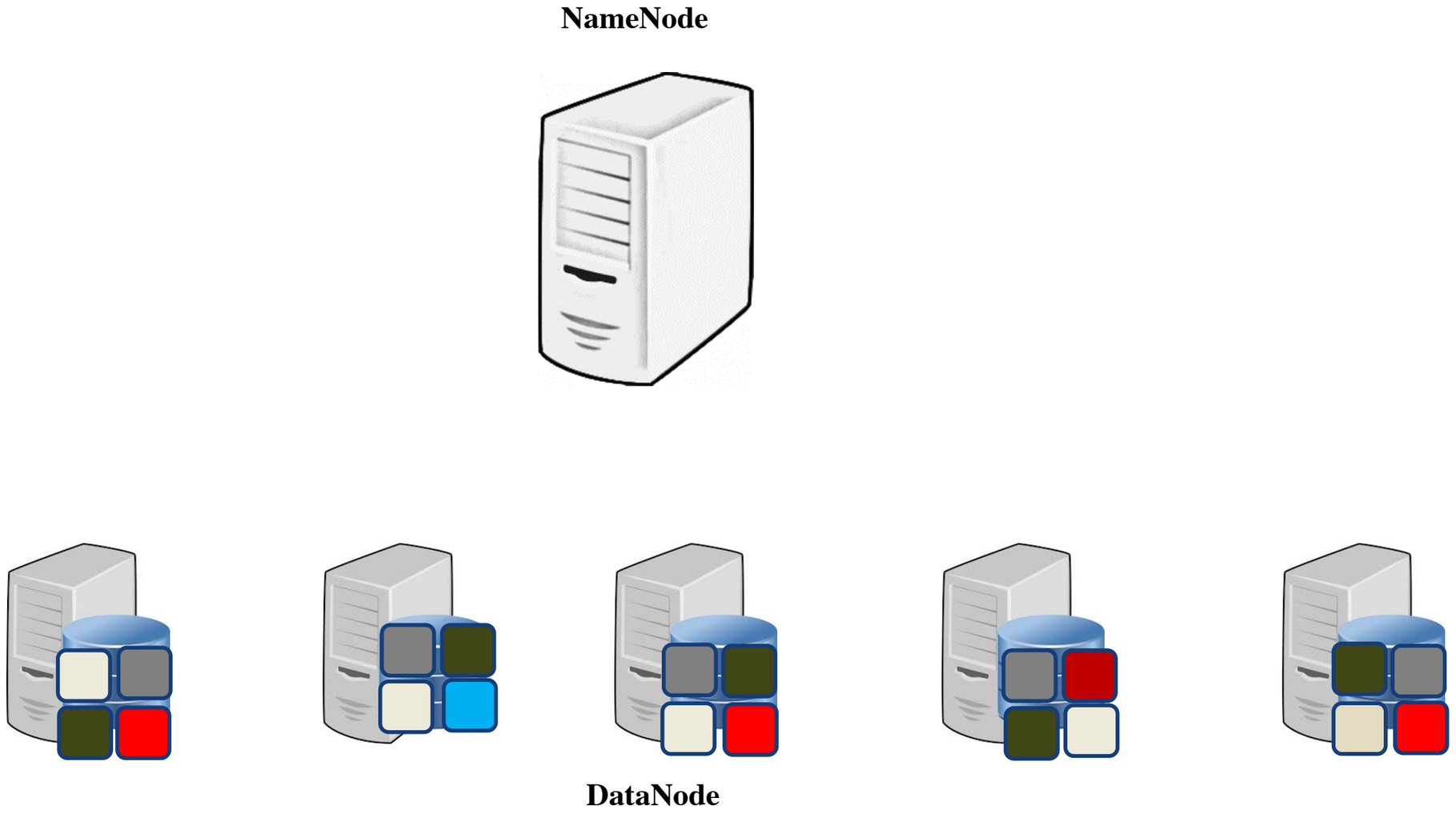
File System Metadata

File	Blocks Mappings
File1	Blk1 → DN1, Blk2 → DN5, Blk3 → DN3
File2	Blk1 → DN1, Blk2 → DN4
File3	Blk1 → DN1, Blk2 → DN2, Blk3 → DN3
File4	Blk1 → DN100
File5	Blk1 → DN4, Blk2 → DN2, Blk3 → DN9
...	...
FileN	Blk1 → DN2, Blk2 → DN8

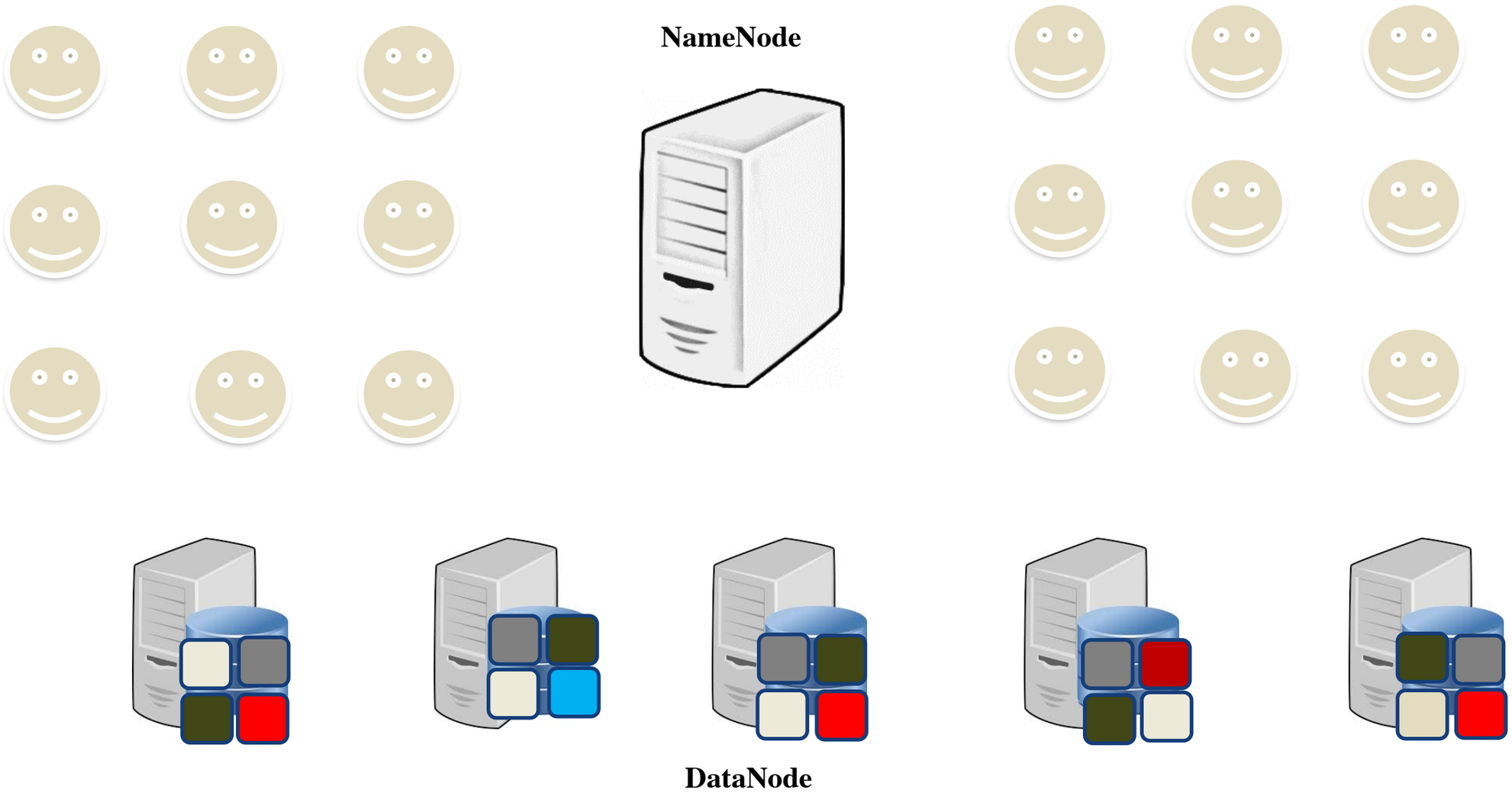


DataNodes

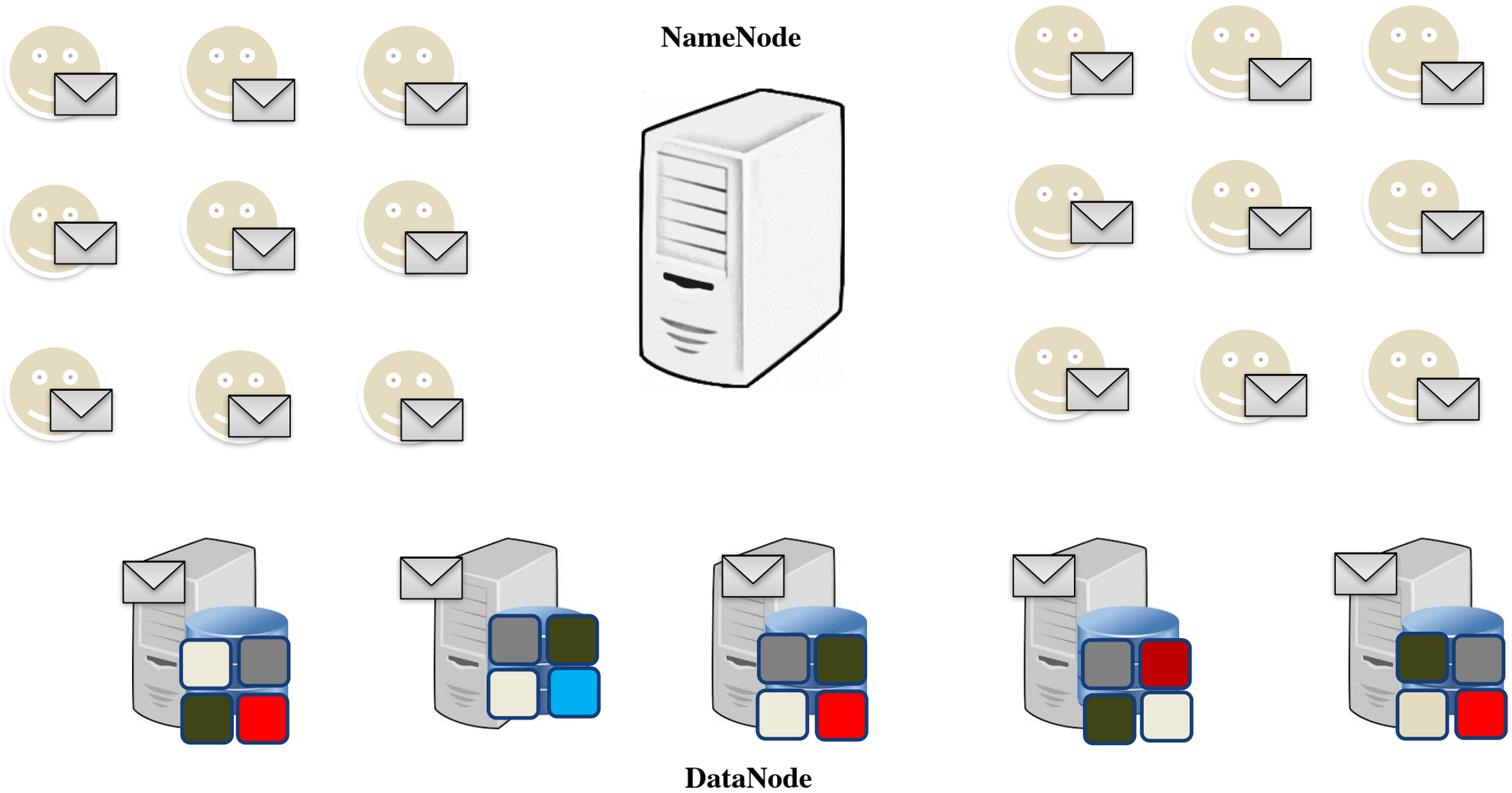
HDFS Performance at Scale



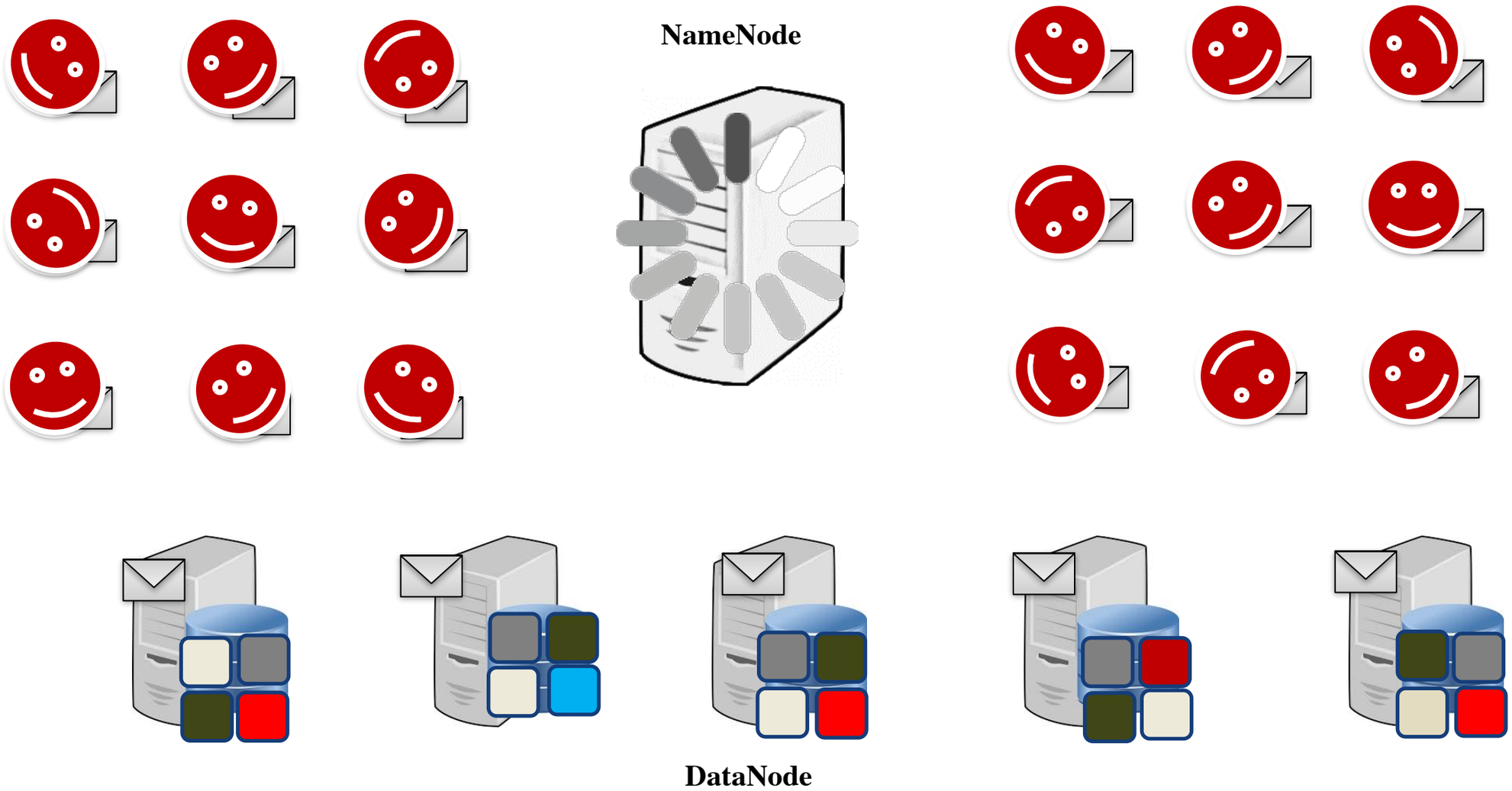
HDFS Performance at Scale



HDFS Performance at Scale



HDFS Performance at Scale



A problem has been detected and windows has been shut down to prevent damage
to your computer.

The problem seems to be caused by the following file: SPCMDCON.SYS

PAGE_FAULT_IN_NONPAGED_AREA

If this is the first time you've seen this Stop error screen,
restart your computer. If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x00000050 (0xFD3094C2,0x00000001,0xFBFE7617,0x00000000)

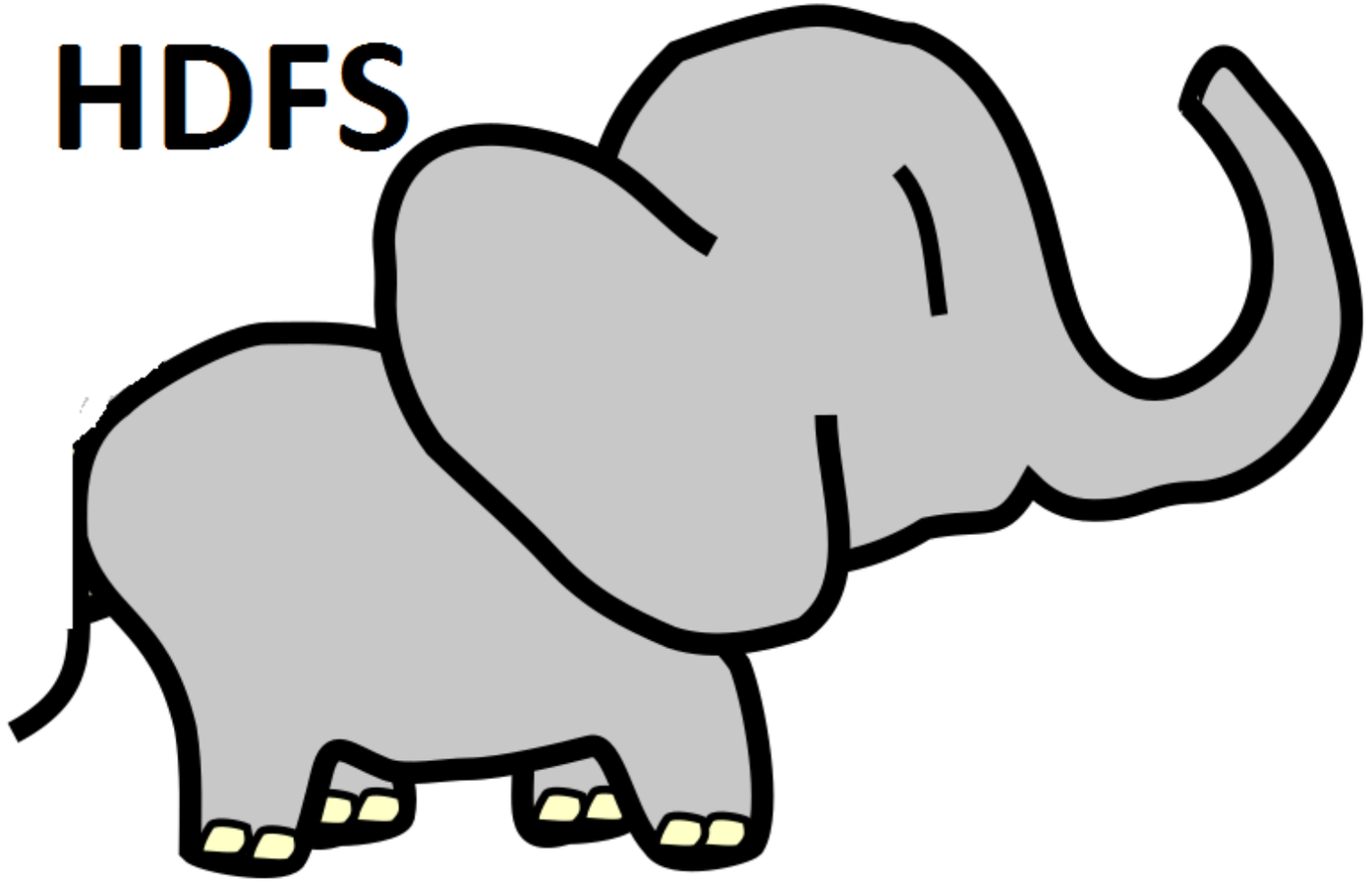
*** SPCMDCON.SYS - Address FBFE7617 base at FBFE5000, DateStamp 3d6dd67c

HDFS Limitations

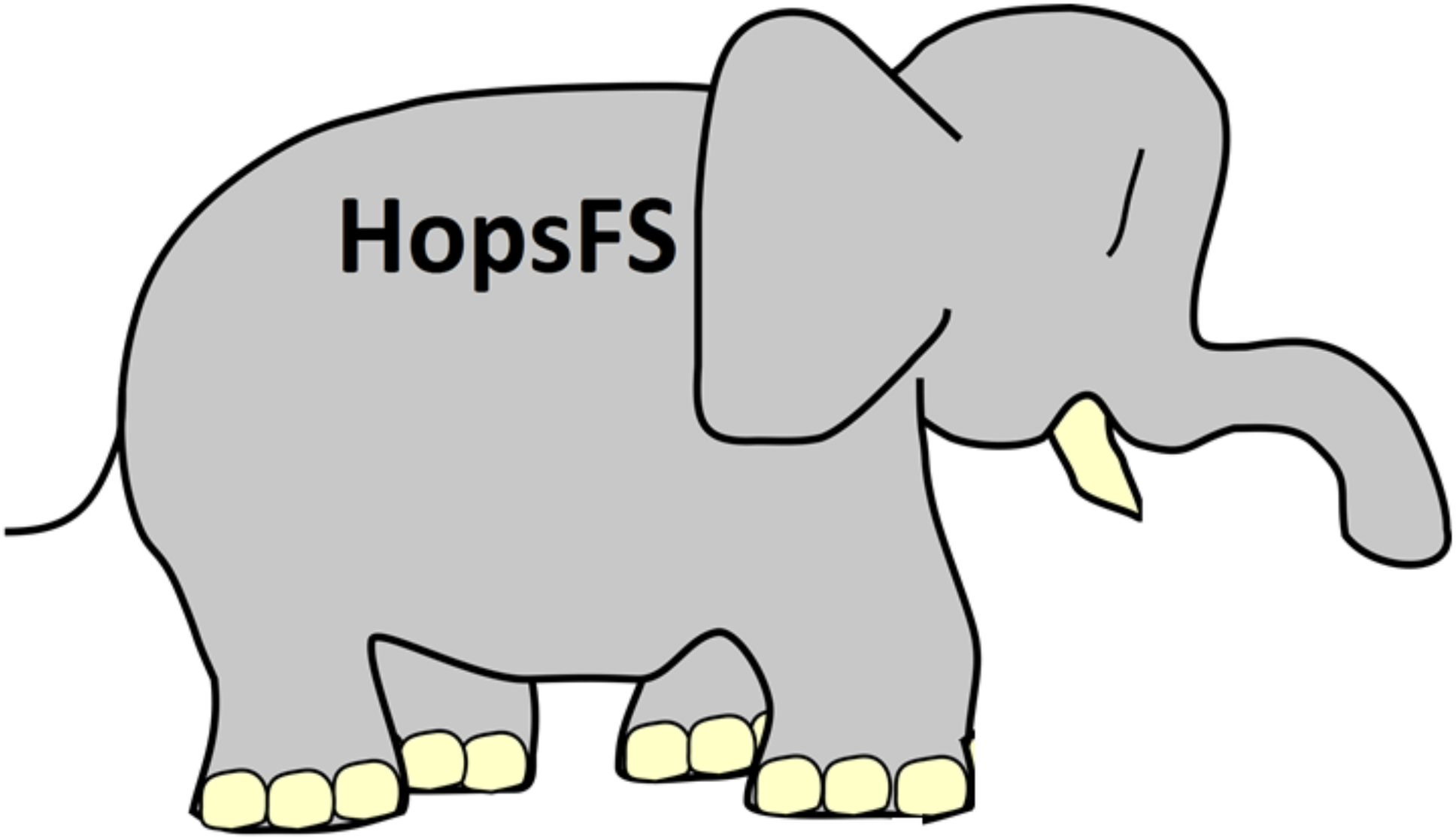
- Namespace size upper bound: ~ 500 million files
- At most 70-80 thousands file system operations / sec

HopsFS

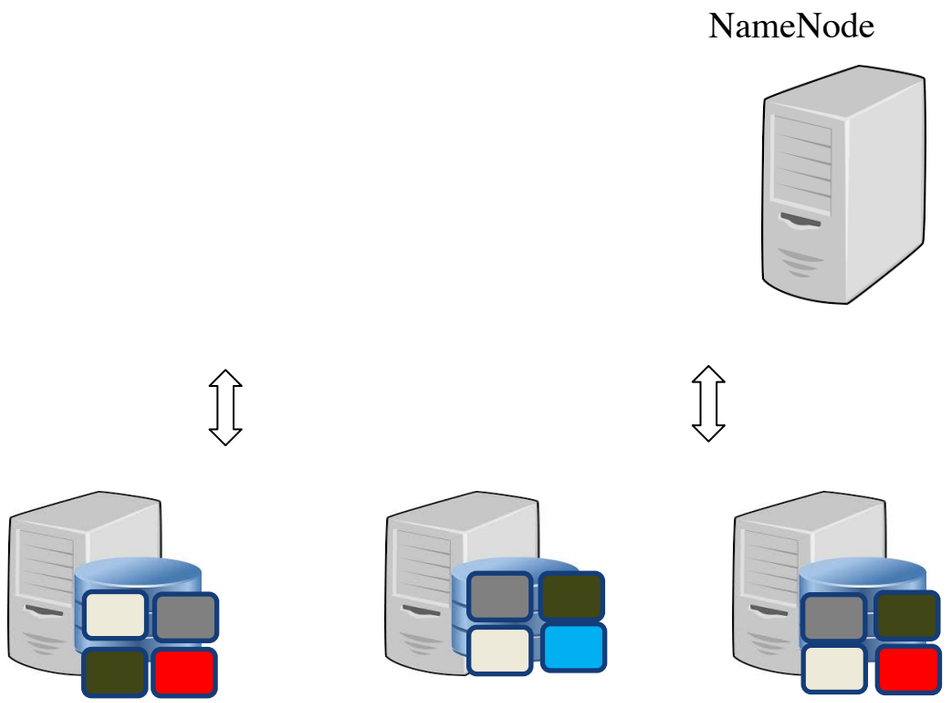
HopsFS



HopsFS

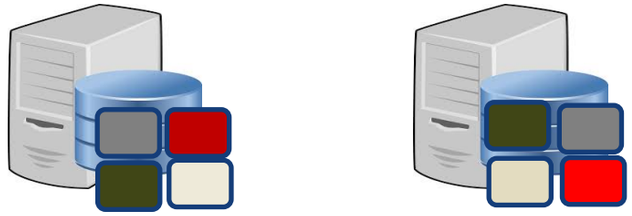


HopsFS Architecture

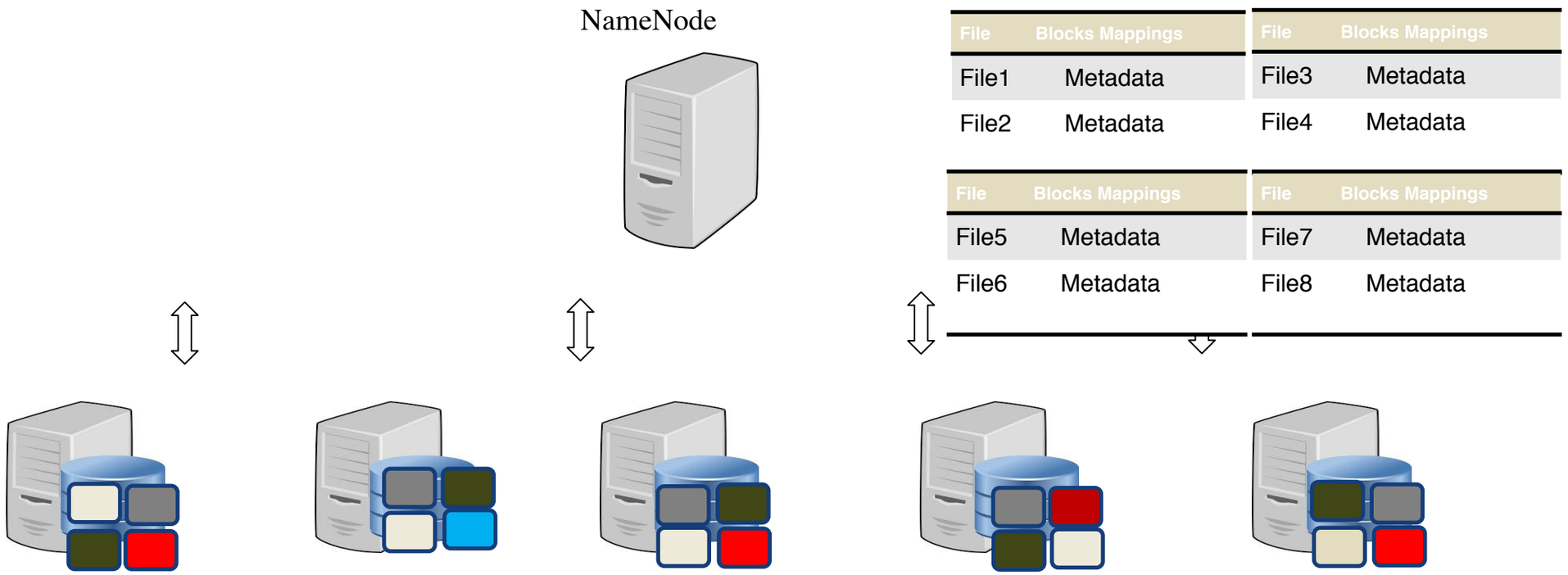


File System Metadata

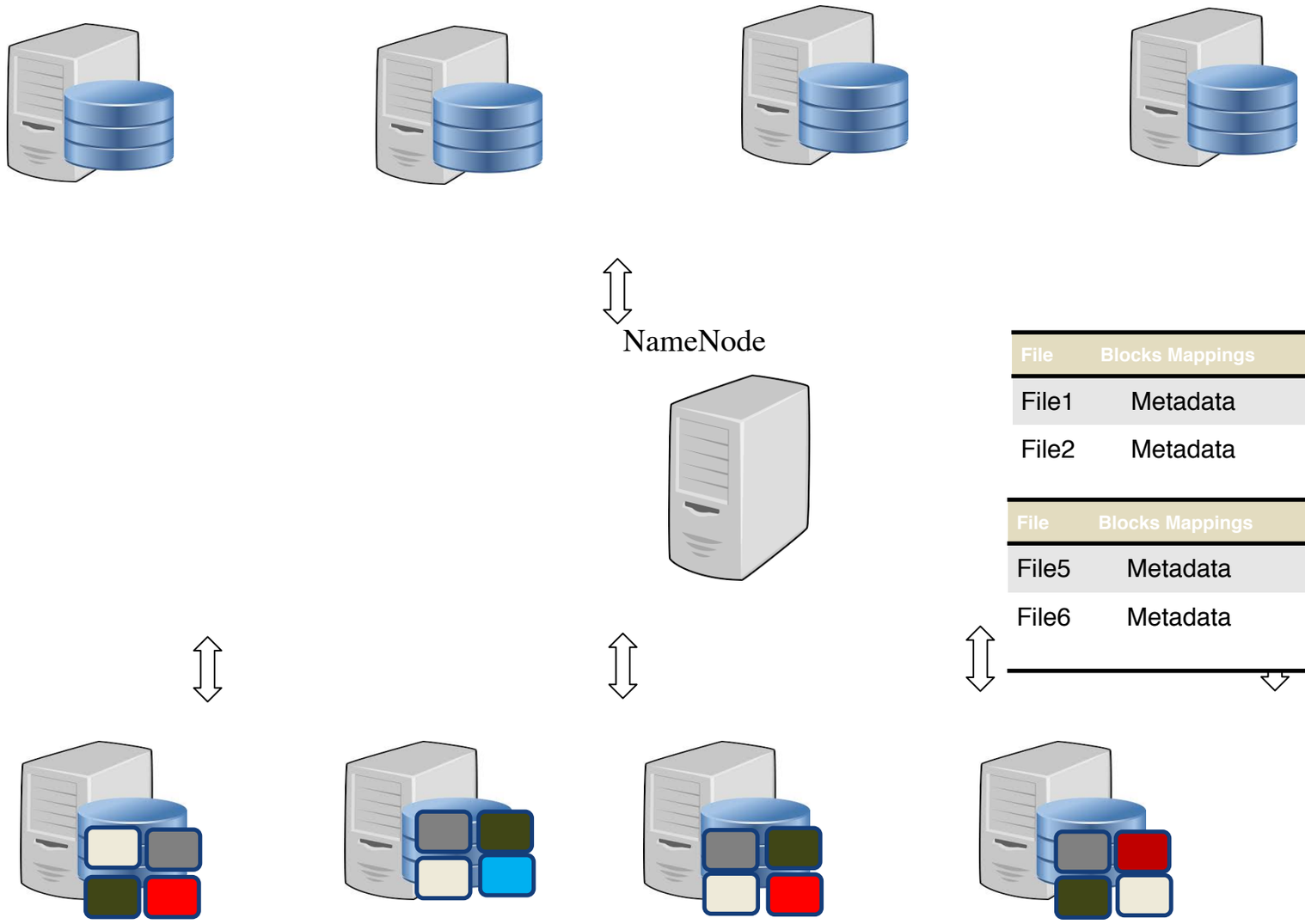
File	Blocks Mappings
File1	Blk1 → DN1, Blk2 → DN4, Blk3 → DN5
File2	Blk1 → DN1, Blk2 → DN4
File3	Blk1 → DN1, Blk2 → DN2, Blk3 → DN3
File4	Blk1 → DN100
File5	Blk1 → DN4, Blk2 → DN2, Blk3 → DN9
FileN	Blk1 → DN2, Blk2 → DN8



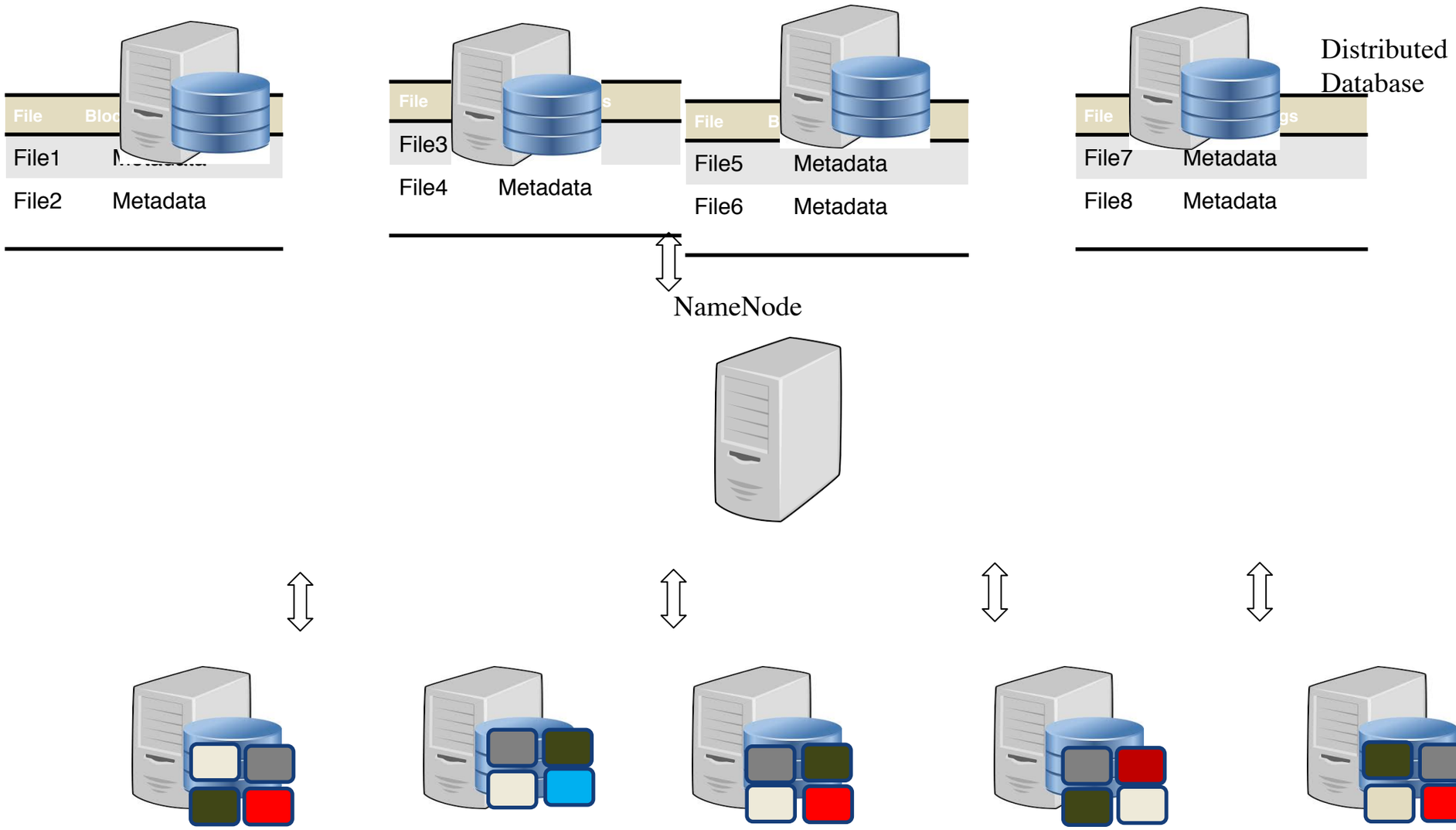
HopsFS Architecture



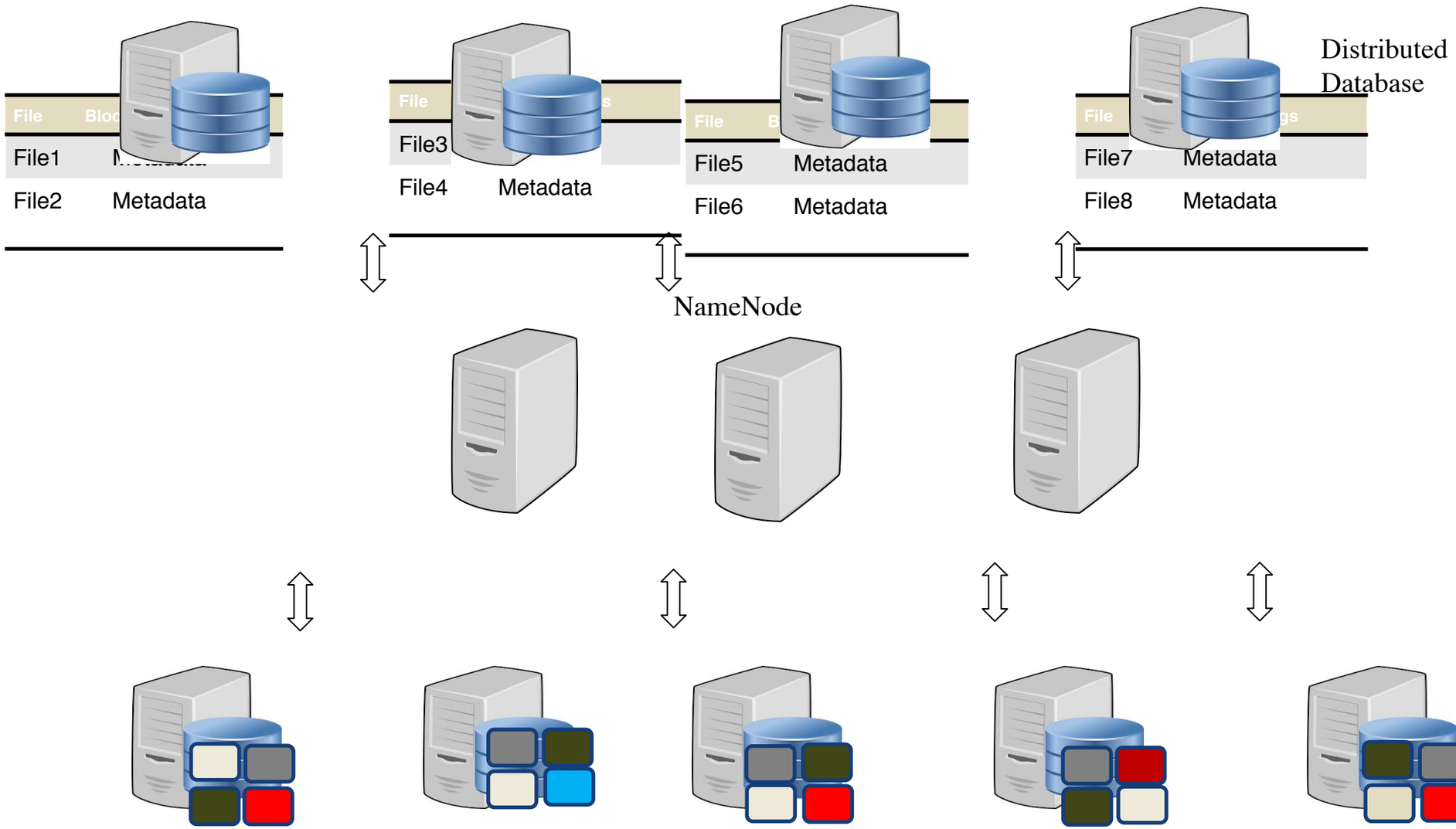
HopsFS Architecture



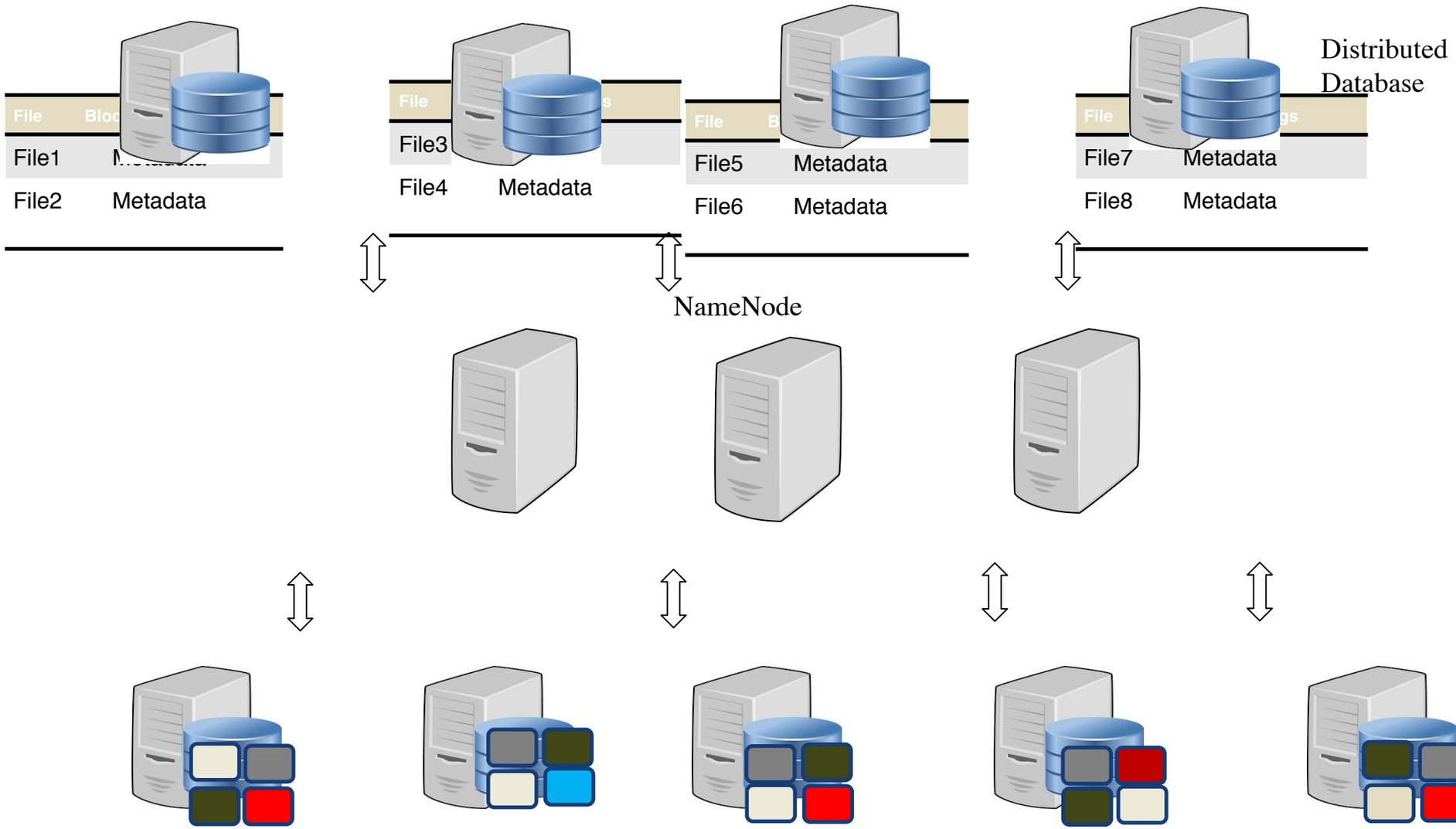
HopsFS Architecture



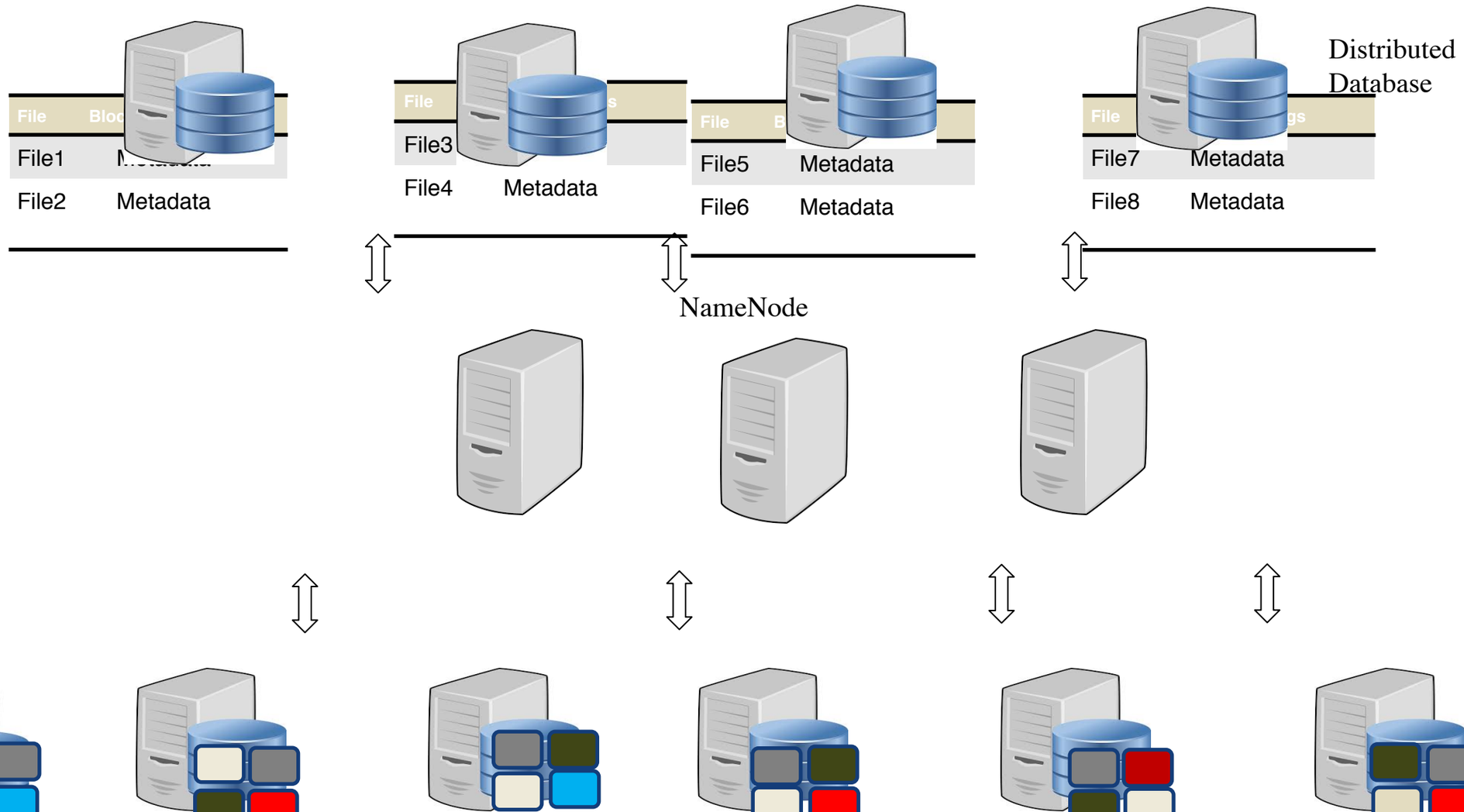
HopsFS Architecture



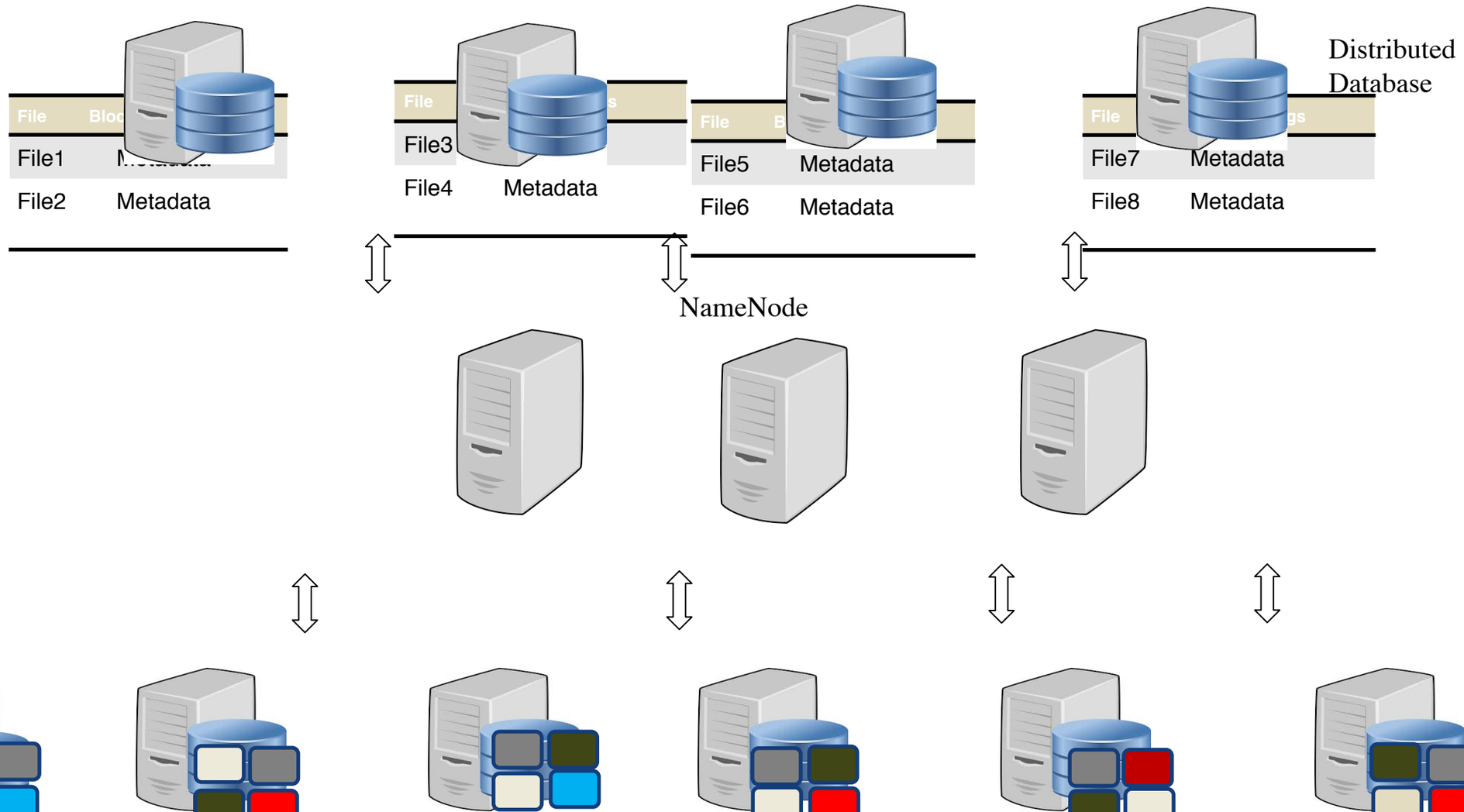
HopsFS Architecture



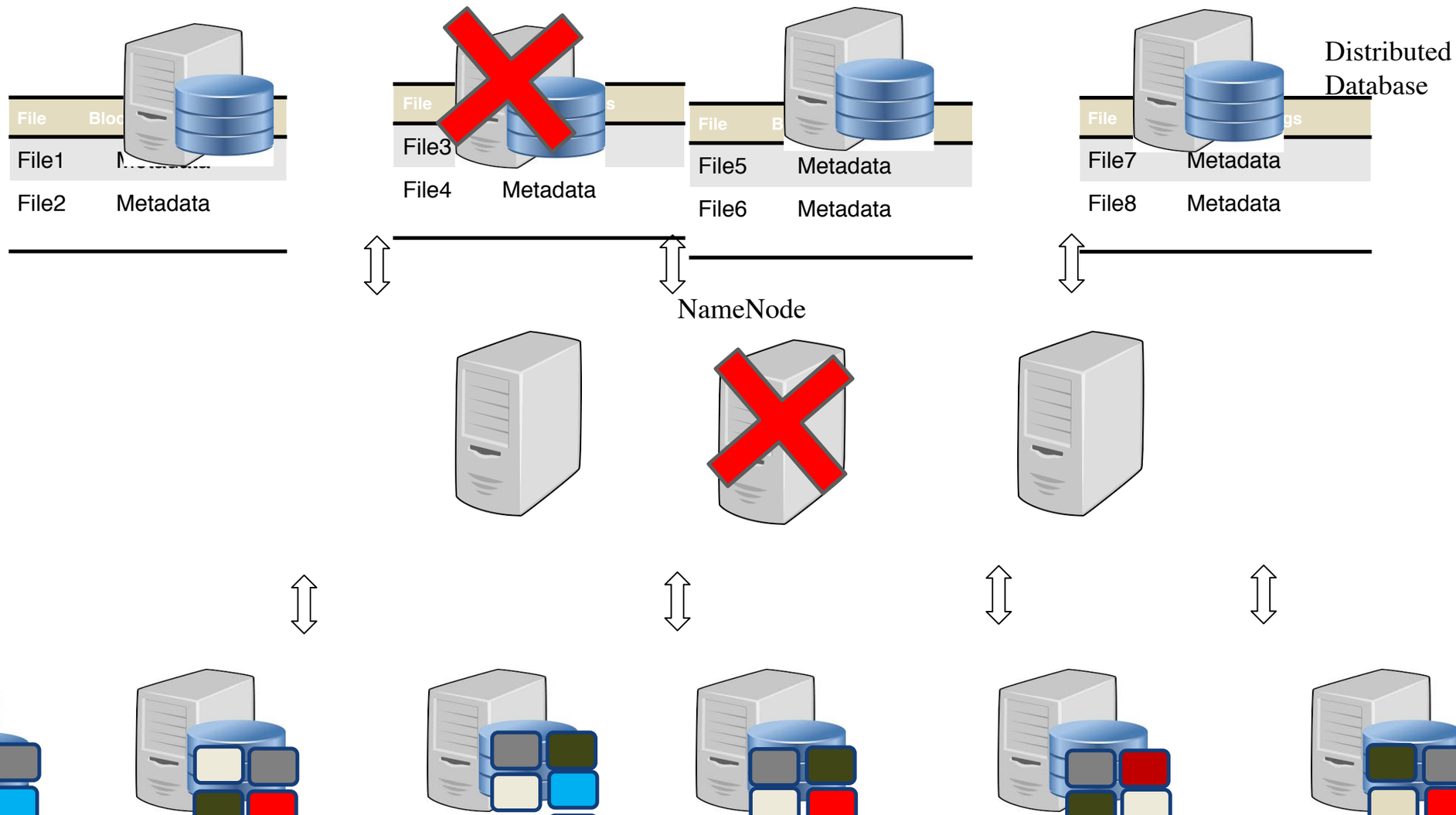
HopsFS Architecture



HopsFS Architecture

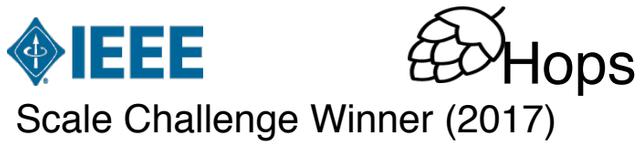


HopsFS Architecture



HopsFS Scalability

- **16X-37X** the throughput of HDFS
- **37** times more files than HDFS
- **10** times lower latency



Integration with NVMe

	Standard persistent disks	Regional persistent disks	Standard SSD persistent disks	Regional SSD persistent disks	Local SSD (SCSI)	Local SSD (NVMe)
Maximum sustained IOPS						
Read IOPS per GB	0.75	0.75	30	30	266.7	453.3
Write IOPS per GB	1.5	1.5	30	30	186.7	240
Read IOPS per instance	3,000	3,000	15,000 - 60,000*	15,000 - 60,000*	400,000	680,000
Write IOPS per instance	15,000	15,000	15,000 - 30,000*	15,000 - 30,000*	280,000	360,000

https://cloud.google.com/compute/docs/disks/performance

Integration with NVMe

	Standard persistent disks	Regional persistent disks	Standard SSD persistent disks	Regional SSD persistent disks	Local SSD (SCSI)	Local SSD (NVMe)
Maximum sustained IOPS						
Read IOPS per GB	0.75	0.75	30	30	266.7	453.3
Write IOPS per GB	1.5	1.5	30	30	186.7	240
Read IOPS per instance	3,000	3,000	15,000 - 60,000*	15,000 - 60,000*	400,000	680,000
Write IOPS per instance	15,000	15,000	15,000 - 30,000*	15,000 - 30,000*	280,000	360,000

https://cloud.google.com/compute/docs/disks/performance

Integration with NVMe

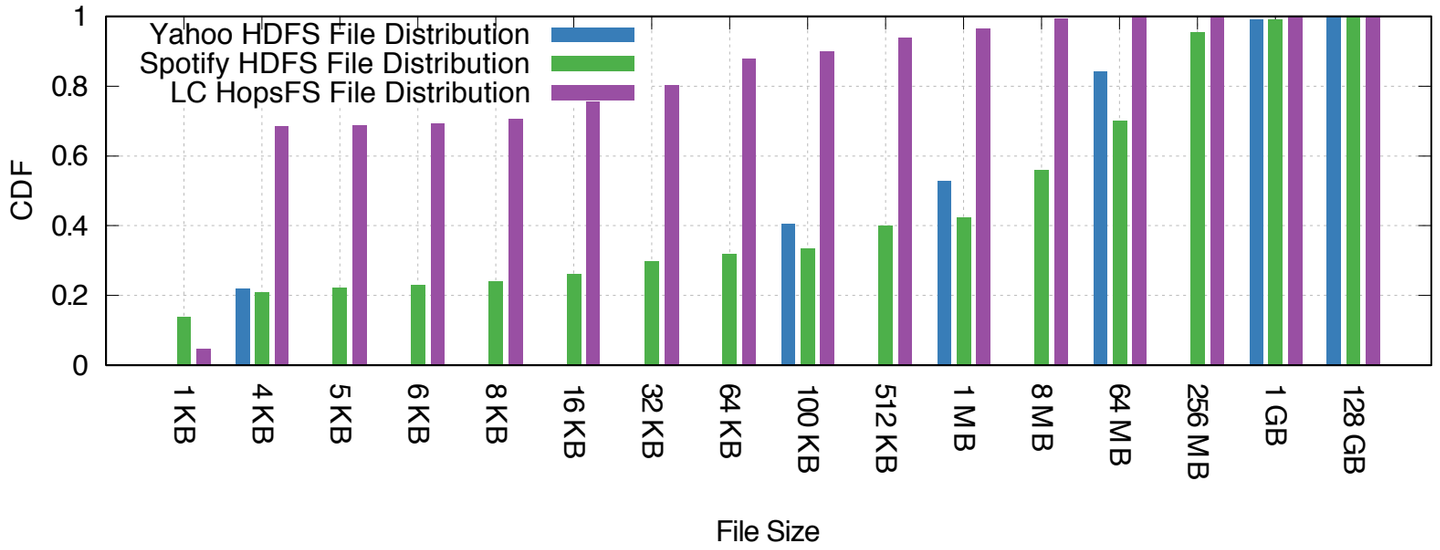
	Standard persistent disks	Regional persistent disks	Standard SSD persistent disks	Regional SSD persistent disks	Local SSD (SCSI)	Local SSD (NVMe)
Maximum sustained IOPS						
Read IOPS per GB	0.75	0.75	30	30	266.7	453.3
Write IOPS per GB	1.5	1.5	30	30	186.7	240
Read IOPS per instance	3,000	3,000	15,000 - 60,000*	15,000 - 60,000*	400,000	680,000
Write IOPS per instance	15,000	15,000	15,000 - 30,000*	15,000 - 30,000*	280,000	360,000

https://cloud.google.com/compute/docs/disks/performance

HDFS (and S3) are designed around large blocks (optimized to overcome slow random I/O on disks), while new NVMe hardware supports fast random disk I/O (and potentially small blocks sizes)

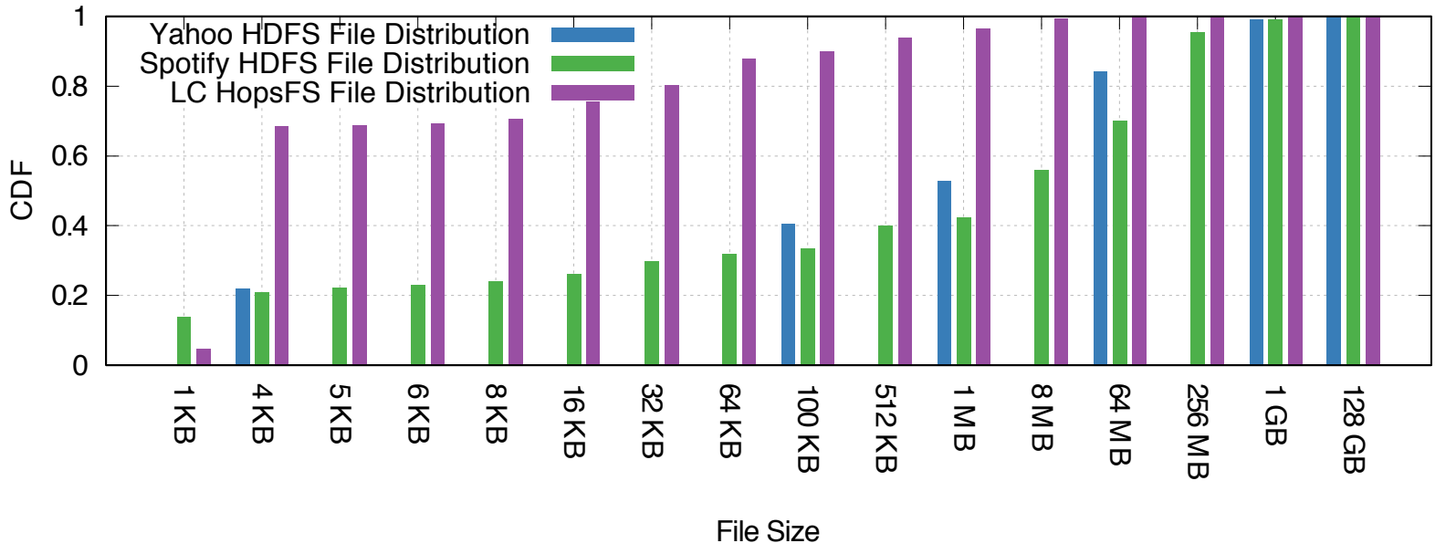
Small files

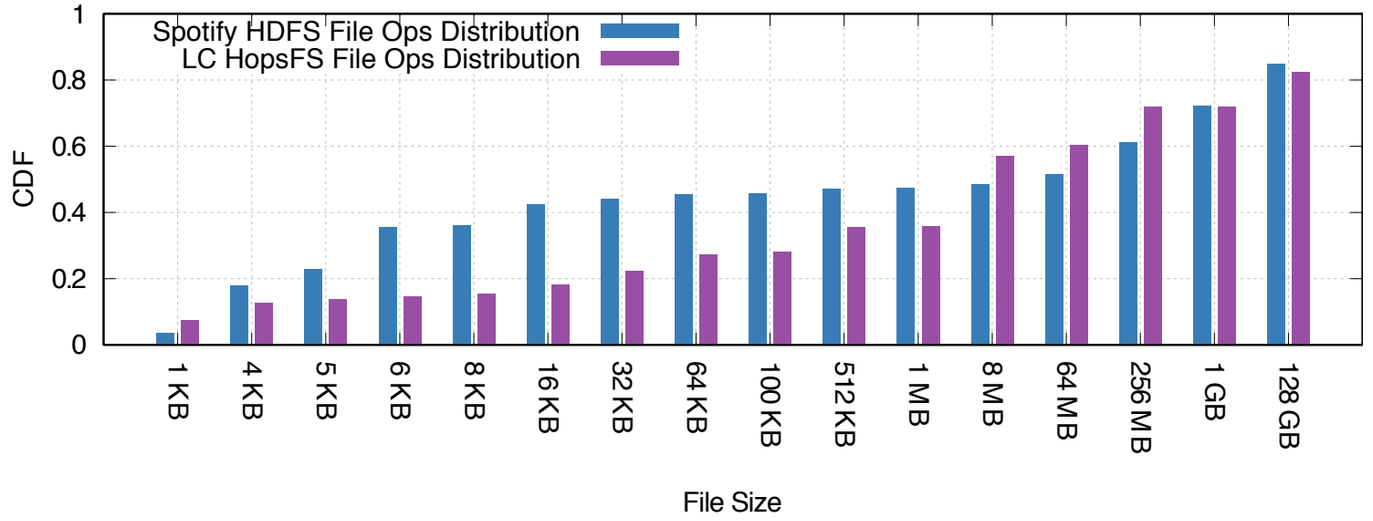
Small files



At Yahoo! and Spotify ≈**20%** of the files are less than 4 KB.
Logical Clocks' HopsFS cluster ≈**68%** of the files are less than 4 KB

Small files

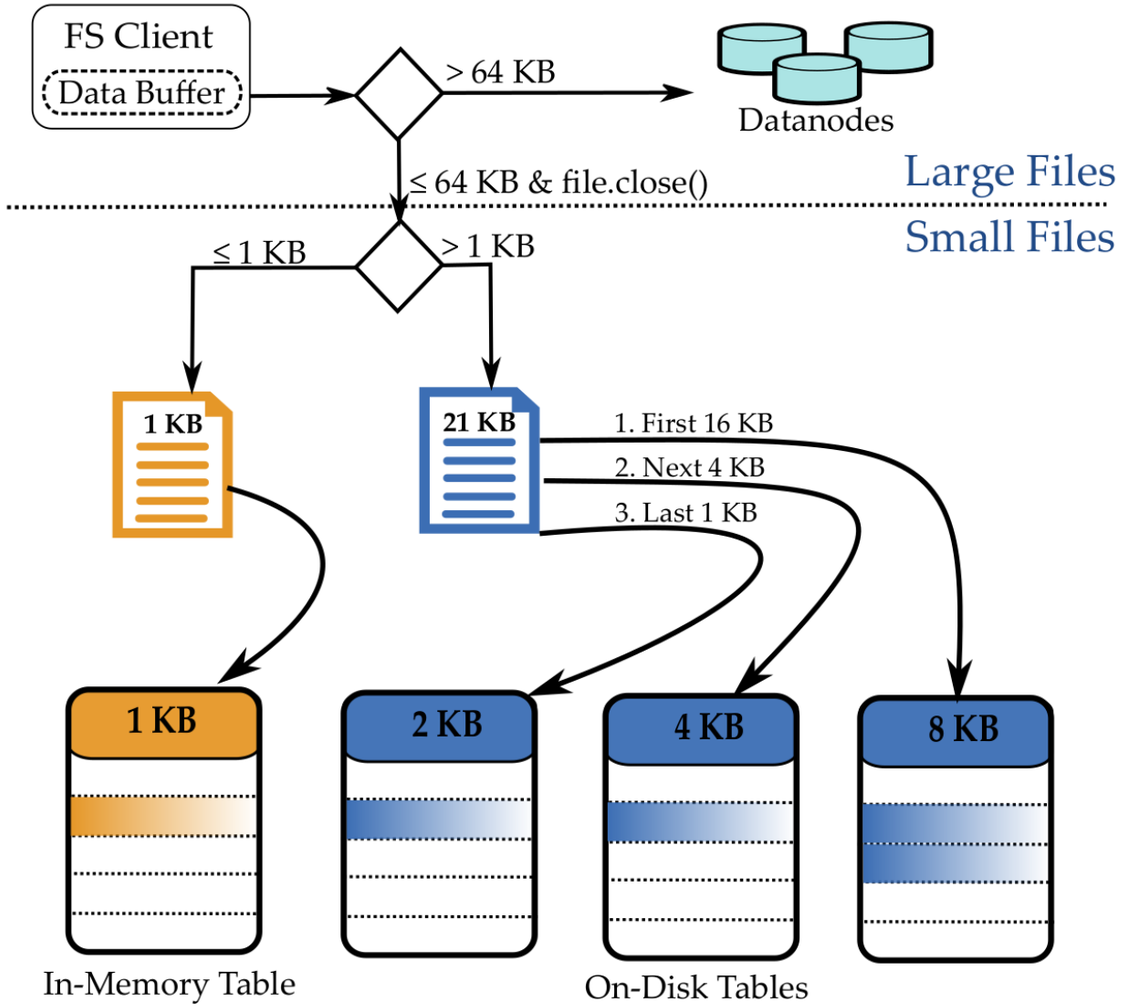




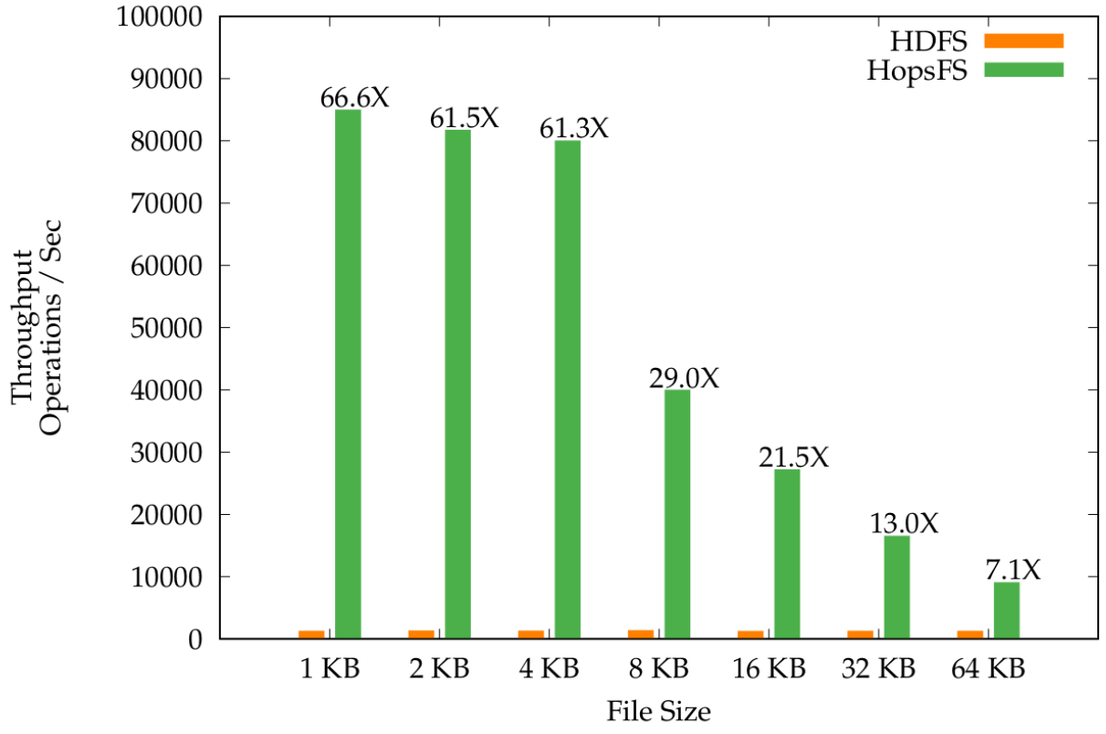
At Yahoo! and Spotify ≈**20%** of the files are less than 4 KB.
Logical Clocks' HopsFS cluster ≈**68%** of the files are less than 4 KB

At Spotify, and Logical Clocks ≈ **42**% and ≈**18**% of all the file system operations are performed on files less than 16 KB files

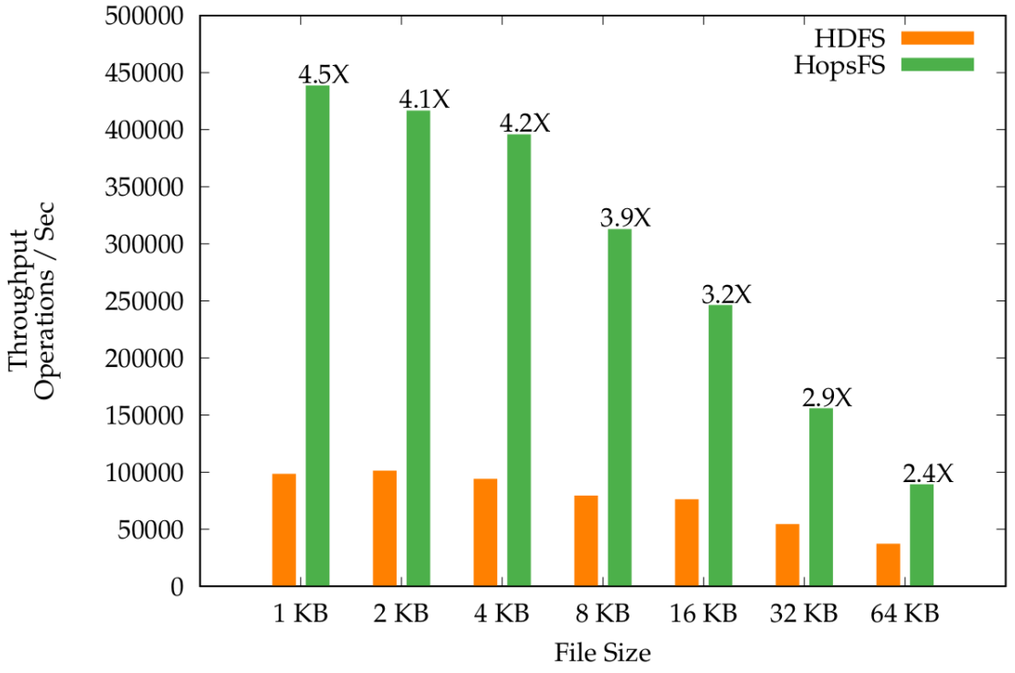
Size Matters



Small Files performance in HopsFS

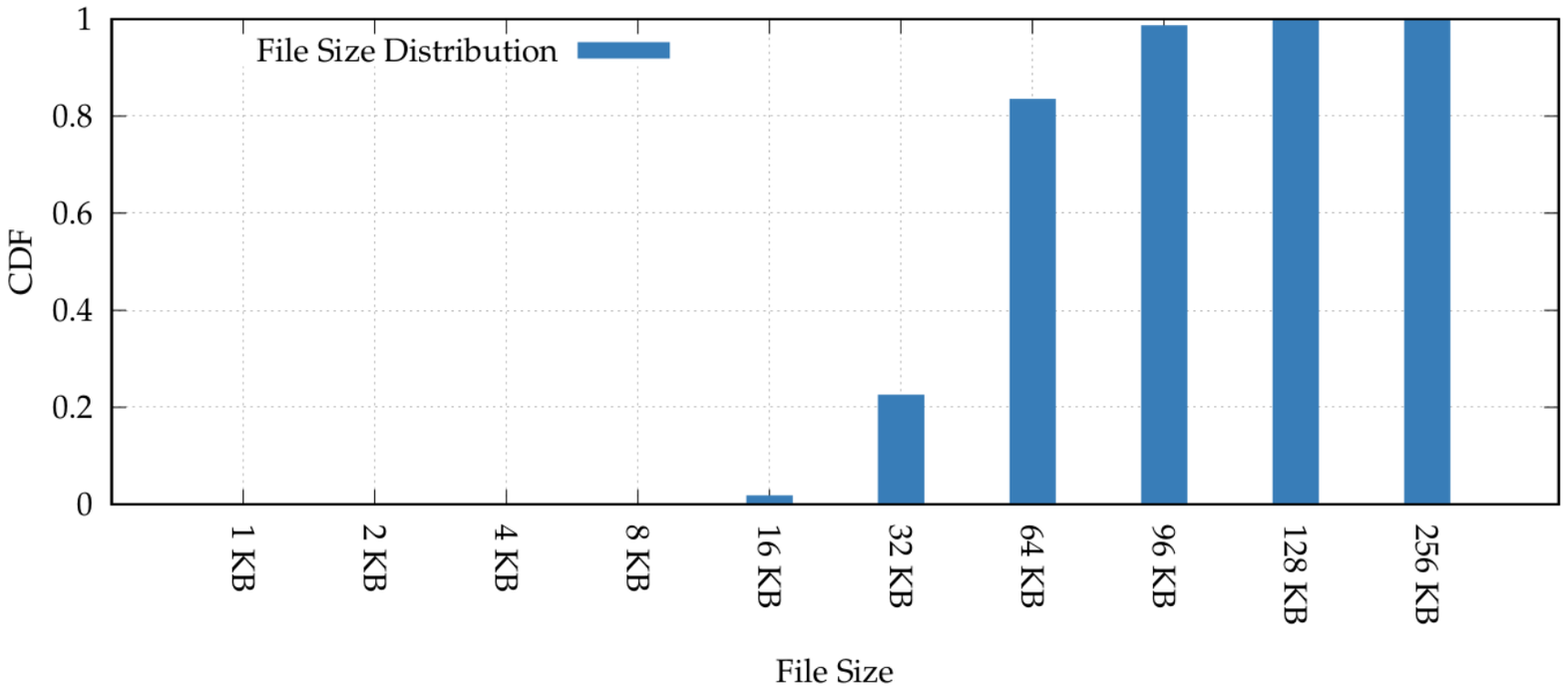


a. File Write Performance



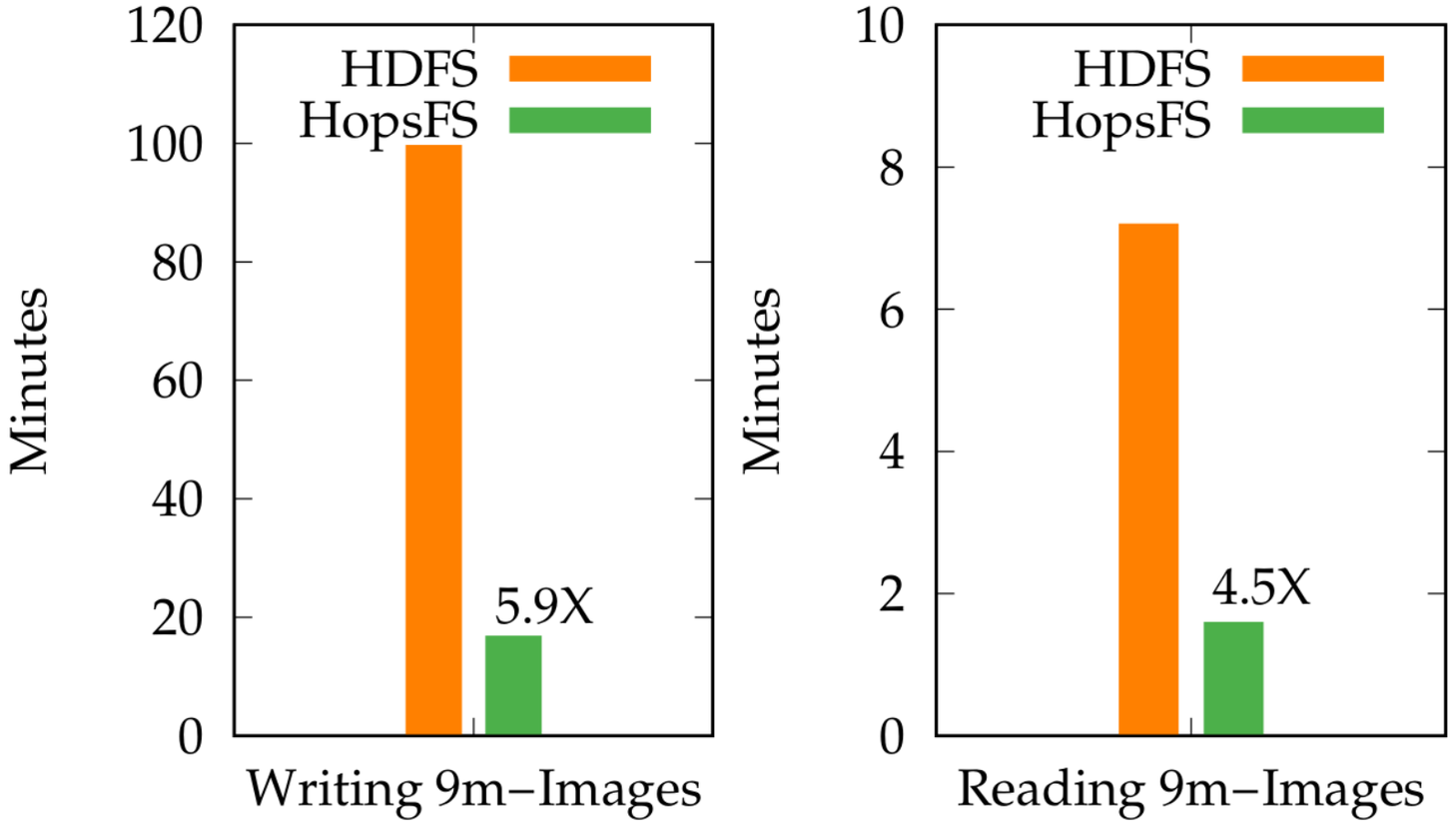
b. File Read Performance

Open Images dataset



83.5% of the files in the dataset are ⩽ **64** KB.

Open Images Dataset



Requirements

- ~~Reading/Writing millions of images with high throughput~~

- Attaching annotations to each image, and then searching using these annotations

Attaching Extended Metadata

Attaching Extended Metadata

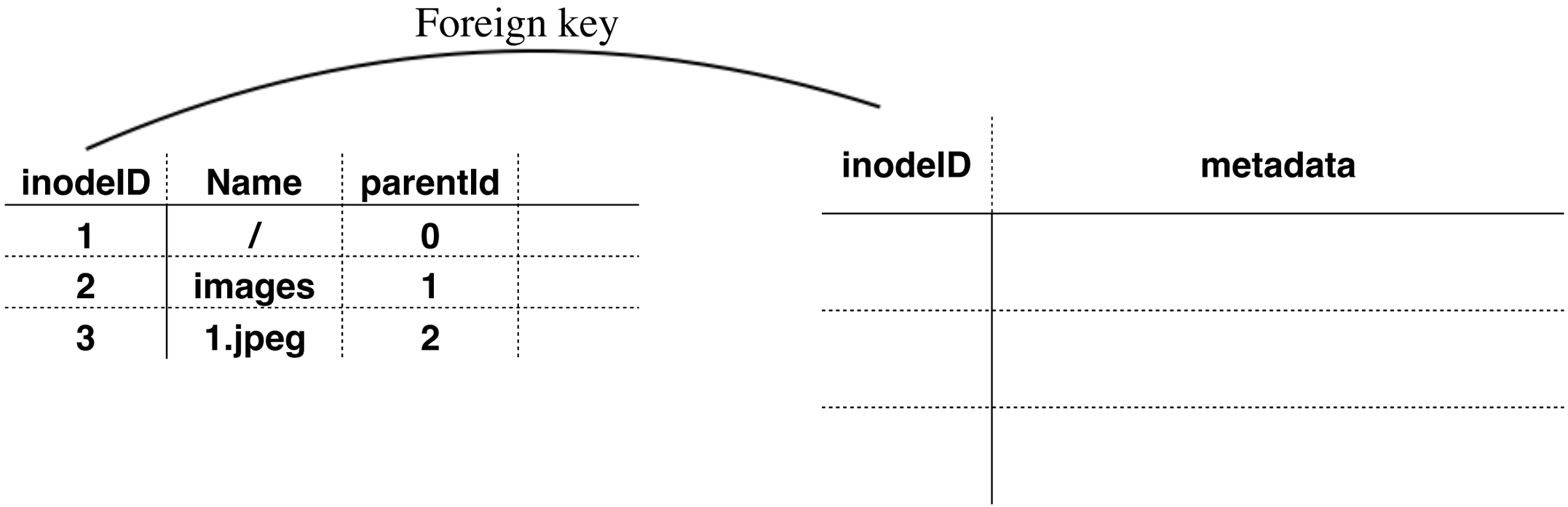
inodeID	Name	parentId	
1	/	0	
2	images	1	
3	1.jpeg	2	

Attaching Extended Metadata

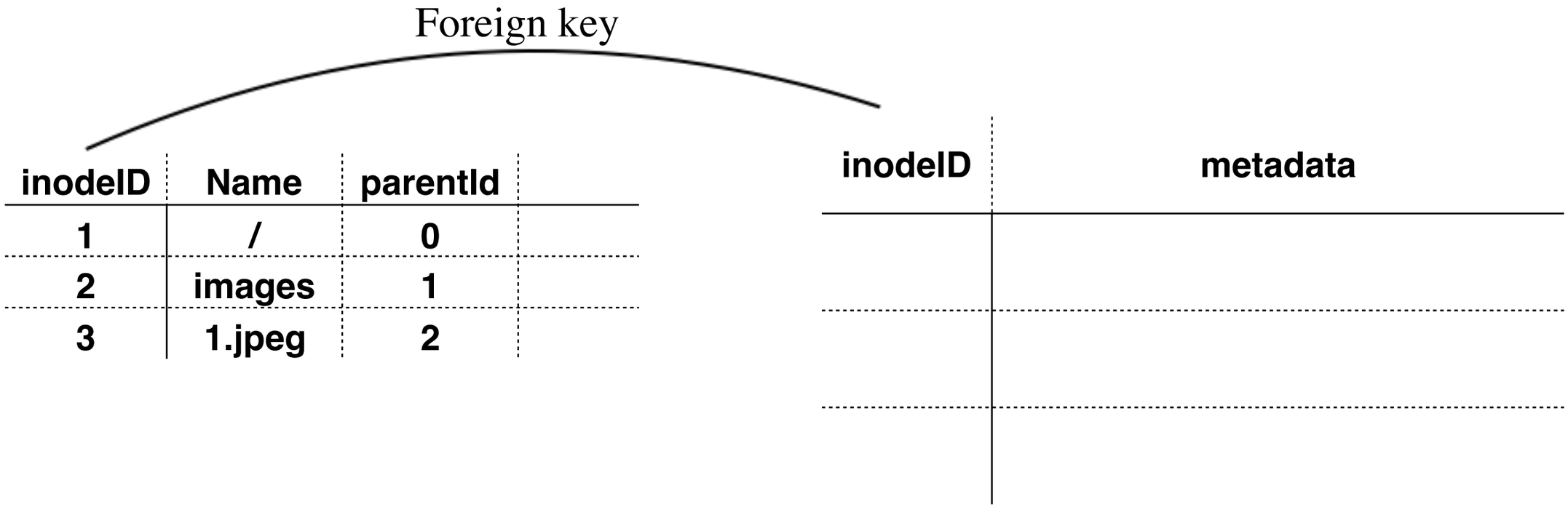
inodeID	Name	parentId	
1	/	0	
2	images	1	
3	1.jpeg	2	

inodeID	metadata

Attaching Extended Metadata

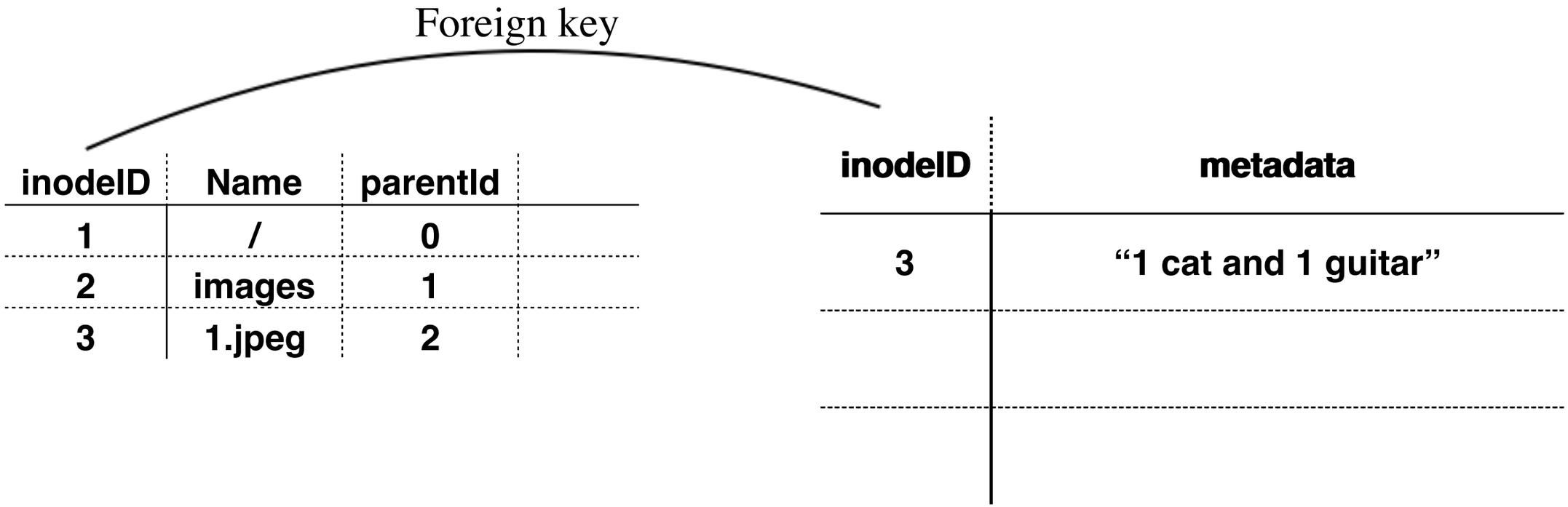


Attaching Extended Metadata



attach /images/1.jpeg ’1 cat and 1 guitar’

Attaching Extended Metadata



attach /images/1.jpeg ’1 cat and 1 guitar’

Attaching Extended Metadata

Foreign key

inodeID	Name	parentId	
1	/	0	
2	images	1	
3	1.jpeg	2	

inodeID	metadata
3	“1 cat and 1 guitar”
	Free text search?

attach /images/1.jpeg ’1 cat and 1 guitar’

HopsFS | ElasticSearch

HopsFS | ElasticSearch

HopsFS | ElasticSearch

HopsFS

ElasticSearch

HopsFS | ElasticSearch

HopsFS | ElasticSearch

1.jpeg

HopsFS I ElasticSearch

1.jpeg

1 Dog

HopsFS | ElasticSearch

1.jpeg

1 Dog

HopsFS

dog	[1.jpeg,……]

ElasticSearch

HopsFS | ElasticSearch

1.jpeg

1 Dog

HopsFS

Get All images that has a dog

dog	[1.jpeg,……]

ElasticSearch

HopsFS | ElasticSearch

1.jpeg

~~1 Dog~~

HopsFS

Get All images that has a dog

dog	[1.jpeg,……]

ElasticSearch

HopsFS | ElasticSearch

1.jpeg

~~1 Dog~~

1 Cat and 1 Guitar

HopsFS

Get All images that has a dog

dog	[1.jpeg,……]

ElasticSearch

HopsFS I ElasticSearch

1.jpeg

~~1 Dog~~

1 Cat and 1 Guitar

HopsFS

?

ElasticSearch

Get All images that has a dog

dog	[1.jpeg,……]

HopsFS | ElasticSearch

1.jpeg

~~1 Dog~~

1 Cat and 1 Guitar

HopsFS

?

Get All images that has a dog

dog	[1.jpeg,……]

ElasticSearch

Store X

HopsFS | ElasticSearch

1.jpeg

~~1 Dog~~

1 Cat and 1 Guitar

HopsFS

?

Get All images that has a dog

dog	[1.jpeg,……]

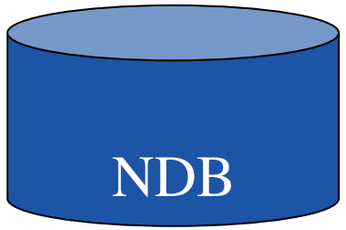
ElasticSearch

Store X

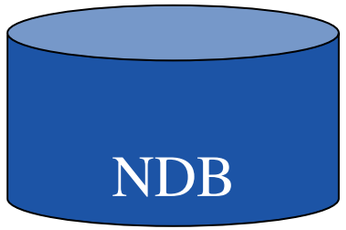
ePipe

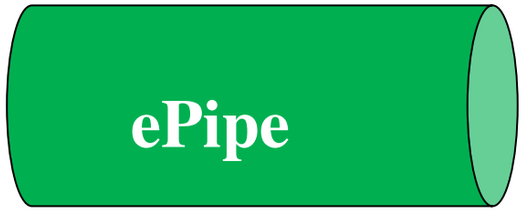
ePipe

ePipe

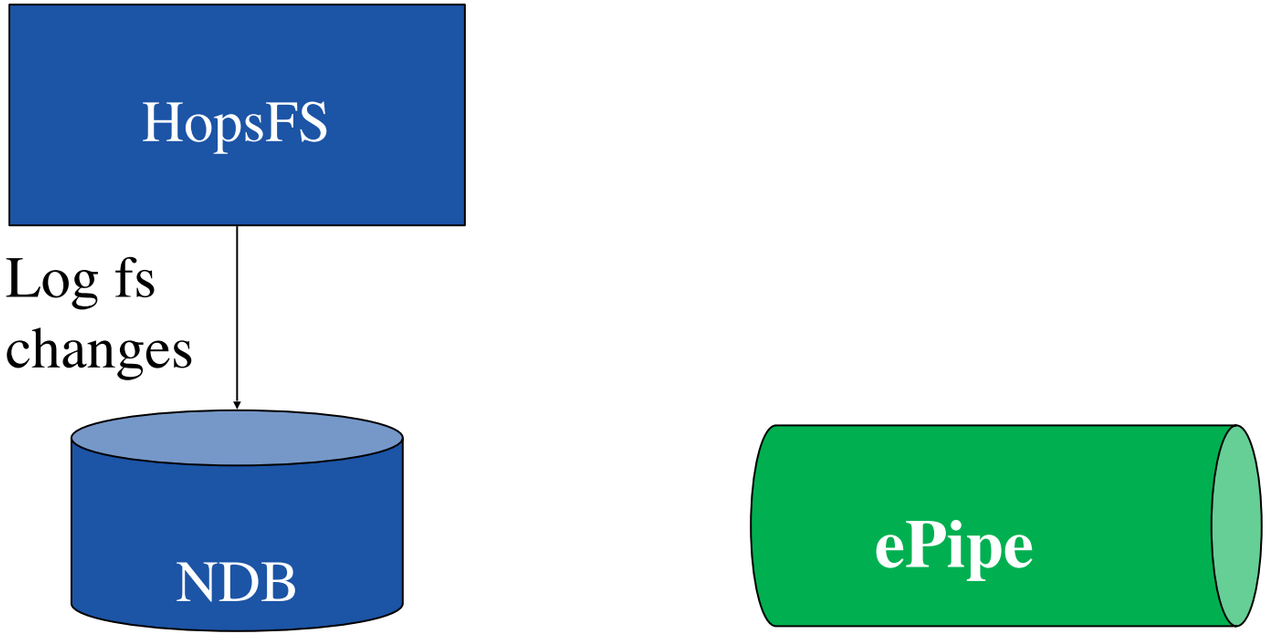


ePipe

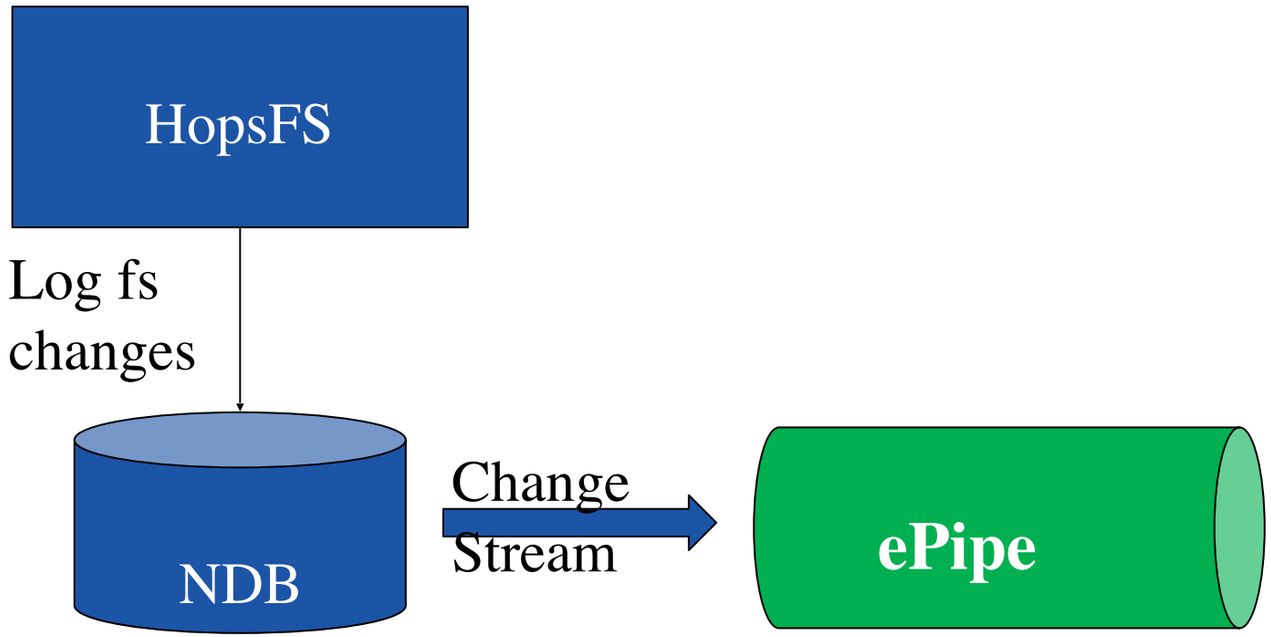




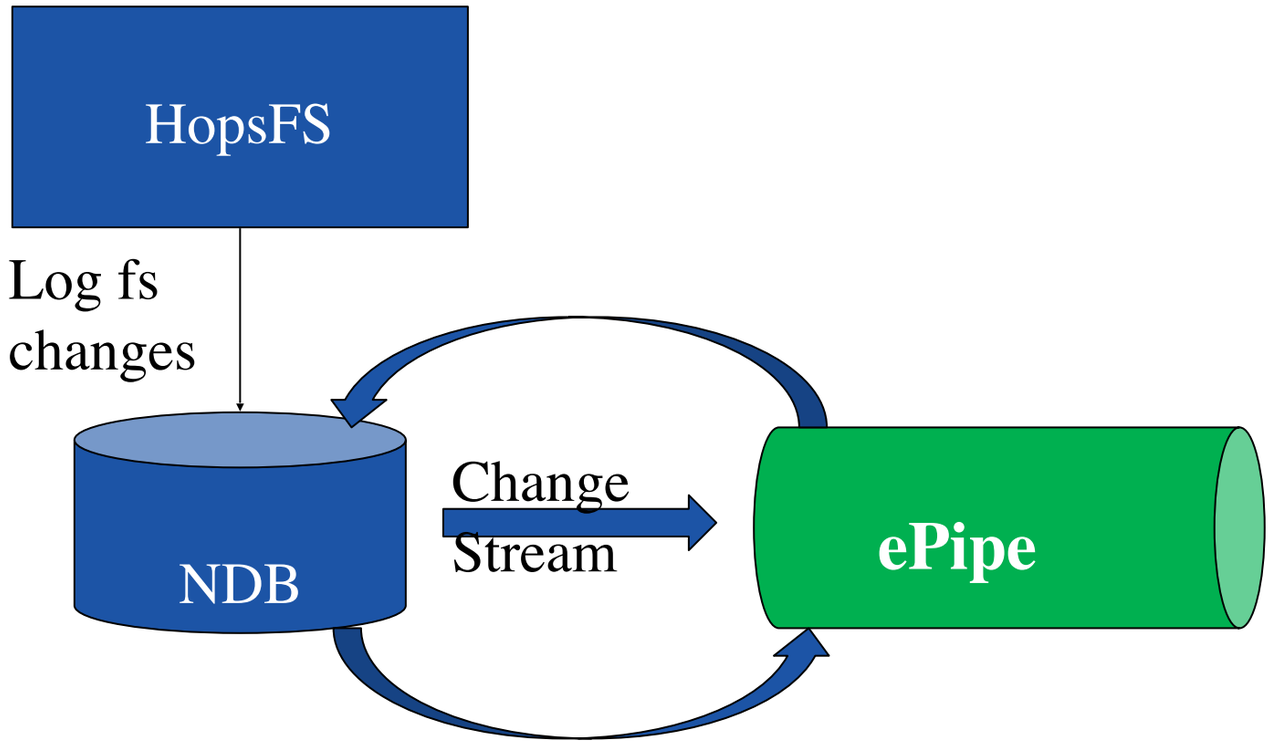
ePipe



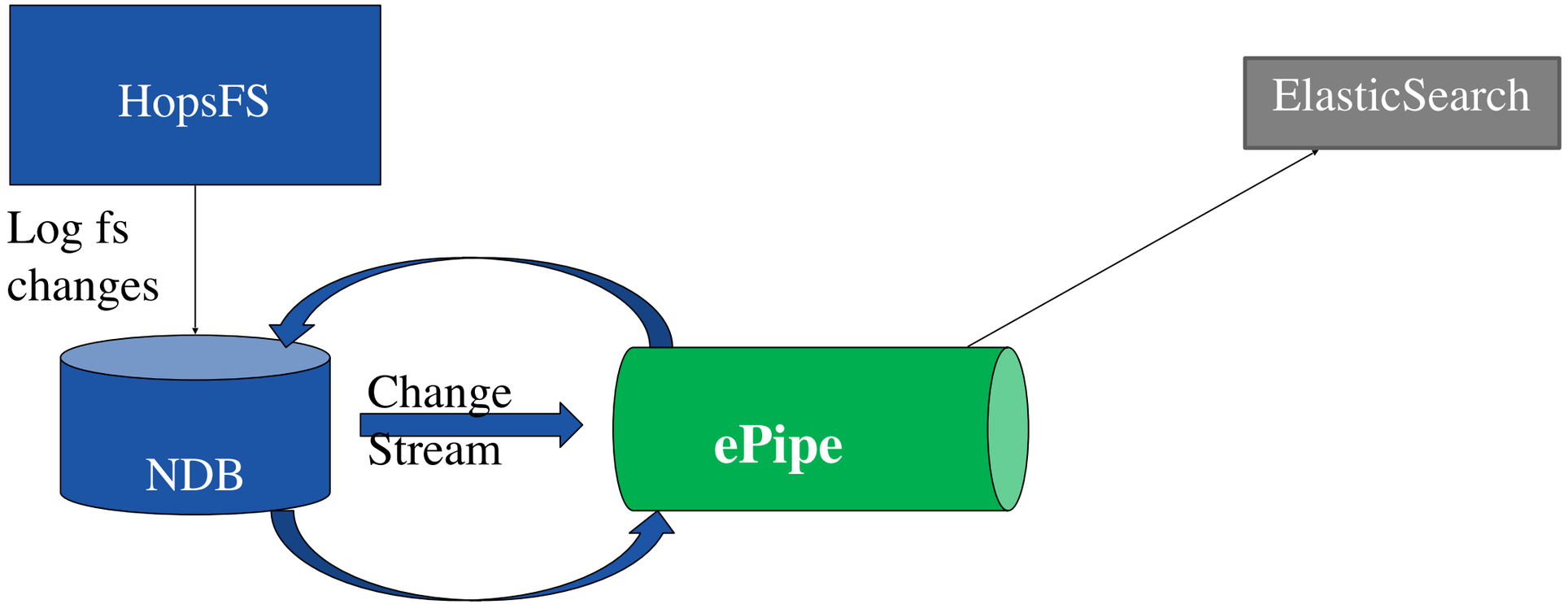
ePipe



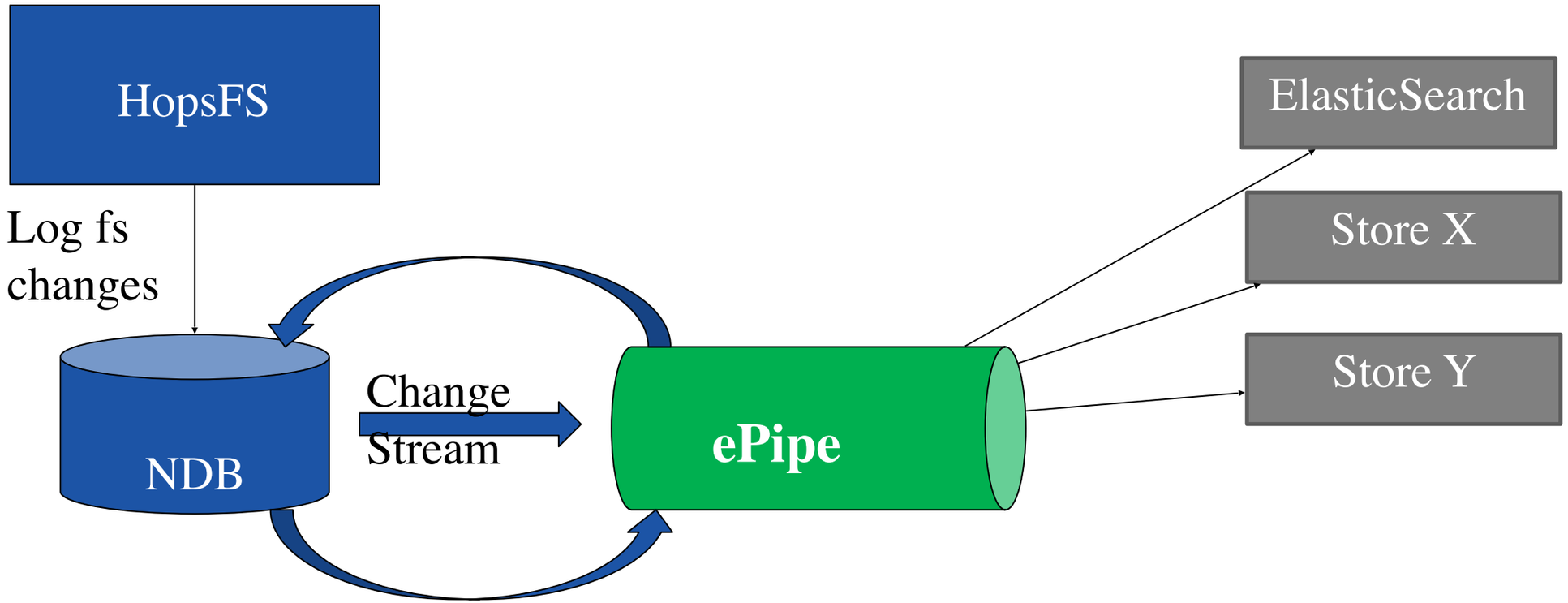
ePipe



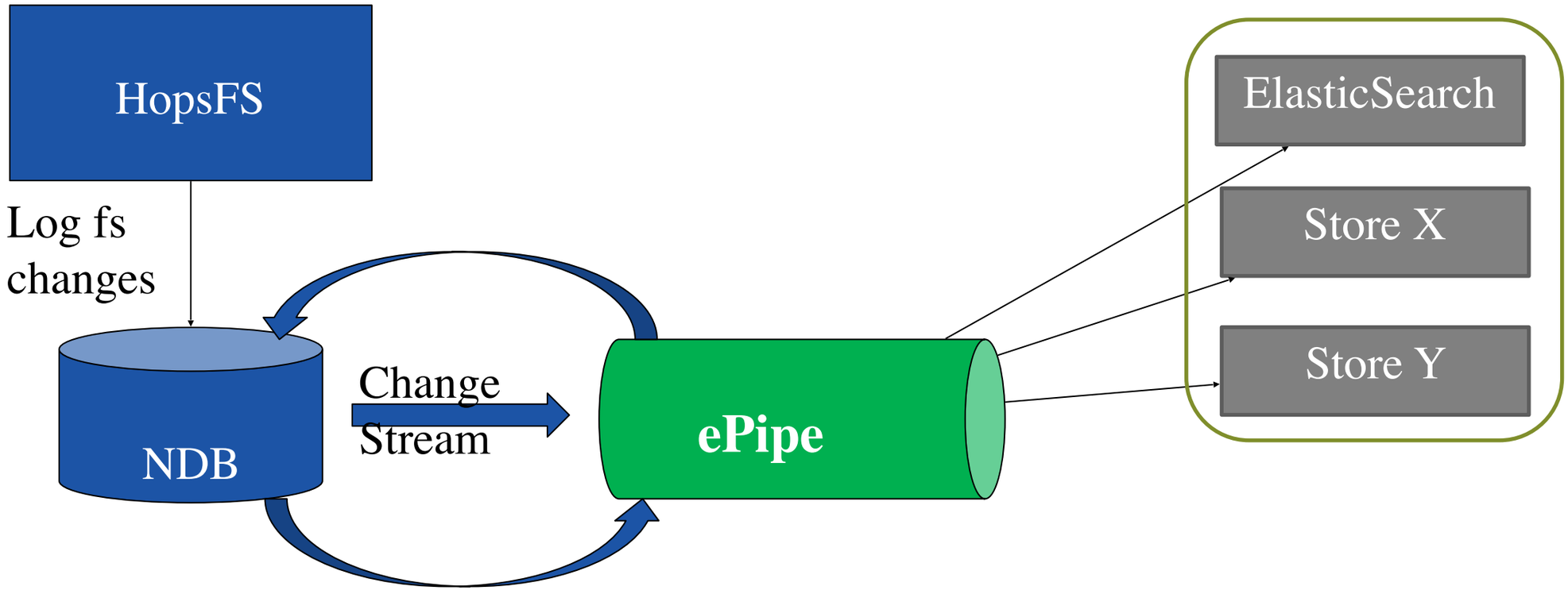
ePipe


HopsFS
Log fs changes
NDB
Change Stream
ePipe
ElasticSearch

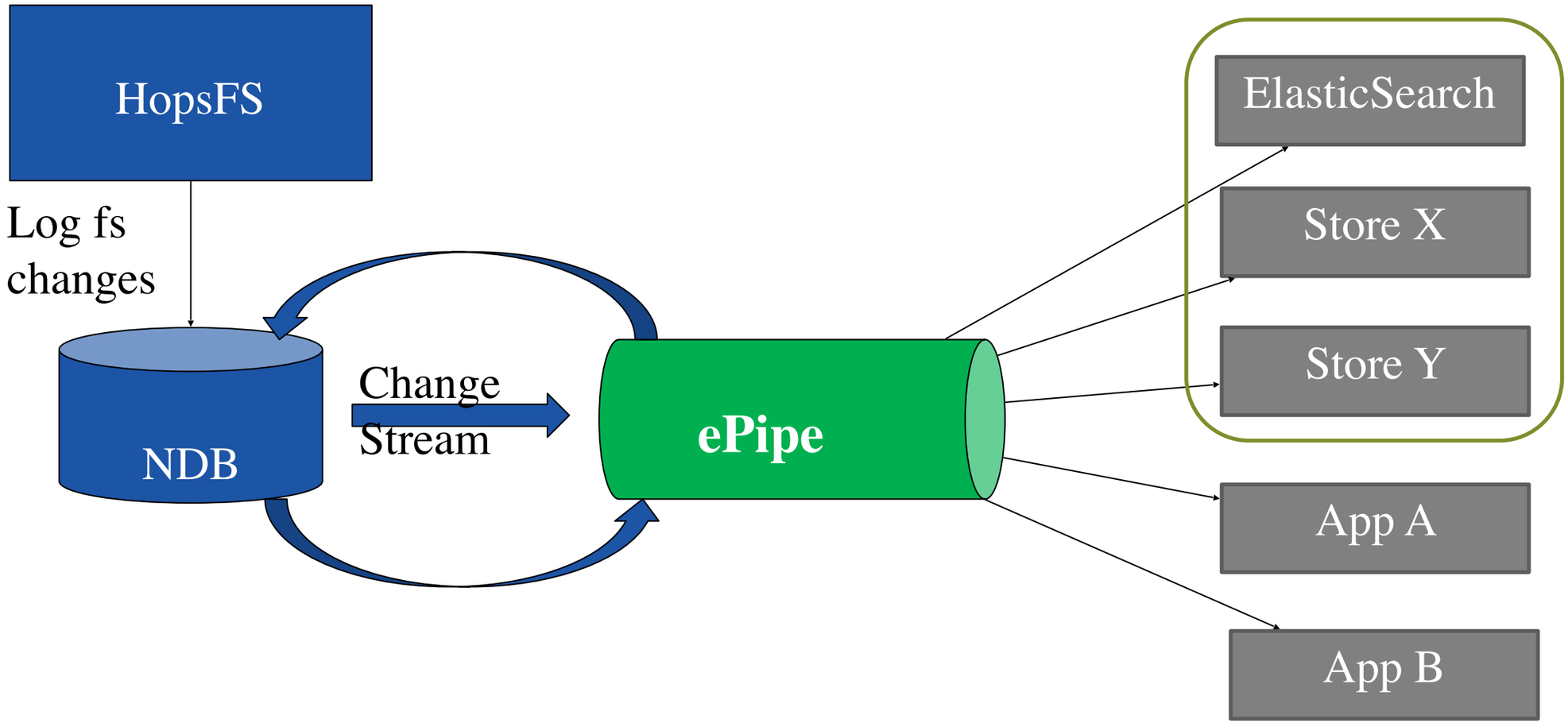
ePipe


HopsFS
Log fs changes
NDB
Change Stream
ePipe
ElasticSearch
Store X
Store Y

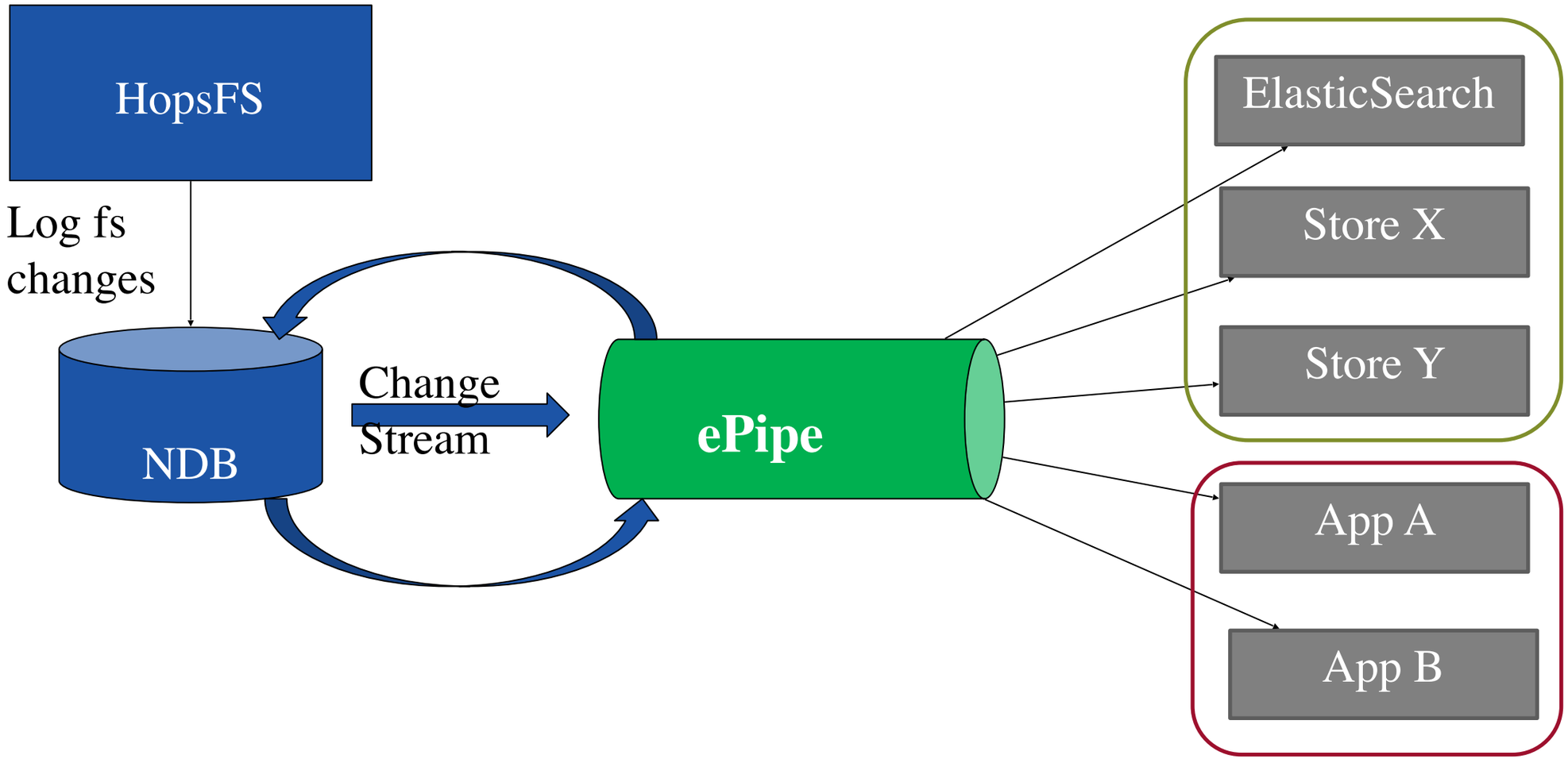
ePipe


HopsFS
Log fs changes
NDB
Change Stream
ePipe
ElasticSearch
Store X
Store Y

ePipe


HopsFS
Log fs changes
NDB
Change Stream
ePipe
ElasticSearch
Store X
Store Y
App A
App B

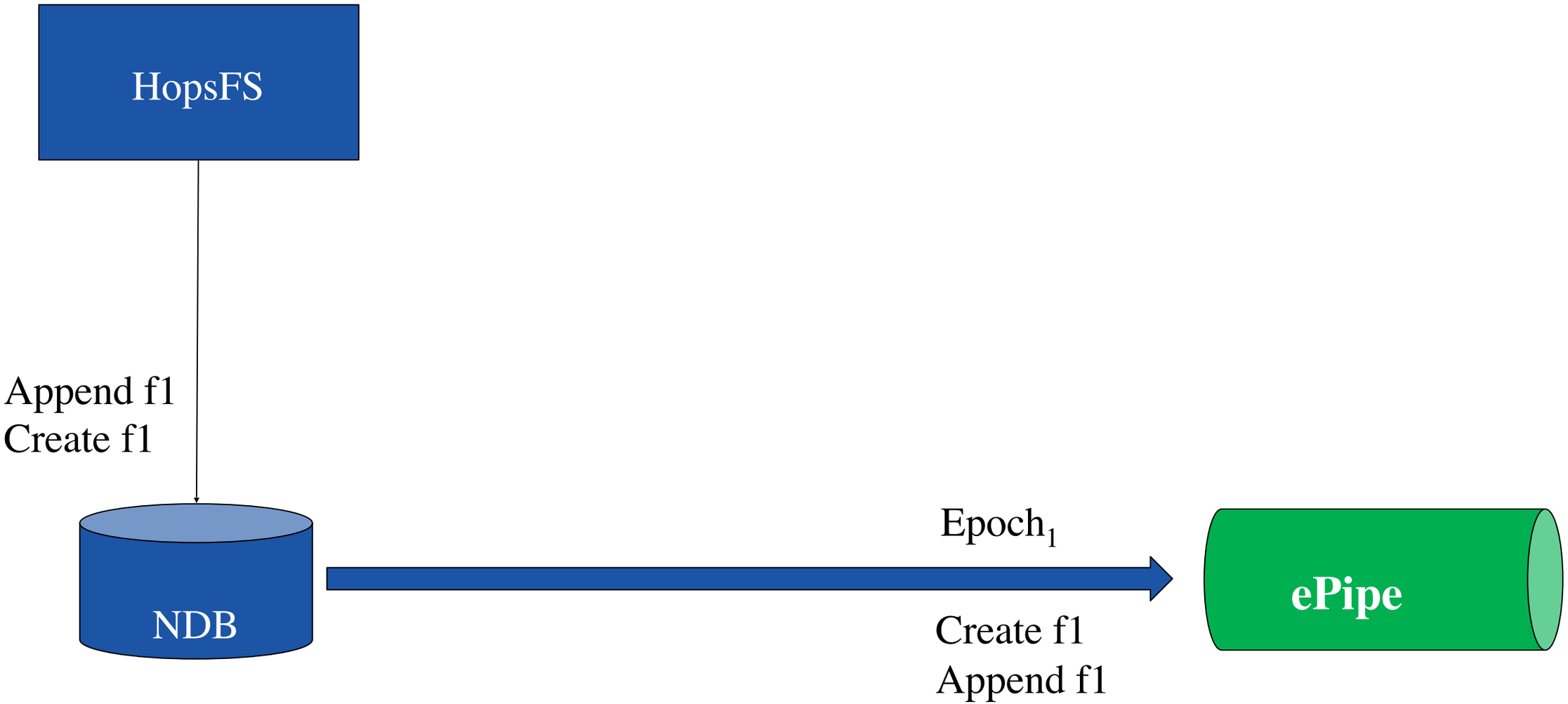
ePipe


HopsFS
Log fs changes
NDB
Change Stream
ePipe
ElasticSearch
Store X
Store Y
App A
App B

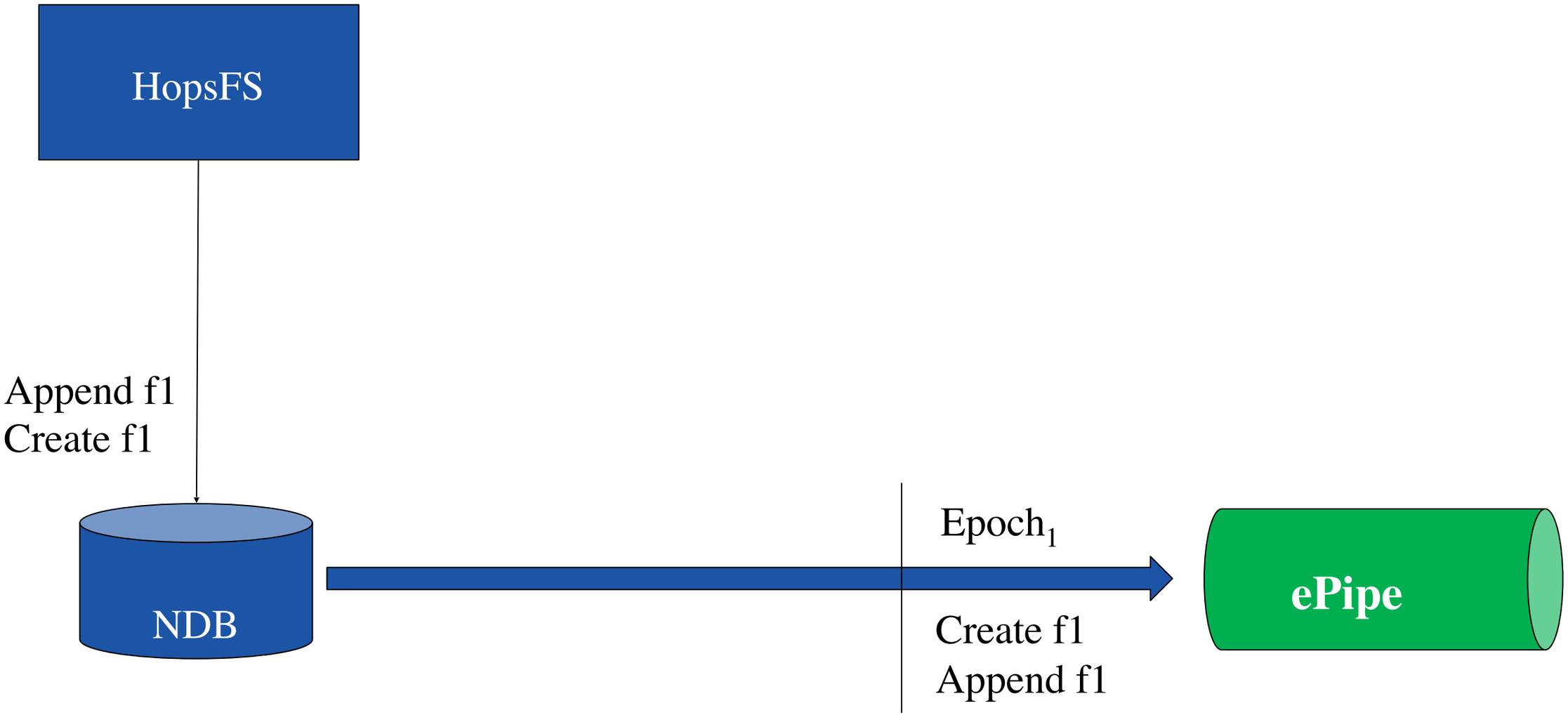
ePipe

ePipe

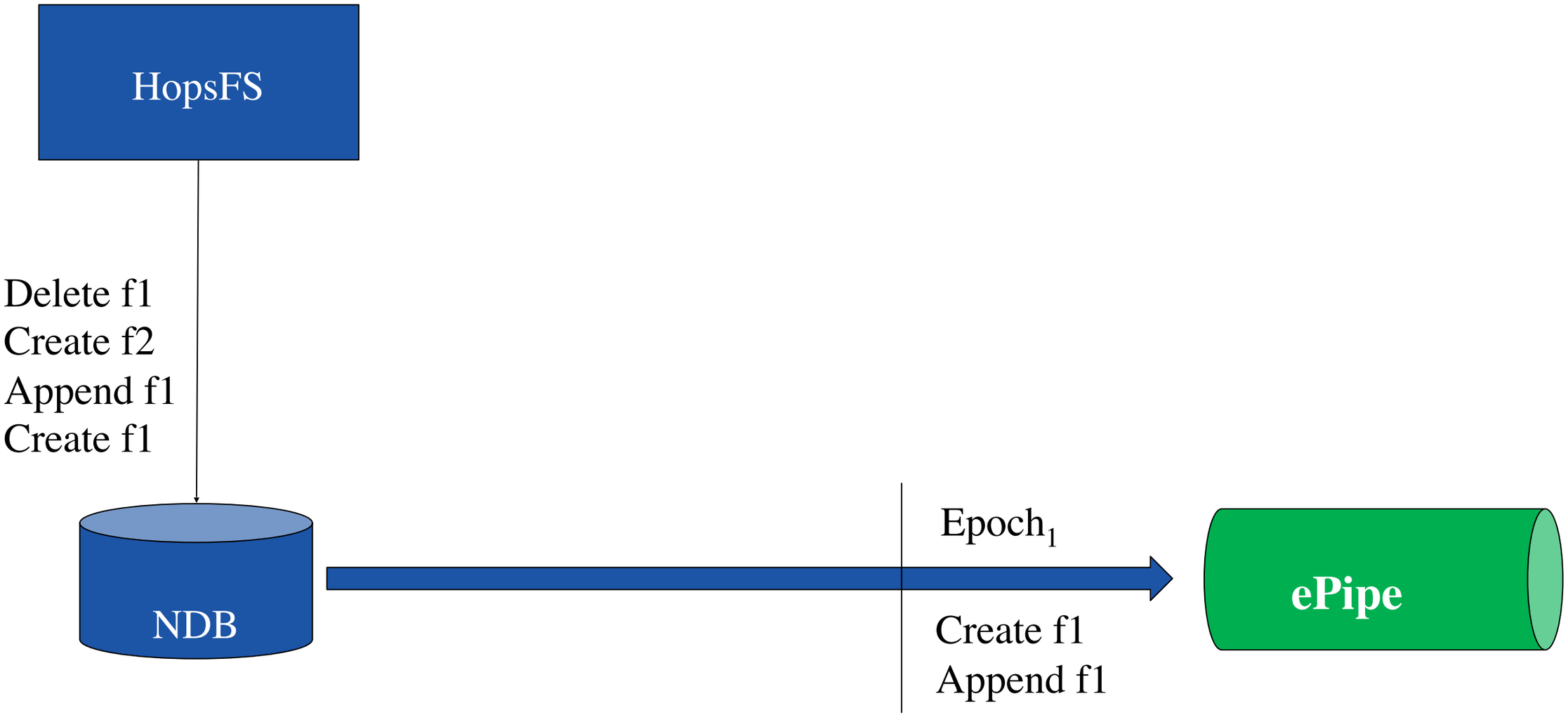
ePipe


HopsFS
Append f1
Create f1
NDB
Epoch$_1$
Create f1
Append f1
ePipe

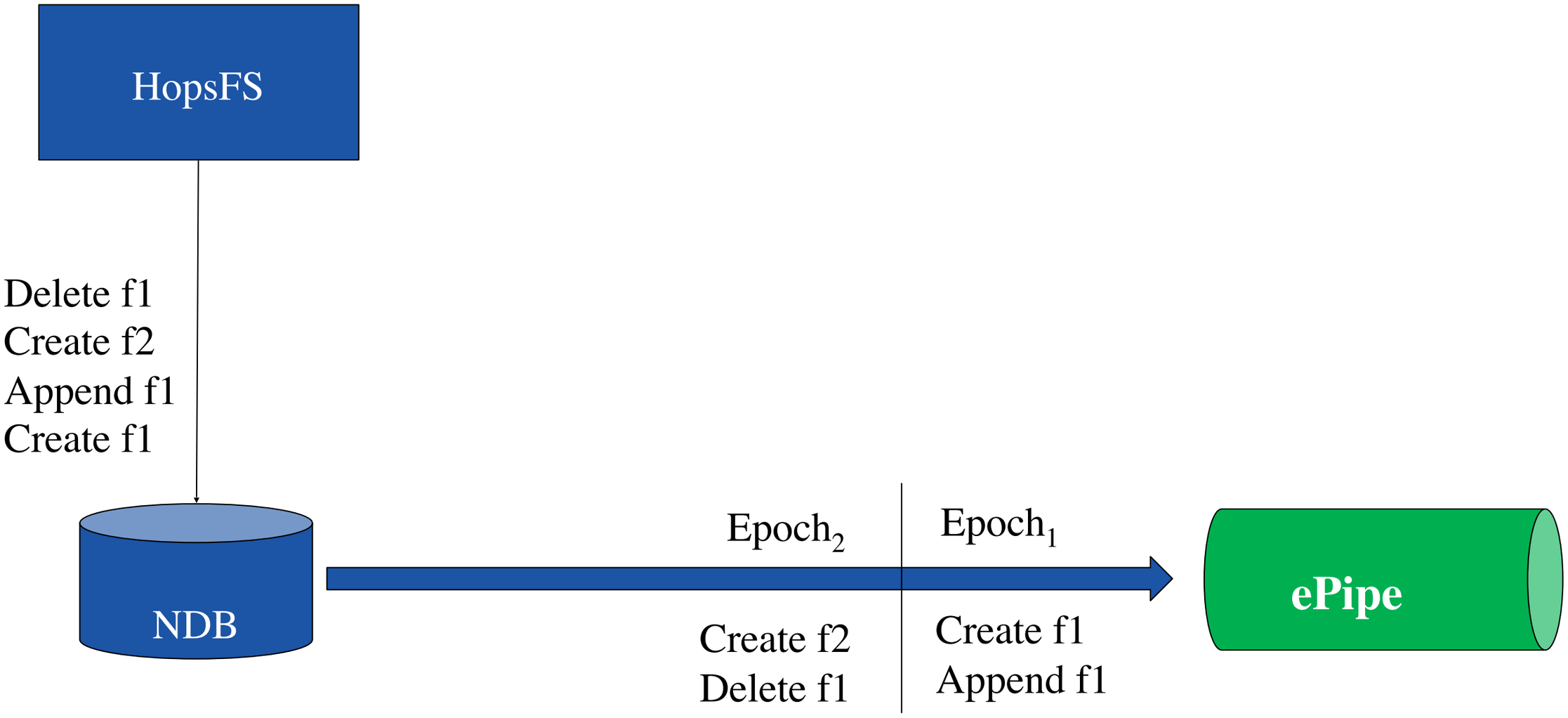
ePipe


HopsFS
Append f1
Create f1
NDB
Epoch$_1$
Create f1
Append f1
ePipe

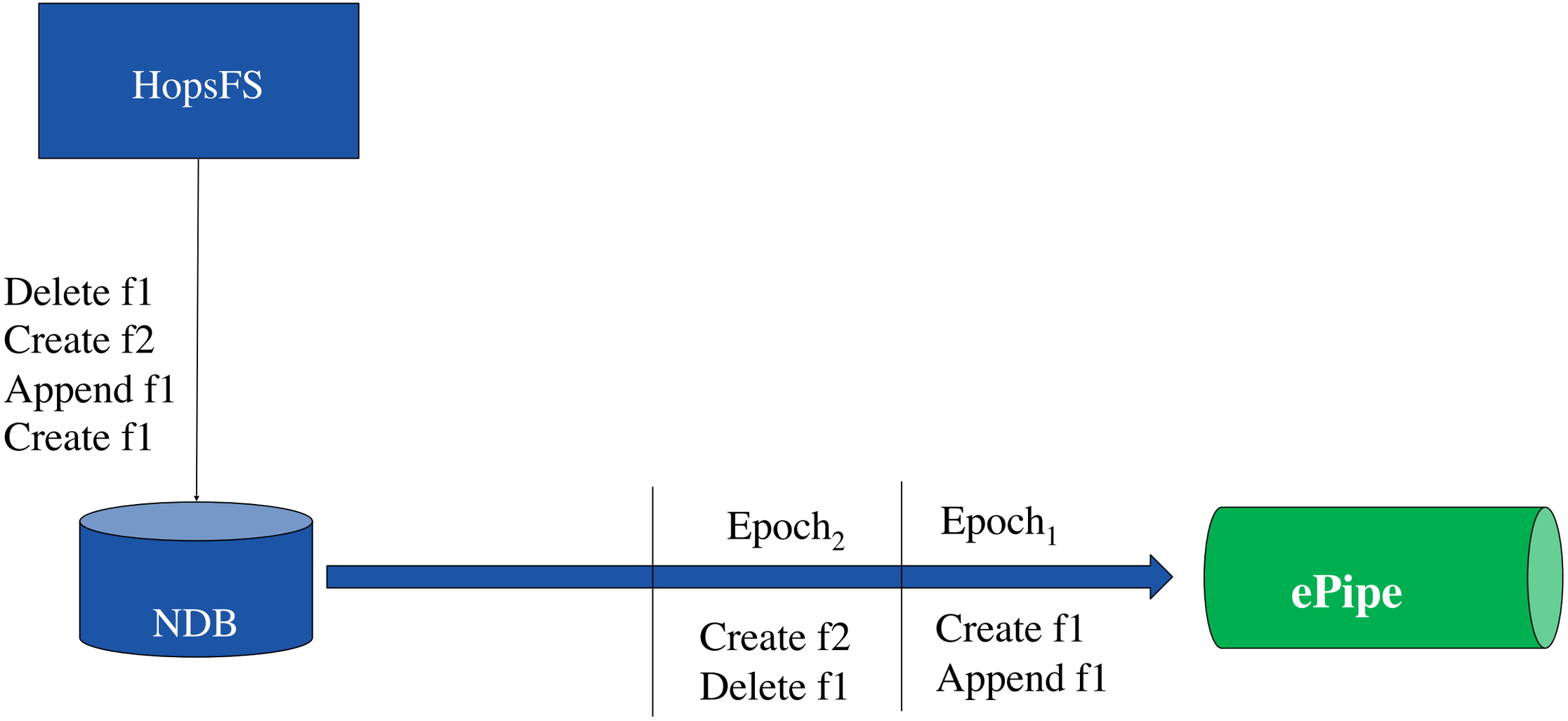
ePipe


HopsFS
Delete f1
Create f2
Append f1
Create f1
NDB
Epoch$_1$
Create f1
Append f1
ePipe

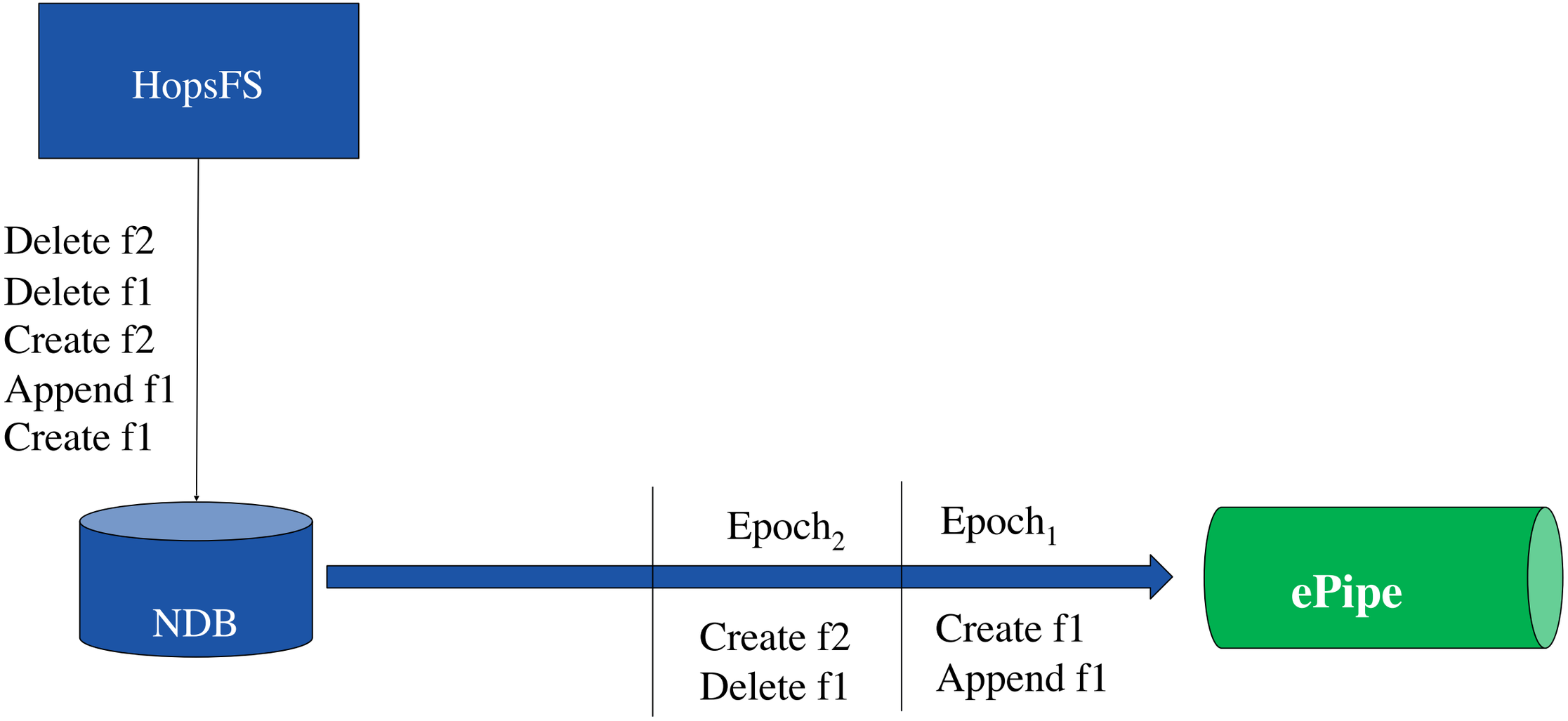
ePipe


HopsFS
Delete f1
Create f2
Append f1
Create f1
NDB
Epoch2
Epoch1
Create f2
Delete f1
Create f1
Append f1
ePipe

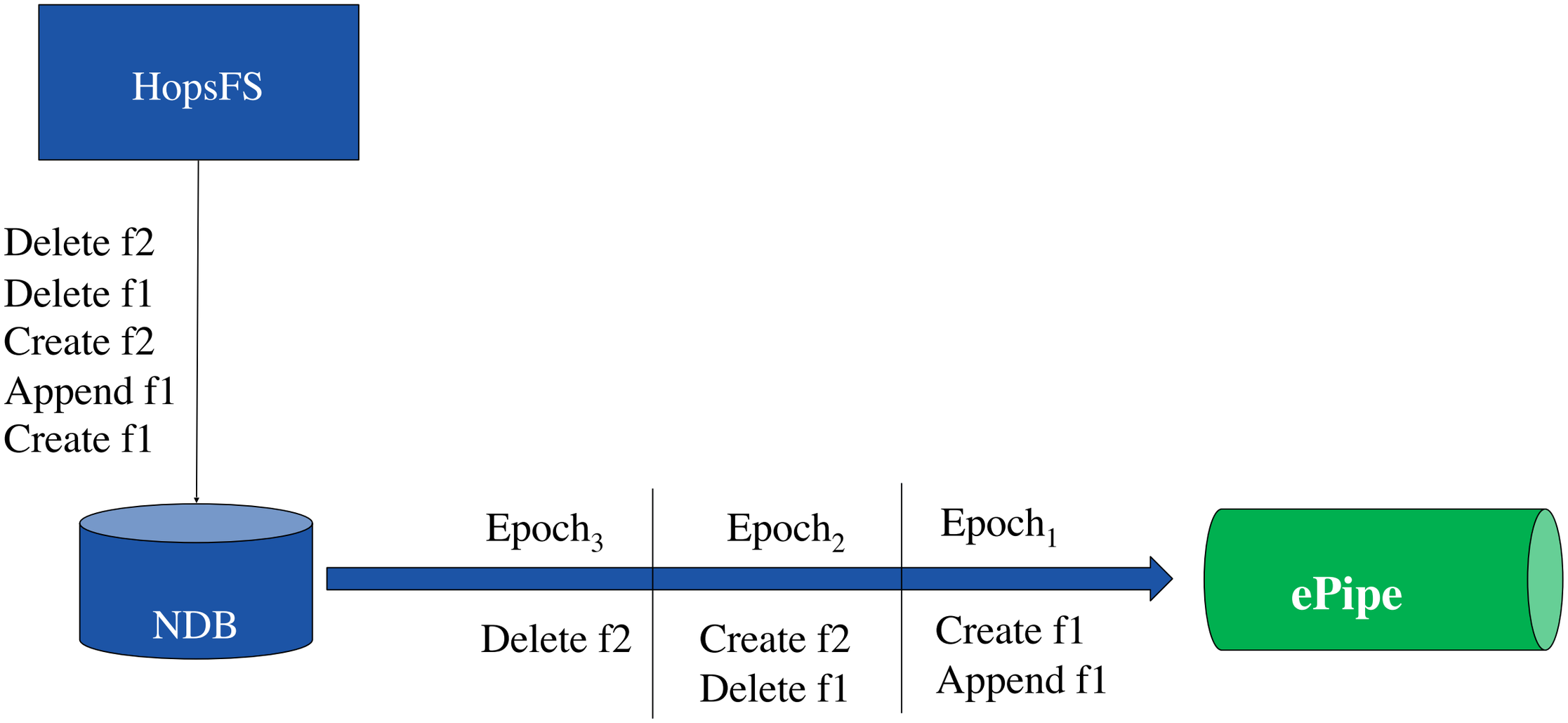
ePipe


HopsFS
Delete f1
Create f2
Append f1
Create f1
NDB
Epoch2
Epoch1
Create f2
Delete f1
Create f1
Append f1
ePipe

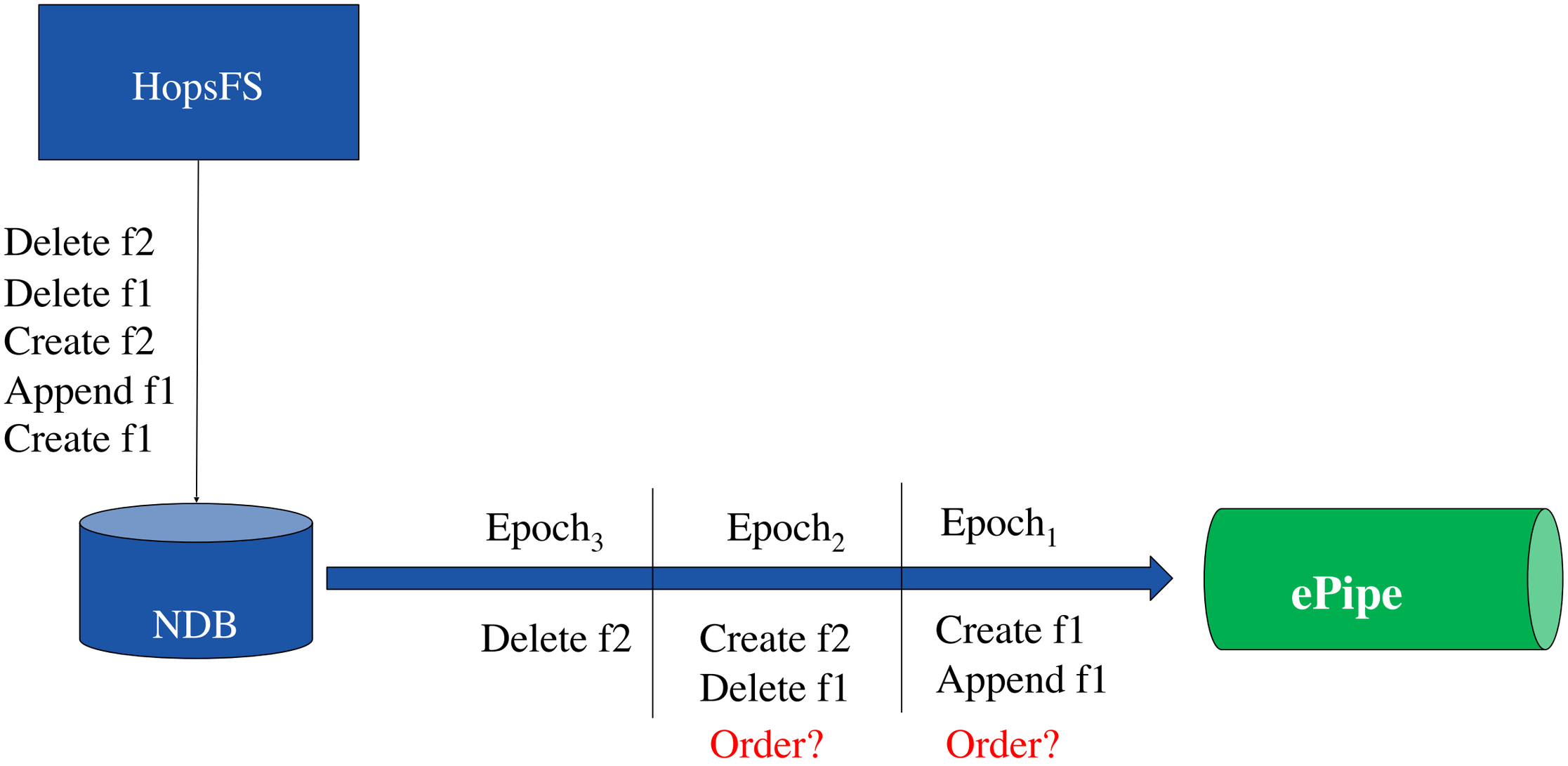
ePipe


HopsFS
Delete f2
Delete f1
Create f2
Append f1
Create f1
NDB
Epoch2
Epoch1
Create f2
Delete f1
Create f1
Append f1
ePipe

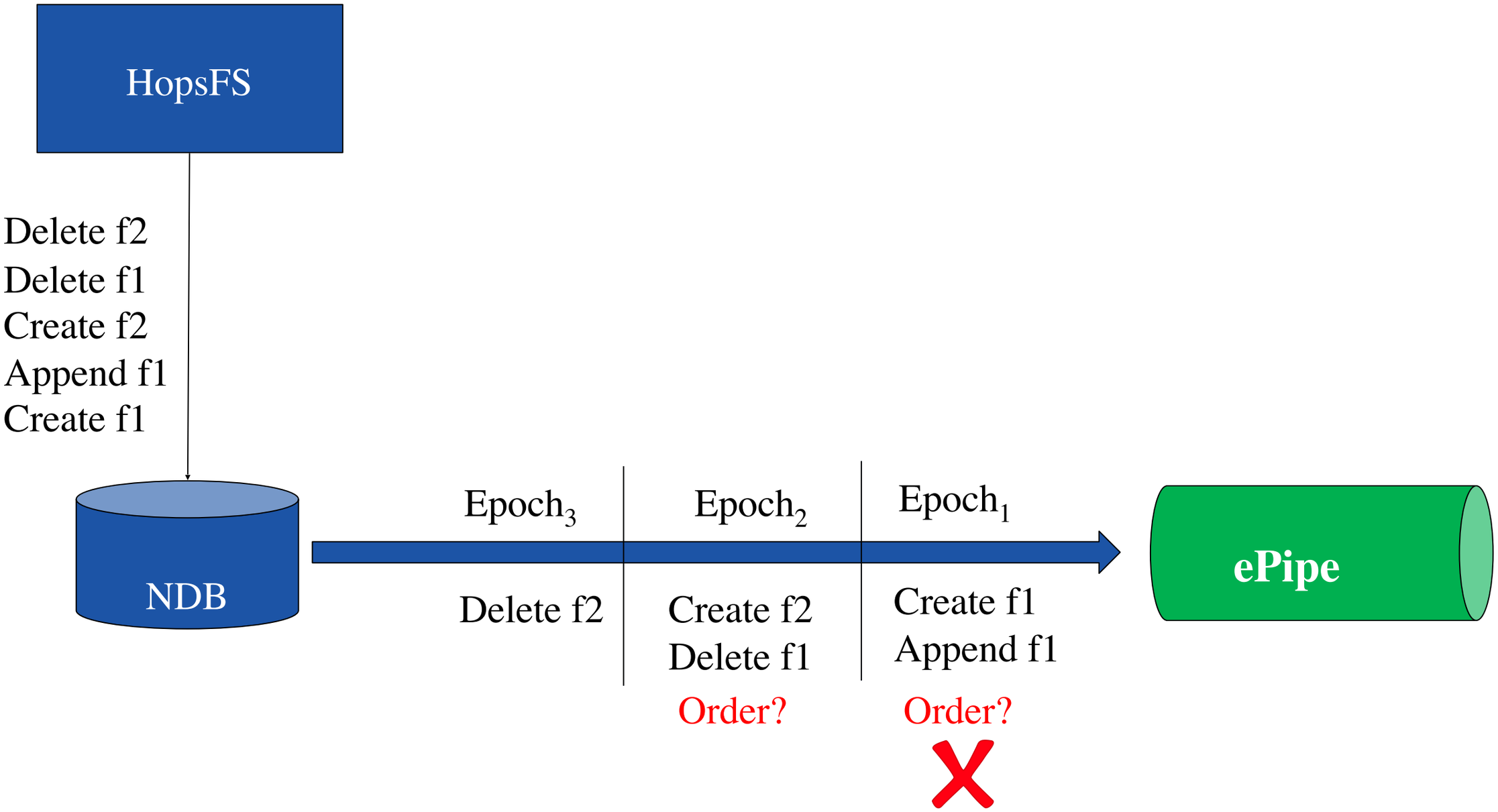
ePipe


HopsFS
Delete f2
Delete f1
Create f2
Append f1
Create f1
NDB
Epoch3
Epoch2
Epoch1
Delete f2
Create f2
Delete f1
Create f1
Append f1
ePipe

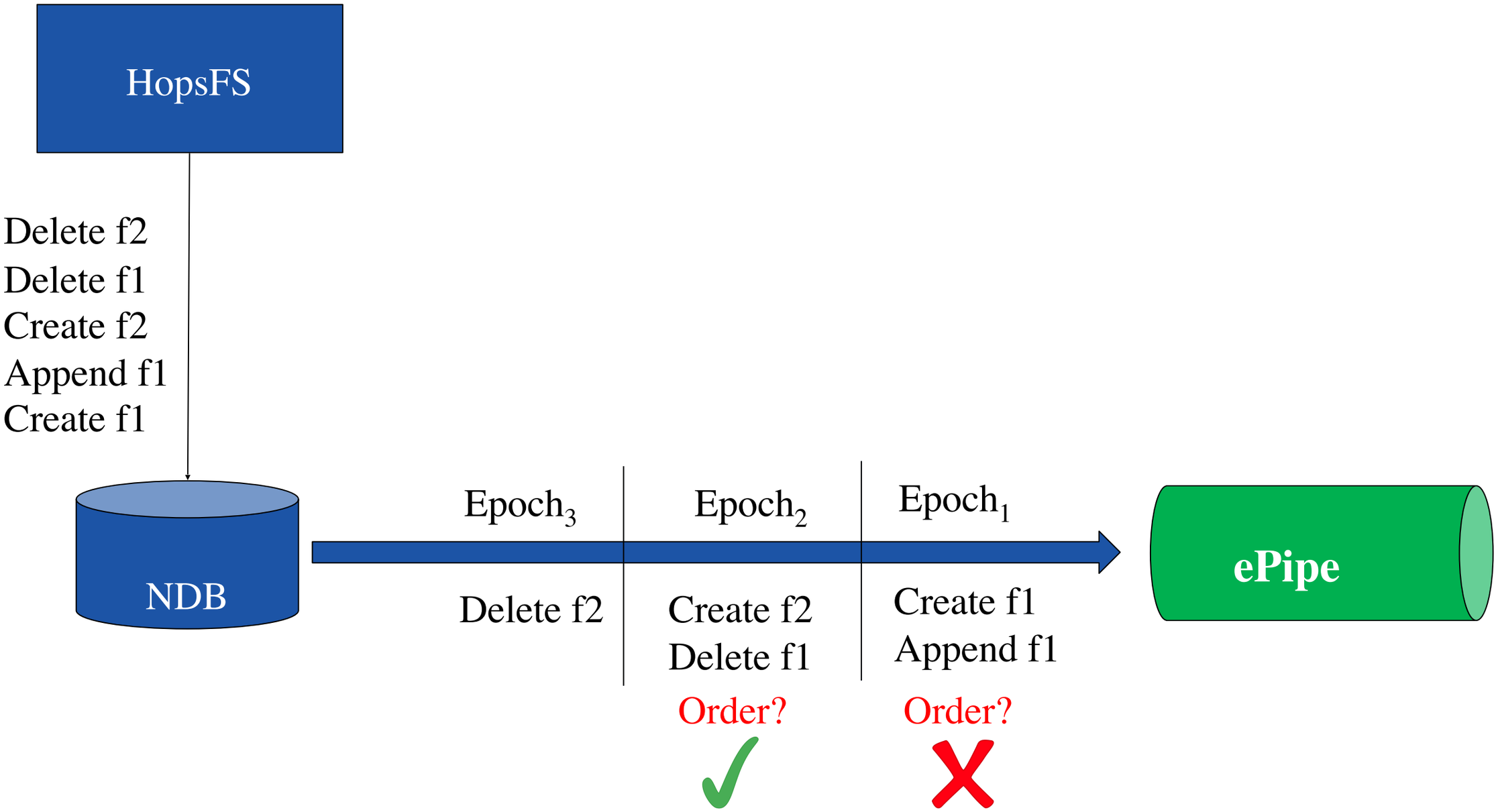
ePipe


HopsFS
Delete f2
Delete f1
Create f2
Append f1
Create f1
NDB
Epoch3
Epoch2
Epoch1
Delete f2
Create f2
Delete f1
Create f1
Append f1
Order?
Order?
ePipe

ePipe


HopsFS
Delete f2
Delete f1
Create f2
Append f1
Create f1
NDB
Epoch3
Epoch2
Epoch1
Delete f2
Create f2
Delete f1
Create f1
Append f1
Order?
Order?
ePipe

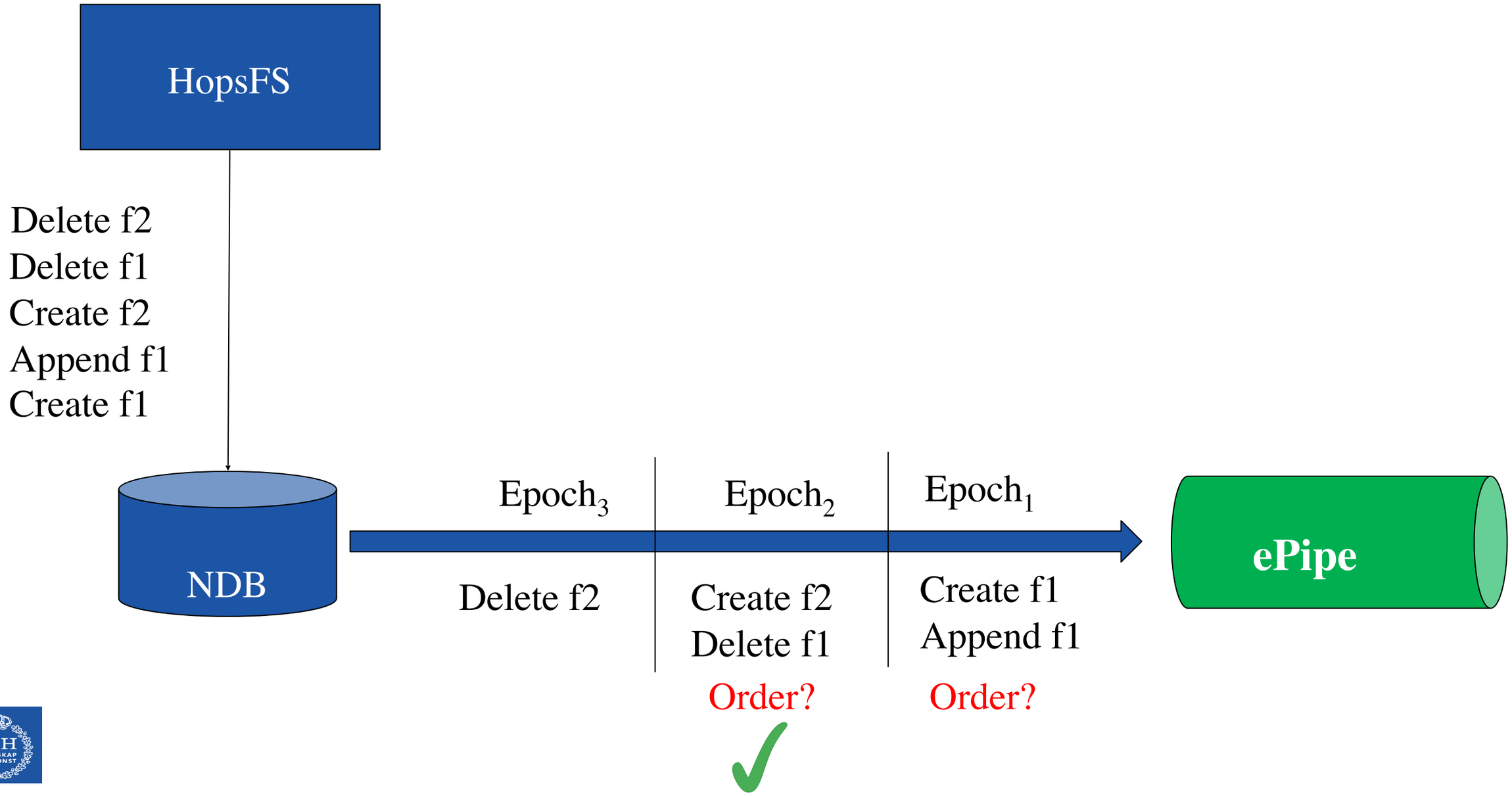
ePipe


HopsFS
Delete f2
Delete f1
Create f2
Append f1
Create f1
NDB
Epoch3
Epoch2
Epoch1
Delete f2
Create f2
Delete f1
Create f1
Append f1
Order?
Order?
ePipe

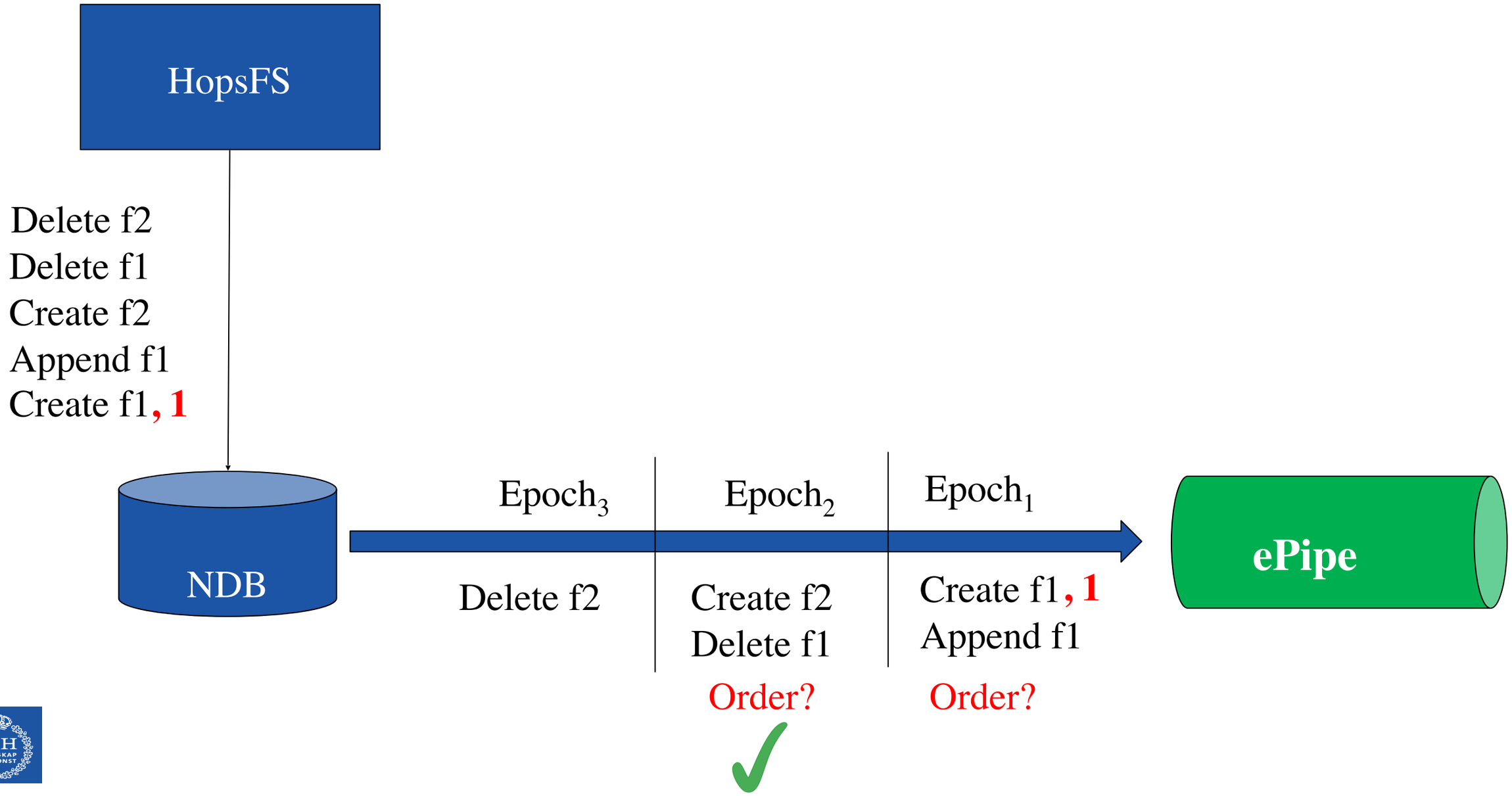
NDB Ordering Properties

- **Property 1:** epochs are totally ordered.
- **Property 2:** Changes within the same transaction happen in the same epoch.
- **Property 3:** Changes on files are ordered only if they are in different epochs, that is, no ordering is guaranteed within the same epoch

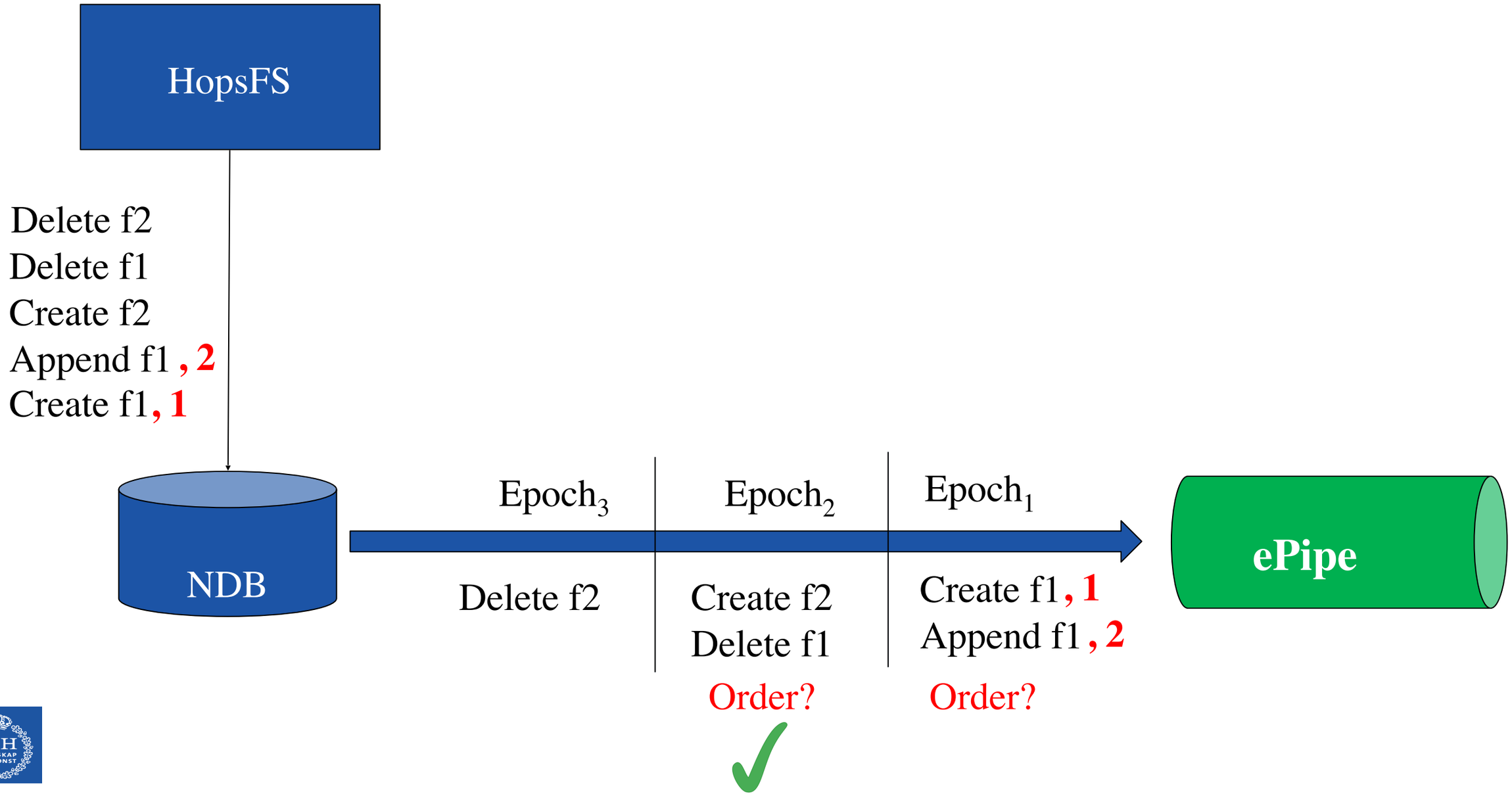
Adding version numbers



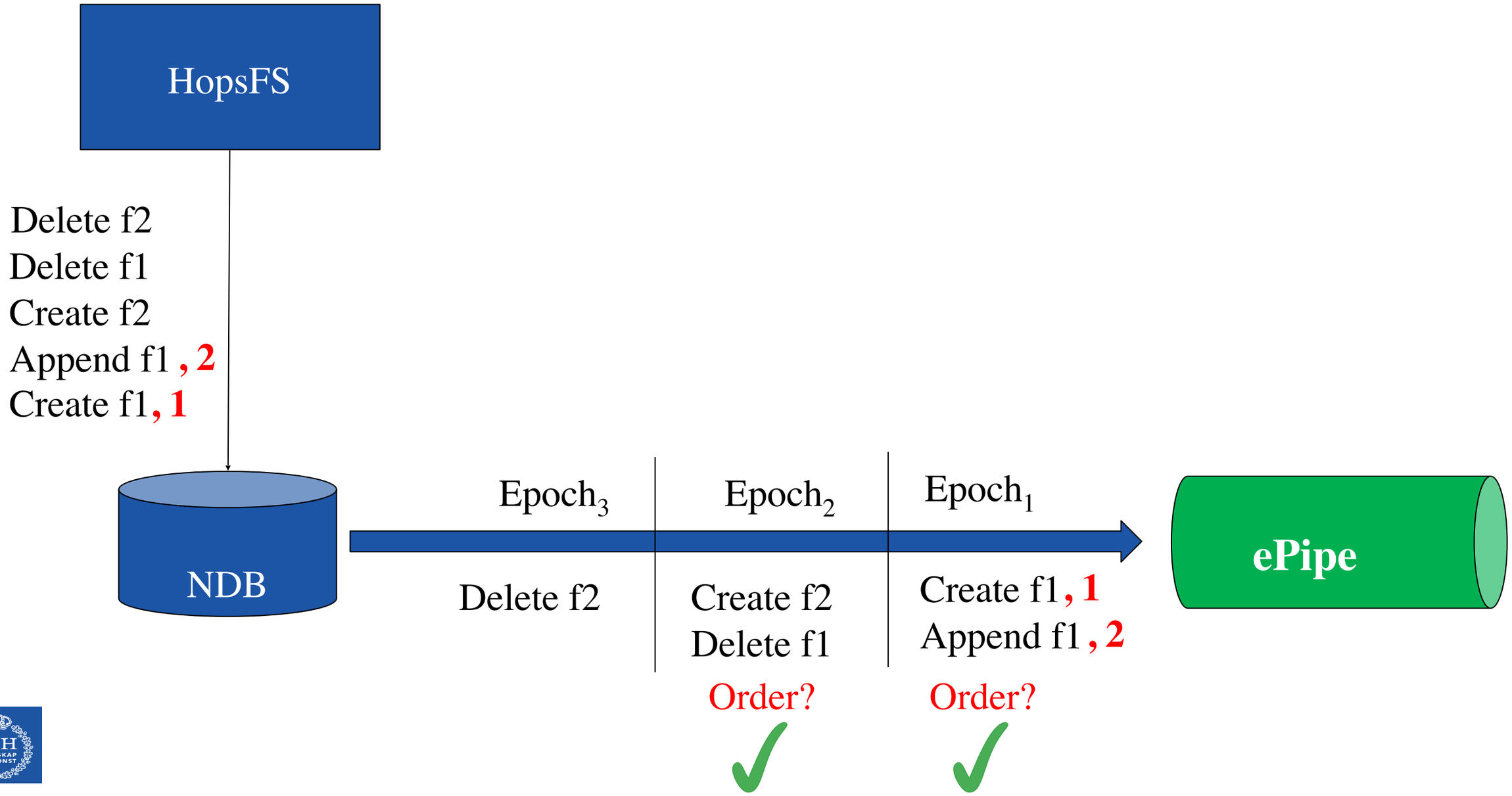
Adding version numbers



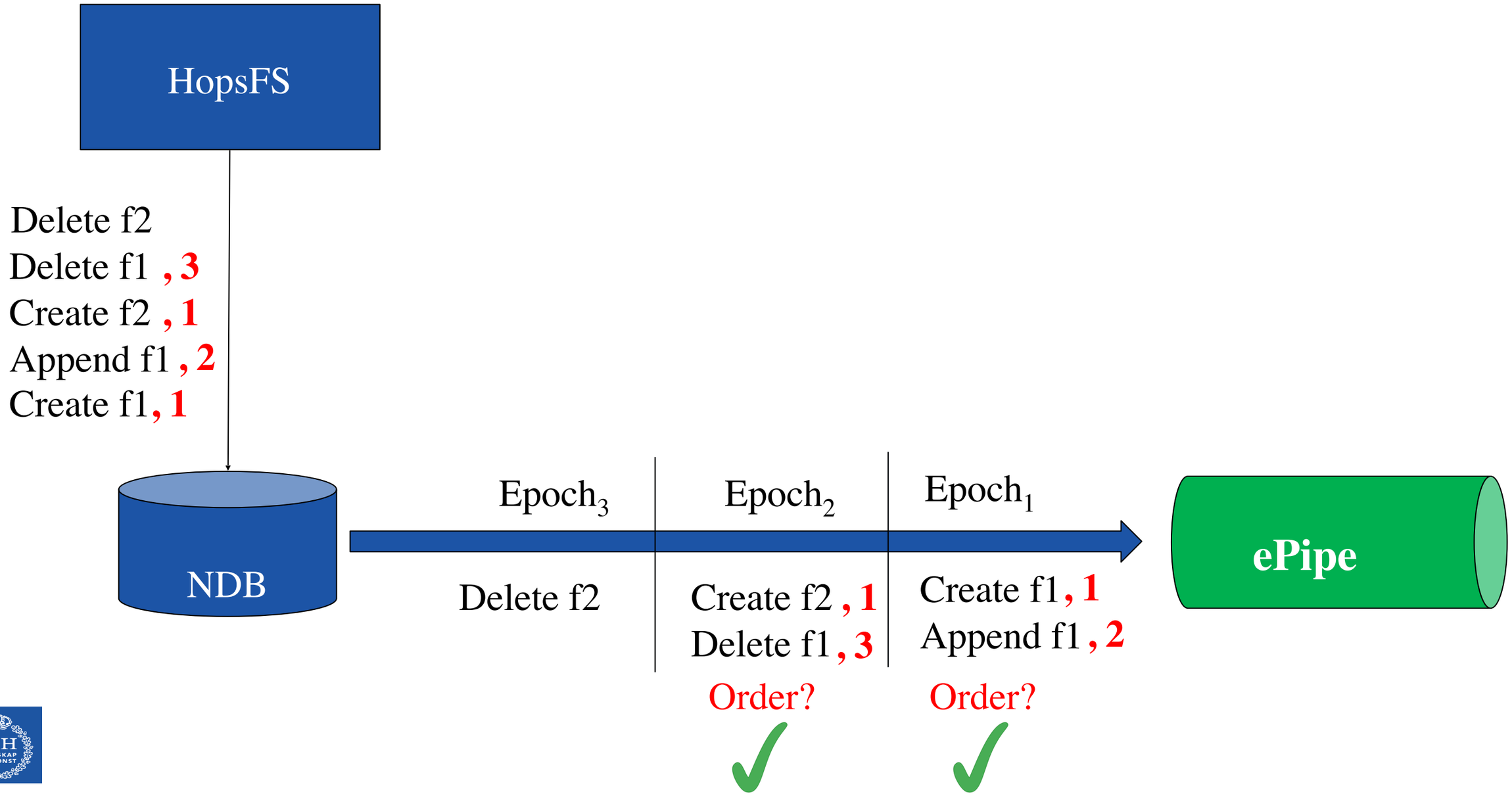
Adding version numbers



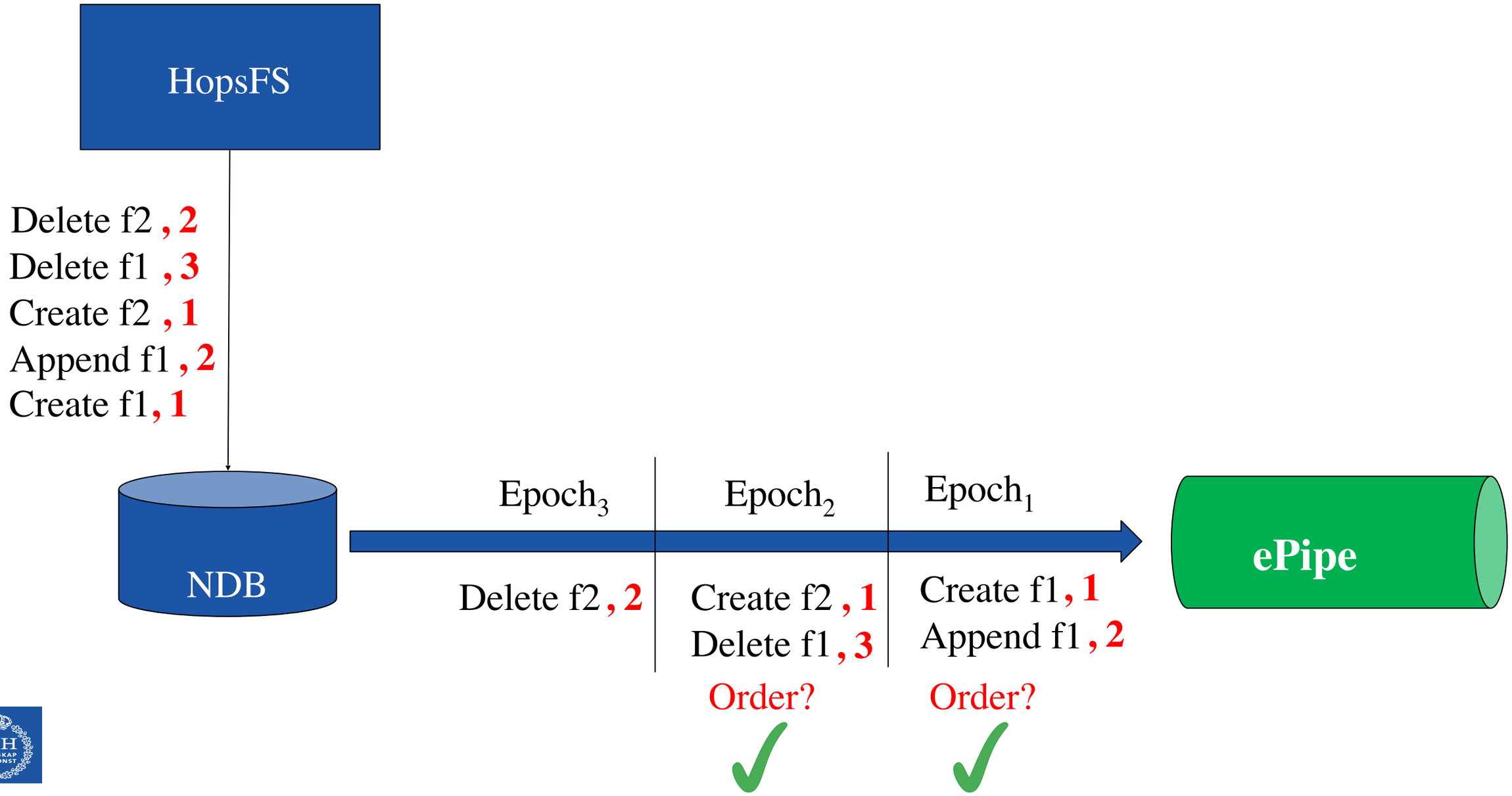
Adding version numbers



Adding version numbers



Adding version numbers



ePipe Ordering Properties

- **Property 4 & 5:** Version number ensures serializability of changes on the same file/directory within epochs.

- **Property 6:** The order of changes for different files/ directories within the same epoch doesn't matter.

ePipe

- Low replication lag (~100msec)
- High throughput

Requirements

- ~~Reading/Writing millions of images with high throughput~~
- ~~Attaching annotations to each image, and then searching using these annotations~~

Questions?