

# Is it time to revisit Erasure Coding in Data-intensive clusters?

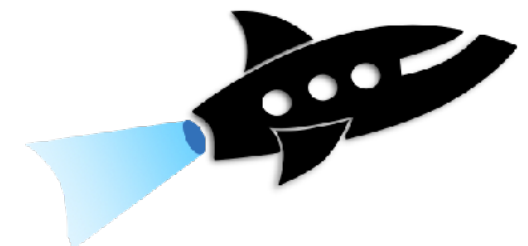
**Jad Darrous<sup>1</sup>, Shadi Ibrahim<sup>2</sup>, Christian Perez<sup>1</sup>**

<sup>1</sup>Univ. Lyon, Inria, CNRS, ENS de Lyon, UCBL, LIP, France

<sup>2</sup>Inria, IMT Atlantique, LS2N, France

MASCOTS

Rennes, 23 October 2019



# Context



# Context



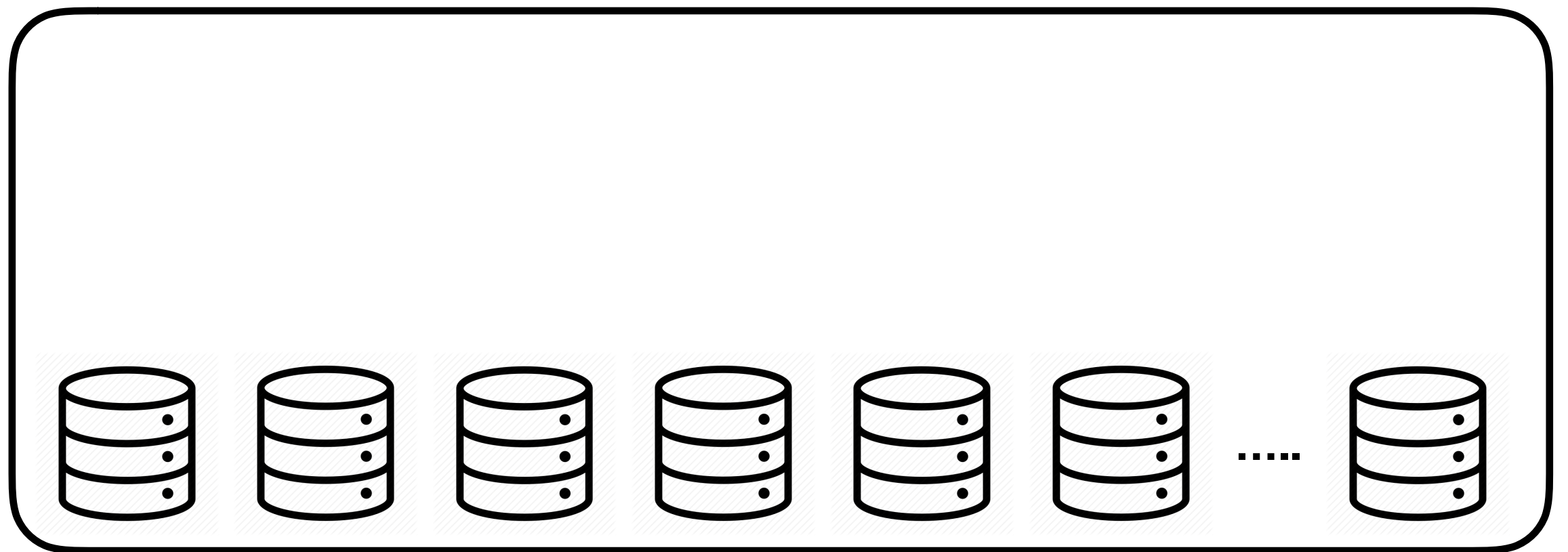
Billions of search queries are handled by Google every day

5 hours of video are uploaded to YouTube every second

350M photos are uploaded every day to Facebook

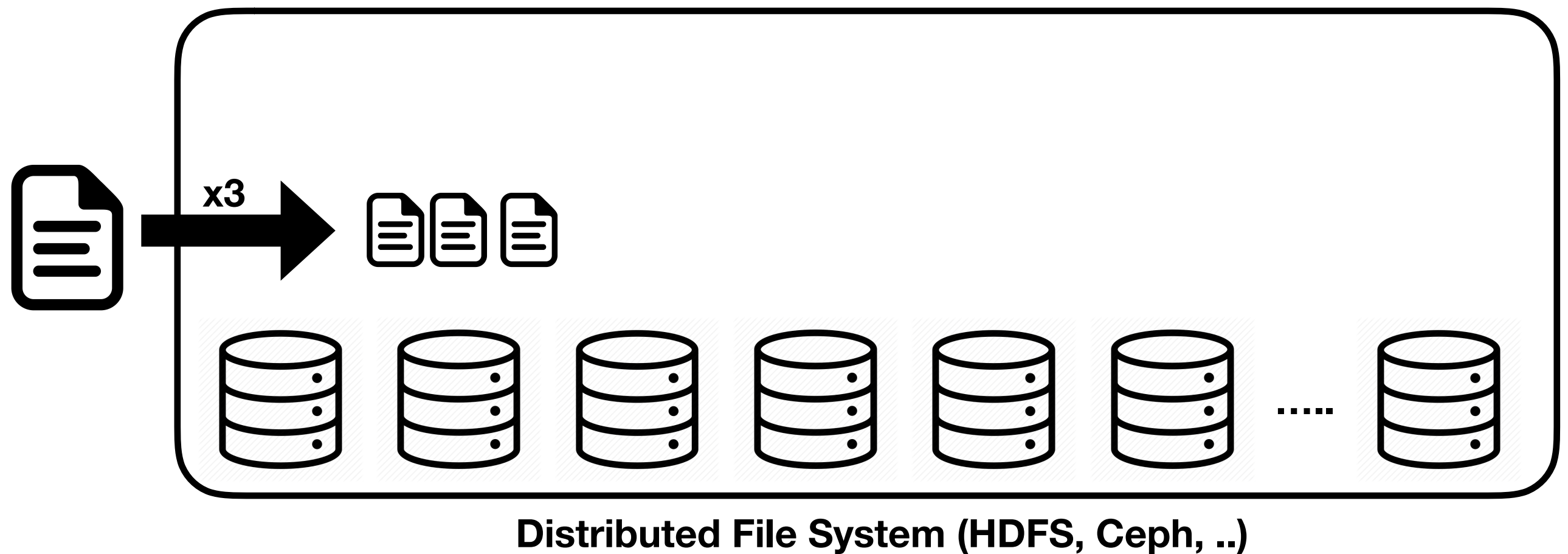
CERN recorded over 100 Petabytes of physics data

# Context



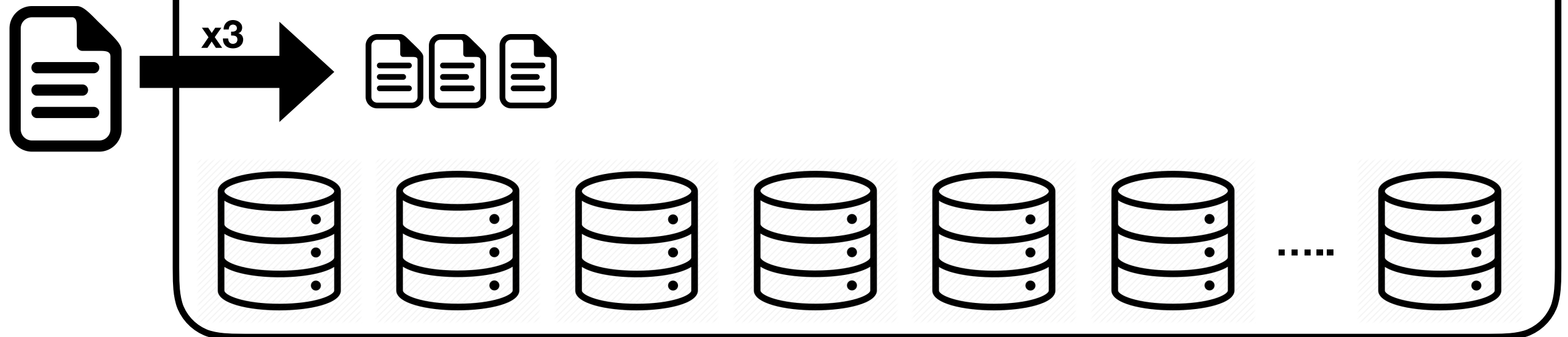
**Distributed File System (HDFS, Ceph, ..)**

# Context



# Context

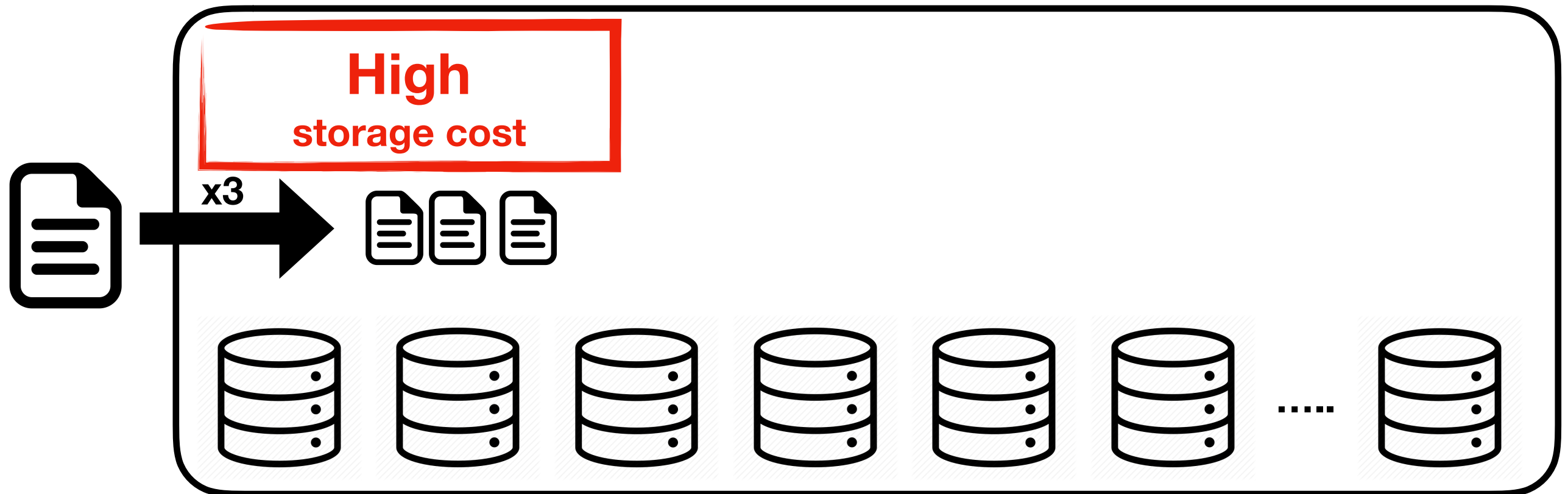
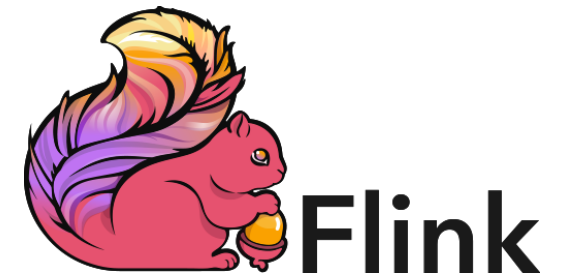
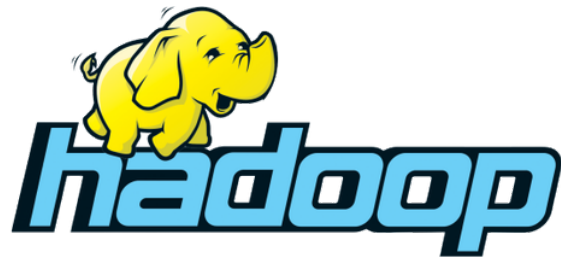
## Analytics frameworks



Distributed File System (HDFS, Ceph, ..)

# Context

## Analytics frameworks

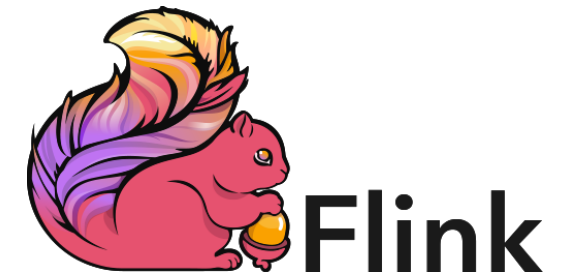


**Distributed File System (HDFS, Ceph, ..)**

1 Chowdhury et al., Leveraging Endpoint Flexibility in Data-Intensive Clusters, ACM SIGCOMM 2013

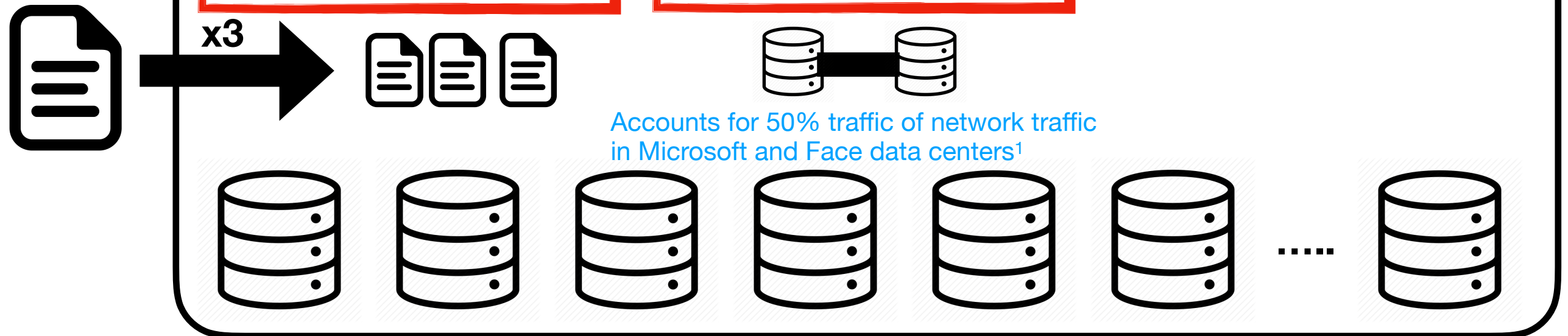
# Context

## Analytics frameworks



**High  
storage cost**

**High  
network cost**



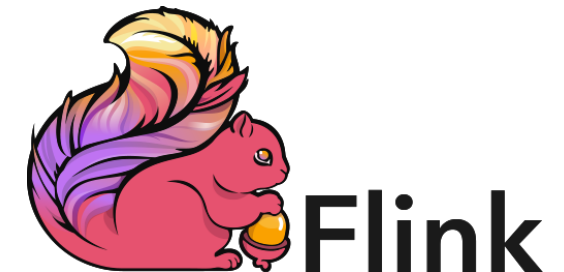
**Distributed File System (HDFS, Ceph, ..)**

<sup>1</sup> Chowdhury et al., Leveraging Endpoint Flexibility in Data-Intensive Clusters, ACM SIGCOMM 2013



# Context

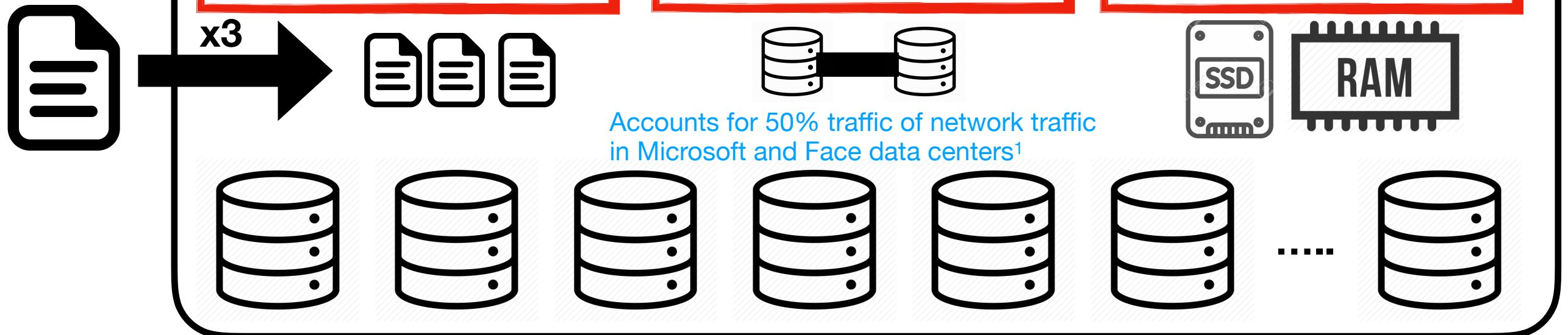
## Analytics frameworks



**High  
storage cost**

**High  
network cost**

**High  
hardware cost**



**Distributed File System (HDFS, Ceph, ..)**

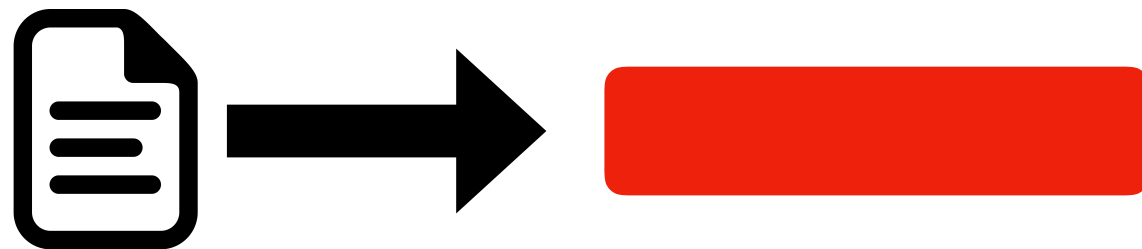
<sup>1</sup> Chowdhury et al., Leveraging Endpoint Flexibility in Data-Intensive Clusters, ACM SIGCOMM 2013

# Erasure coding

The case of Reed-Solomon RS(n, k)

# Erasure coding

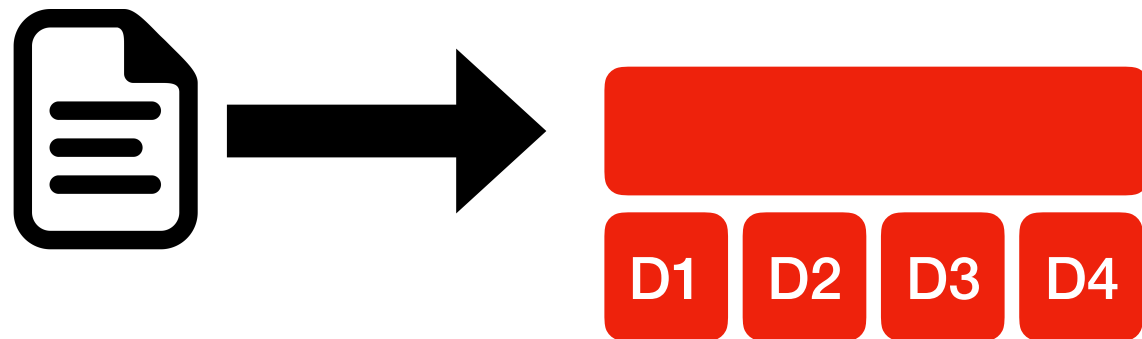
The case of Reed-Solomon RS(n, k)



# Erasure coding

The case of Reed-Solomon RS(n, k)

**D:** Data chunk  
**P:** Parity chunk



# Erasure coding

The case of Reed-Solomon RS(n, k)

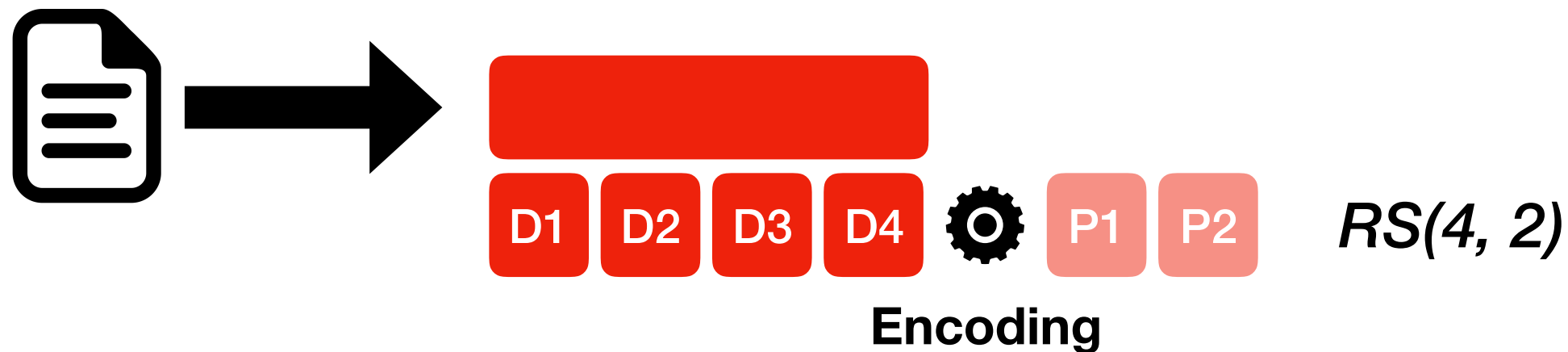
**D:** Data chunk  
**P:** Parity chunk



# Erasure coding

The case of Reed-Solomon  $RS(n, k)$

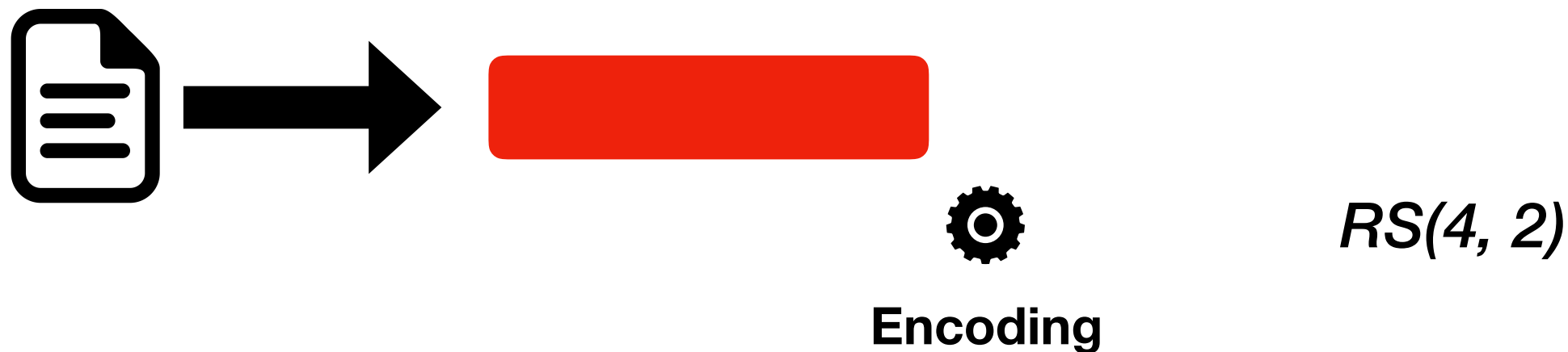
**D:** Data chunk  
**P:** Parity chunk



# Erasure coding

The case of Reed-Solomon  $RS(n, k)$

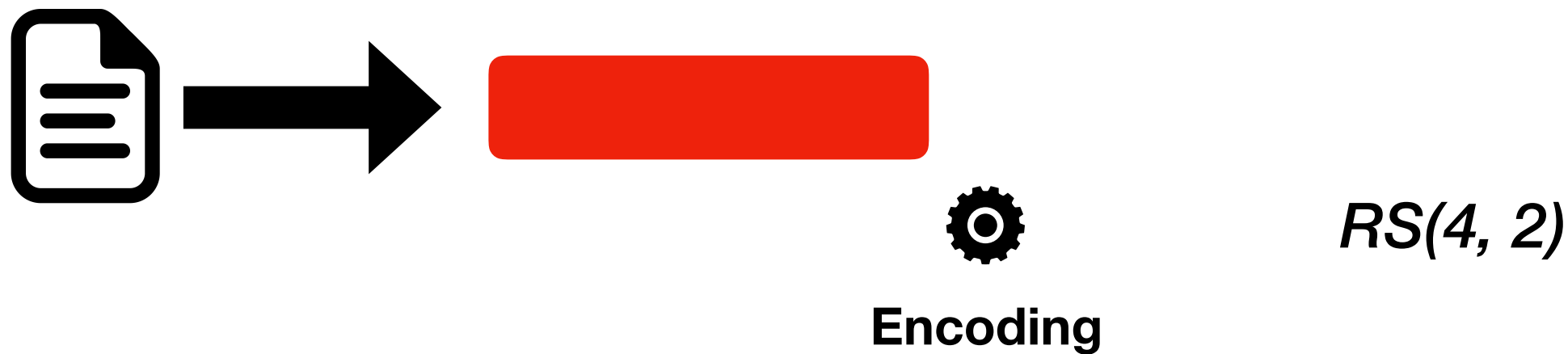
**D:** Data chunk  
**P:** Parity chunk



# Erasure coding

The case of Reed-Solomon  $RS(n, k)$

**D:** Data chunk  
**P:** Parity chunk

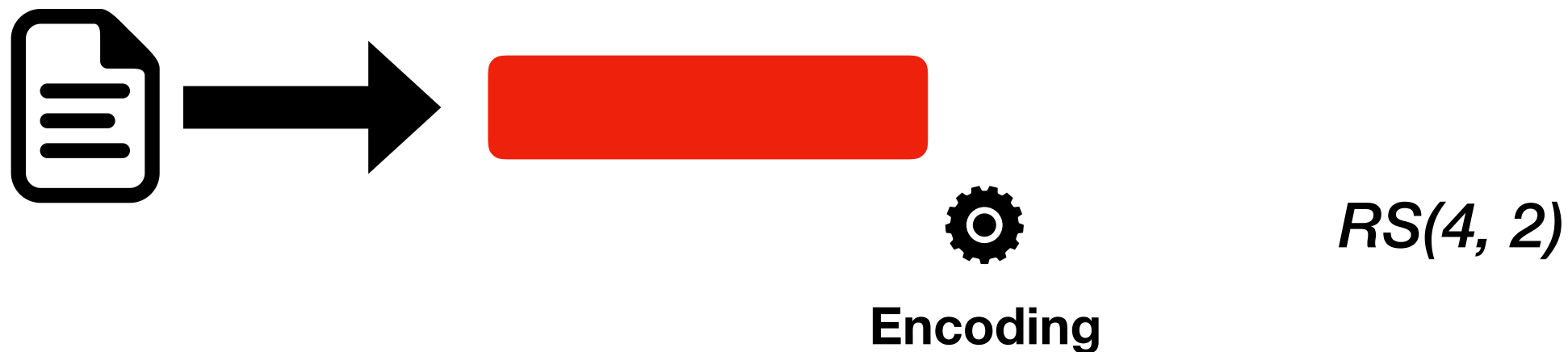




# Erasure coding

The case of Reed-Solomon  $RS(n, k)$

**D:** Data chunk  
**P:** Parity chunk



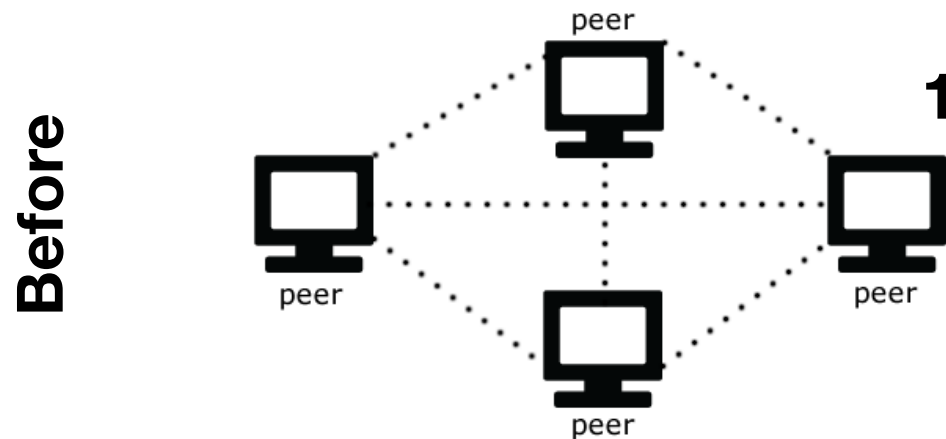
# Where we can find EC?


1 Kubiawicz et al., OceanStore: An Architecture for Global-scale Persistent Storage, ASPLOS'2000.

2 Haeberlen et al., Glacier: Highly durable, decentralized storage despite massive correlated failures, NSDI'2005.

3 Rashmi et al., EC-Cache: Load-Balanced, Low-Latency Cluster Caching with Online Erasure Coding, OSDI'2016.

# Where we can find EC?



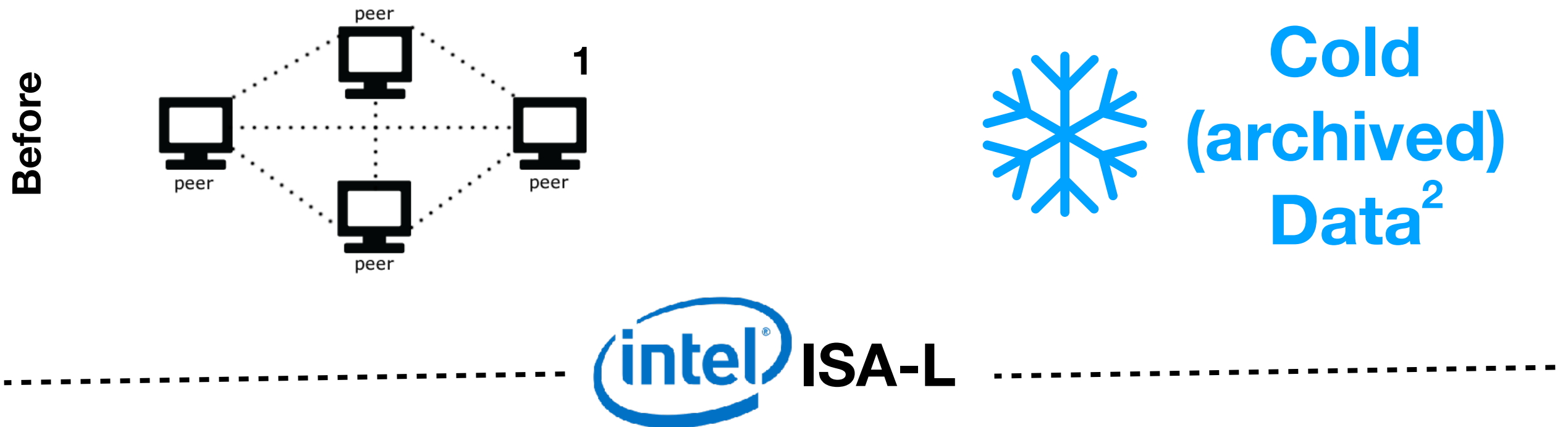
 Cold  
(archived)  
Data<sup>2</sup>

1 Kubiawicz et al., OceanStore: An Architecture for Global-scale Persistent Storage, ASPLOS'2000.

2 Haeberlen et al., Glacier: Highly durable, decentralized storage despite massive correlated failures, NSDI'2005.

3 Rashmi et al., EC-Cache: Load-Balanced, Low-Latency Cluster Caching with Online Erasure Coding, OSDI'2016.

# Where we can find EC?



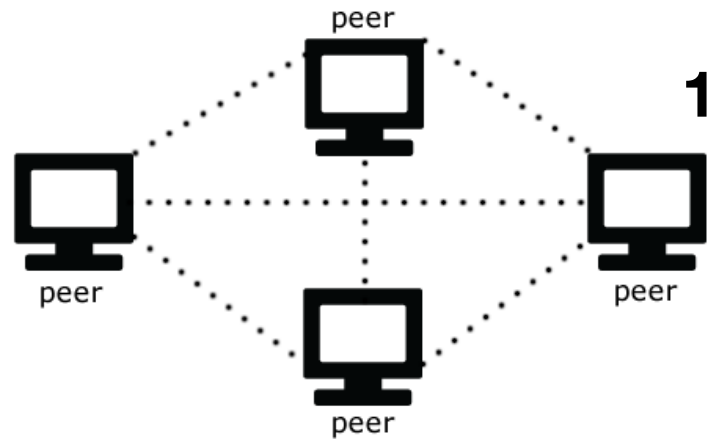
1 Kubiawicz et al., OceanStore: An Architecture for Global-scale Persistent Storage, ASPLOS'2000.


2 Haeberlen et al., Glacier: Highly durable, decentralized storage despite massive correlated failures, NSDI'2005.

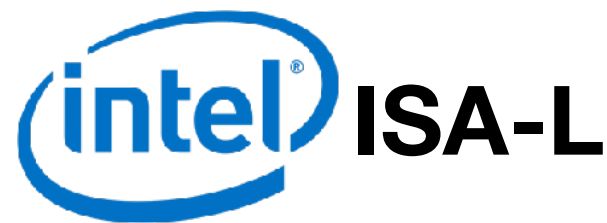
3 Rashmi et al., EC-Cache: Load-Balanced, Low-Latency Cluster Caching with Online Erasure Coding, OSDI'2016.

# Where we can find EC?

Before




 **Cold  
(archived)  
Data<sup>2</sup>**



2014 - Now



**HDFS 3.0.0**

 **Hot  
(cached)  
Data<sup>3</sup>**


1 Kubiawicz et al., OceanStore: An Architecture for Global-scale Persistent Storage, ASPLOS'2000.

2 Haeberlen et al., Glacier: Highly durable, decentralized storage despite massive correlated failures, NSDI'2005.

3 Rashmi et al., EC-Cache: Load-Balanced, Low-Latency Cluster Caching with Online Erasure Coding, OSDI'2016.

**What is the performance characteristics of analysis jobs under erasure coded data?**

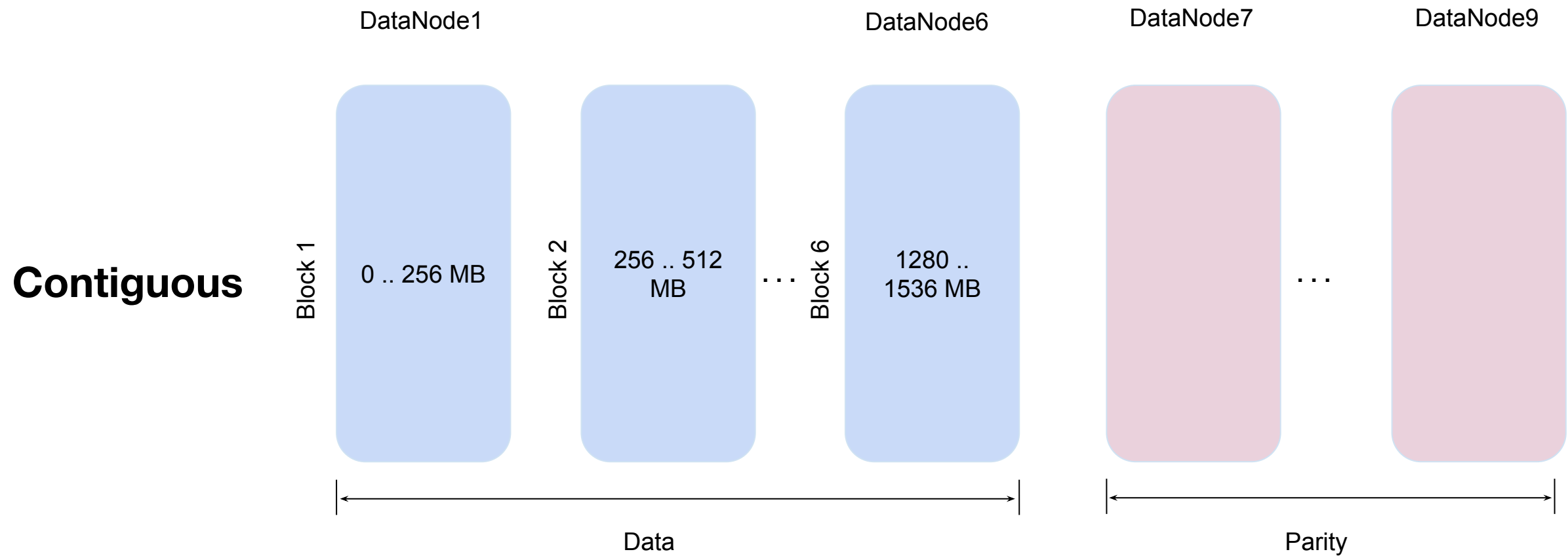
# What is the performance characteristics of analysis jobs under erasure coded data?

- ✓ Lower storage overhead, compared to replication
- ✓ Low CPU overhead, thanks to  ISA-L
- ✗ High disk and network overhead for data recovery

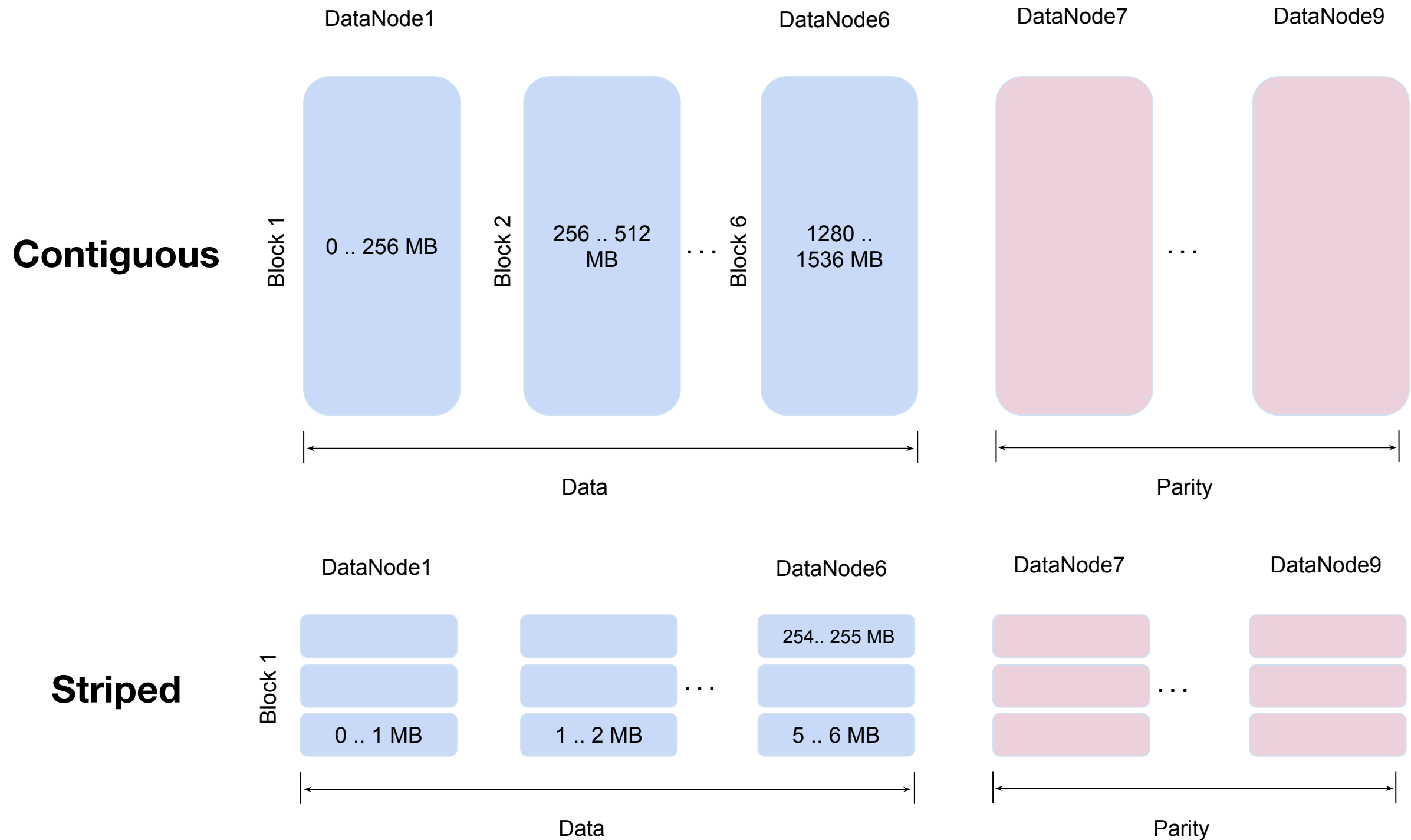
# Block layout



# Block layout



# Block layout



# Advantages of striped layout

- ✓ More efficient for small files
- ✓ Encoding and decoding require less memory overhead
- ✓ Allows parallel I/O when accessing the data block
- ✓ Low *degraded read* overhead

# Advantages of striped layout

- ✓ More efficient for small files
- ✓ Encoding and decoding require less memory overhead
- ✓ Allows parallel I/O when accessing the data block
- ✓ Low *degraded read* overhead

➡ Striped layout is implemented in HDFS 3.0.0

# Advantages of striped layout

- ✓ More efficient for small files
- ✓ Encoding and decoding require less memory overhead
- ✓ Allows parallel I/O when accessing the data block
- ✓ Low *degraded read* overhead

➡ Striped layout is implemented in HDFS 3.0.0

✗ No data locality

# Methodology

# Methodology



## Cluster

**21** machines with two Intel Xeon E5-2660 8-cores processors, 64GB of main memory, and one HDD at 7.2k RPM with 1TB  
10 Gigabit Ethernet network

# Methodology



HDFS

Block size: **256 MB** - Replication factor: **3** - EC policy: **RS(6, 3)**  
with 1 MB cell size

Cluster

**21** machines with two Intel Xeon E5-2660 8-cores processors,  
64GB of main memory, and one HDD at 7.2k RPM with 1TB  
10 Gigabit Ethernet network



# Methodology



YARN

HDFS

8 containers per node (one per core with 1GB memory)

Block size: **256 MB** - Replication factor: **3** - EC policy: **RS(6, 3)**  
with 1 MB cell size

Cluster

**21** machines with two Intel Xeon E5-2660 8-cores processors,  
64GB of main memory, and one HDD at 7.2k RPM with 1TB  
10 Gigabit Ethernet network

# Methodology



Hadoop MapReduce

**Sort** (shuffle intensive)

**Wordcount** (map intensive)

**Kmeans** (Machine Learning application)

YARN

HDFS

8 containers per node (one per core with 1GB memory)

Block size: **256 MB** - Replication factor: **3** - EC policy: **RS(6, 3)**  
with 1 MB cell size

Cluster

**21** machines with two Intel Xeon E5-2660 8-cores processors,  
64GB of main memory, and one HDD at 7.2k RPM with 1TB  
10 Gigabit Ethernet network

# Methodology

- Metrics
  - Job execution time (seconds)
- Software configurations
  - Overlapping (default Hadoop) and non-overlapping shuffle (like Spark)
  - The impact of disk persistence
  - The impact of failures
  - The impact of RS schemes
- Hardware configurations
  - Storage Device: HDD and MEM
  - Network bandwidth: 1 Gbps and 10 Gbps

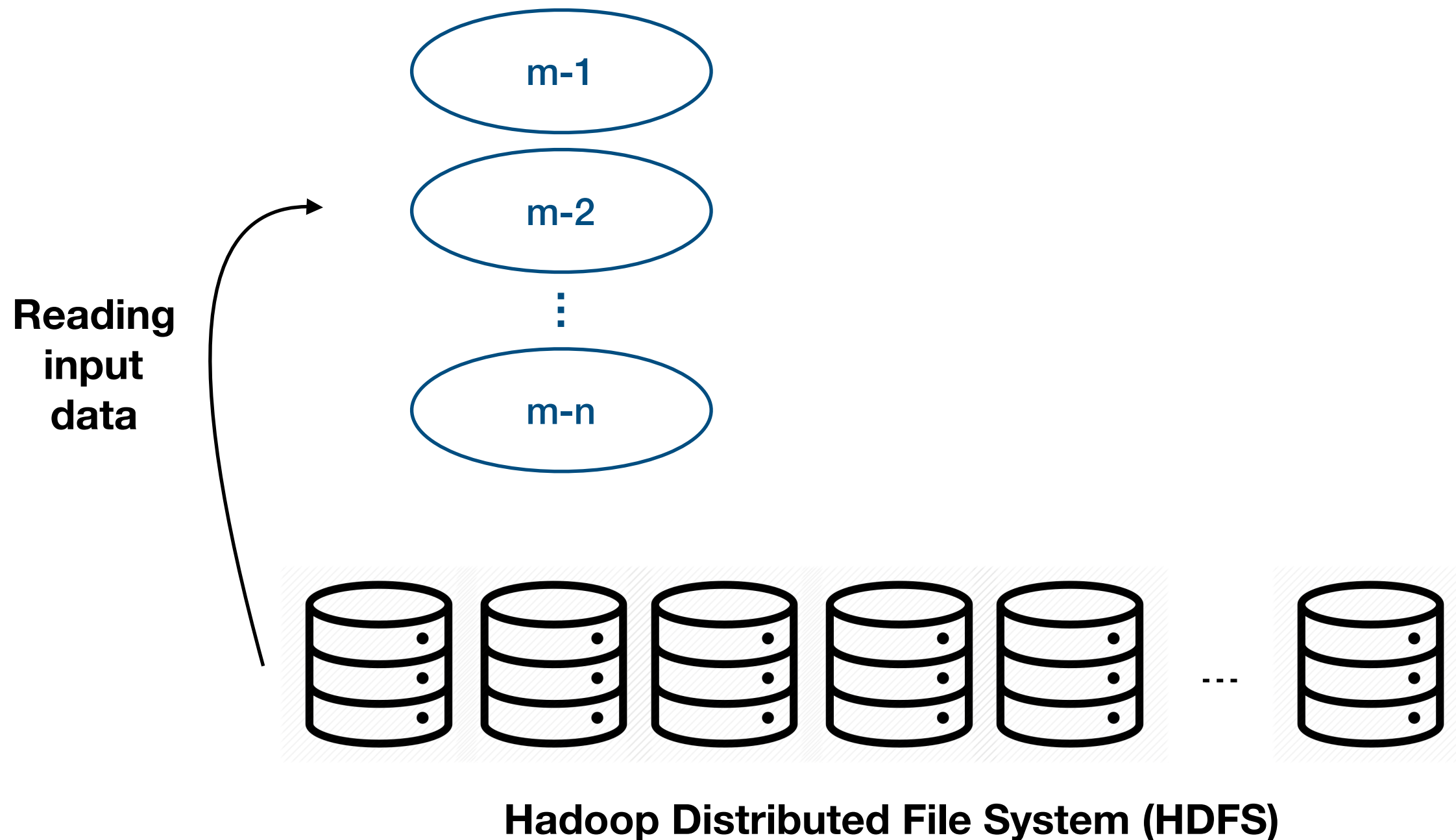
# Hadoop MapReduce



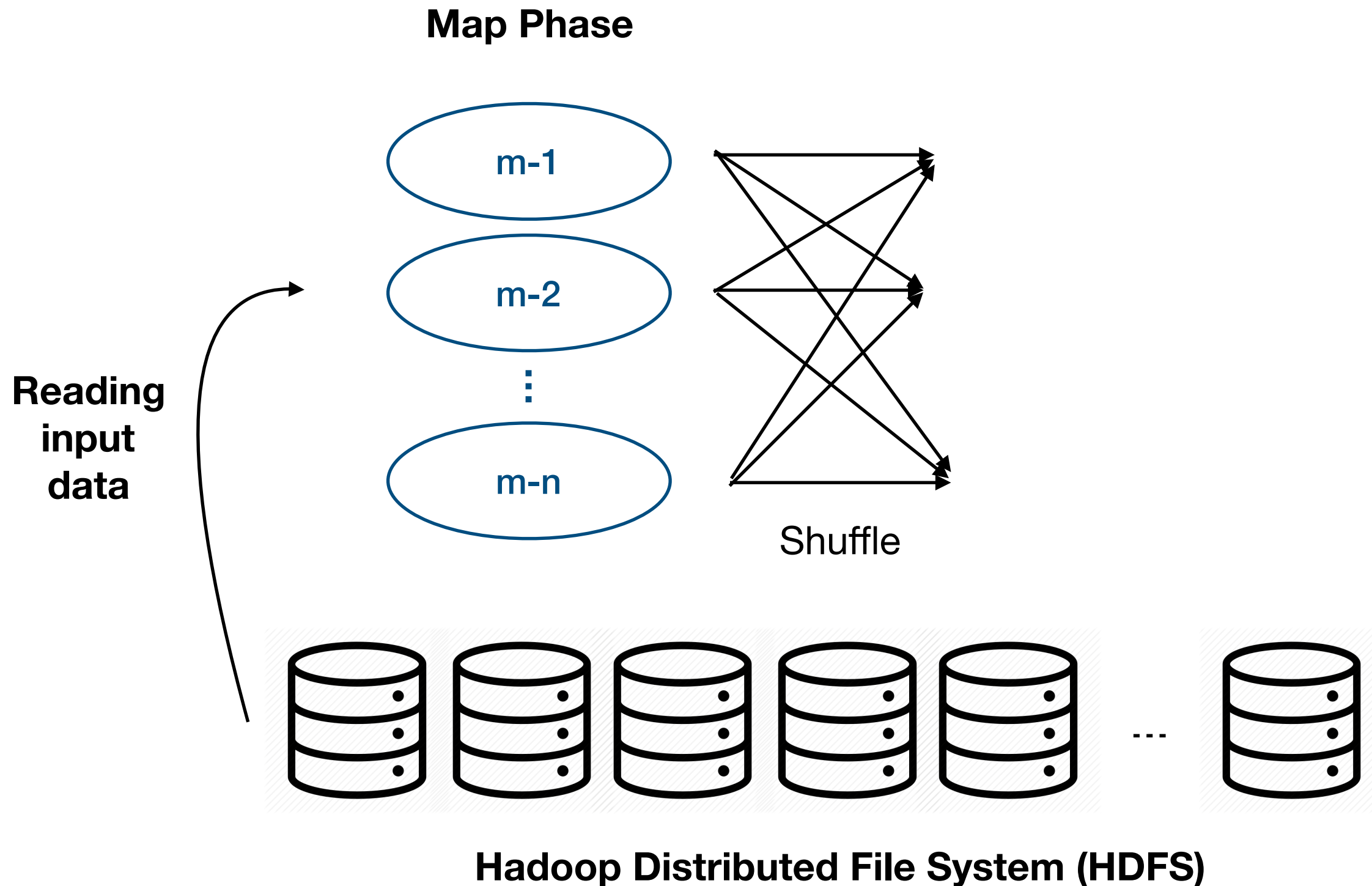
**Hadoop Distributed File System (HDFS)**

# Hadoop MapReduce

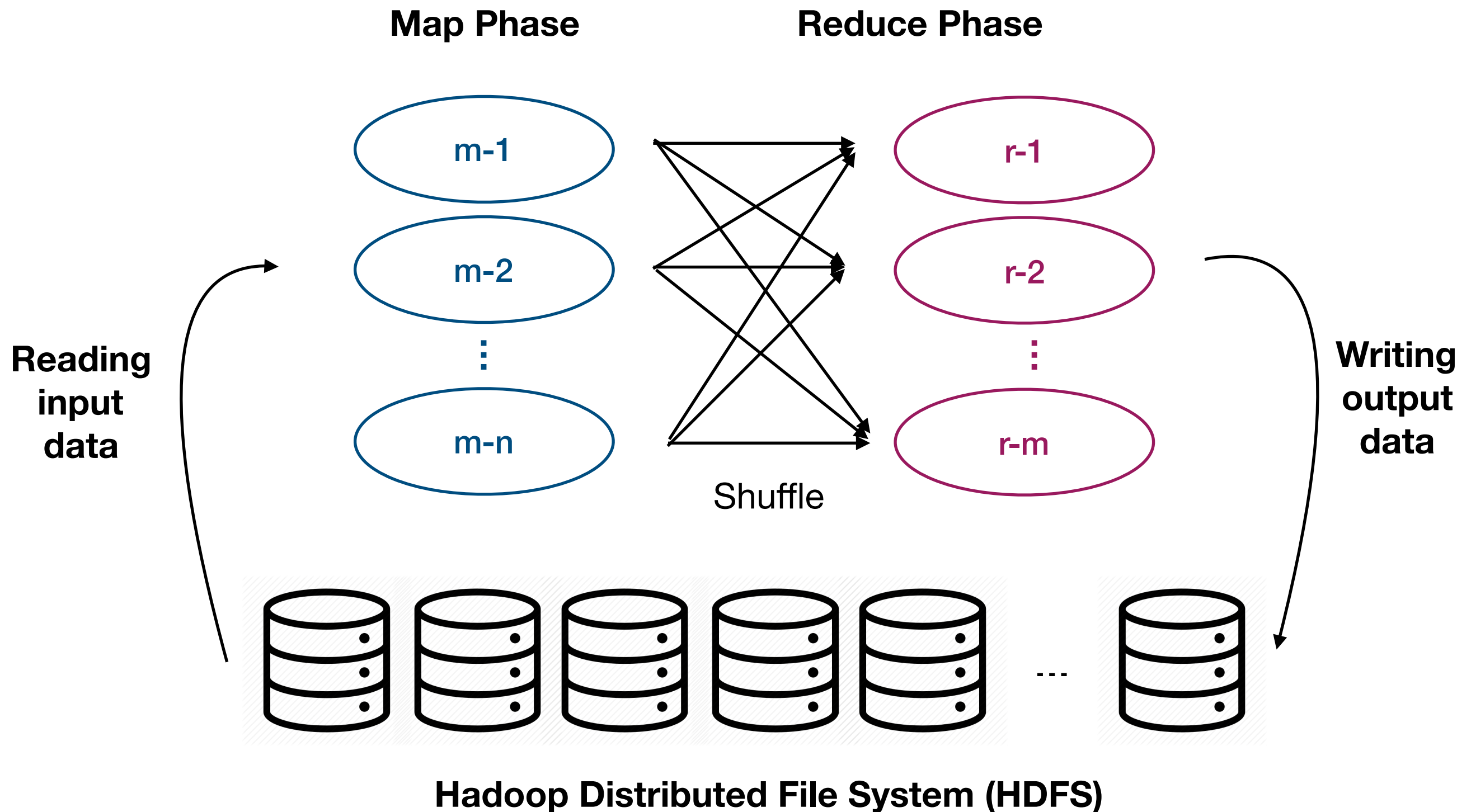
## Map Phase



# Hadoop MapReduce



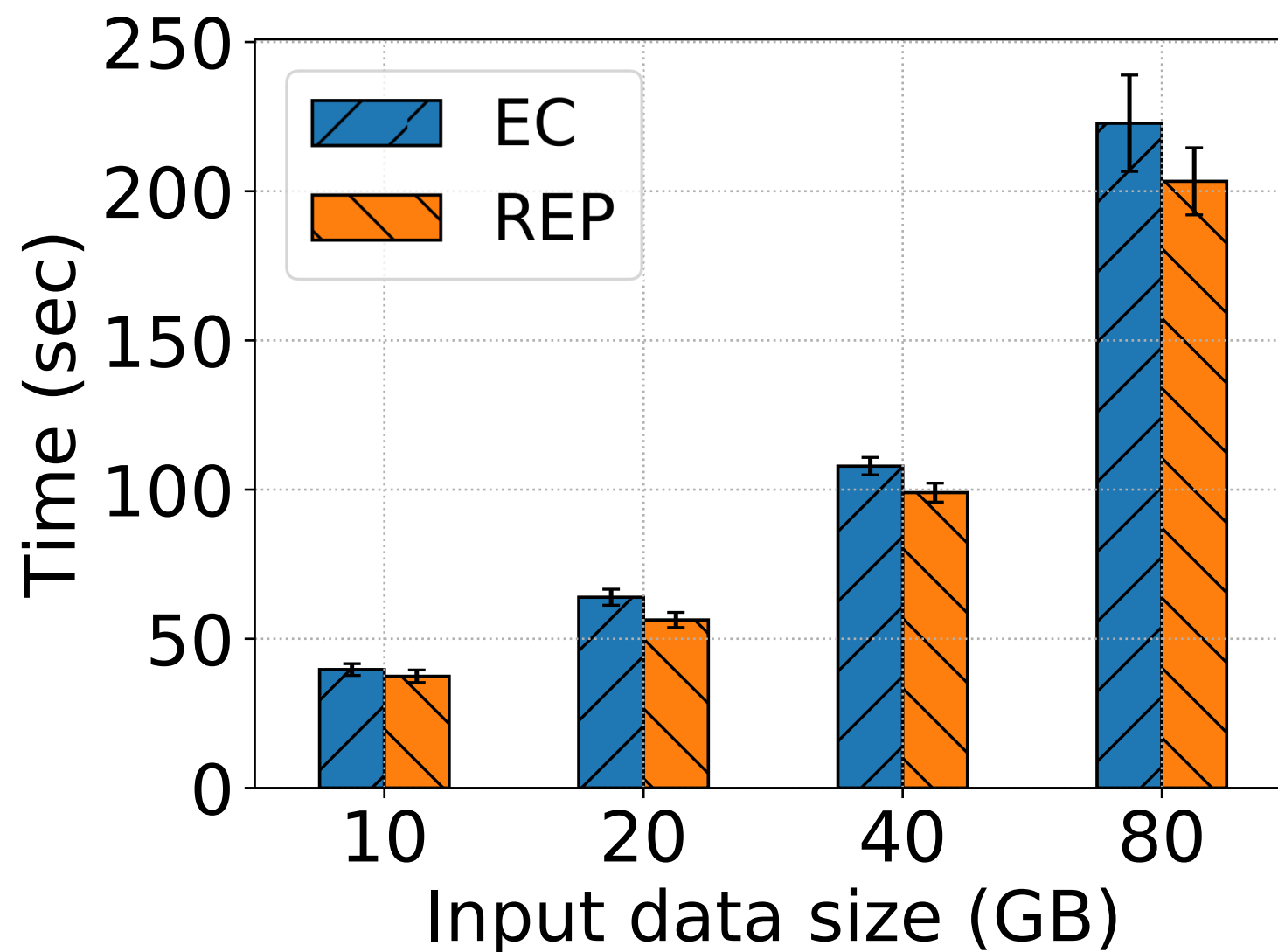
# Hadoop MapReduce



# Sort

## Job execution time

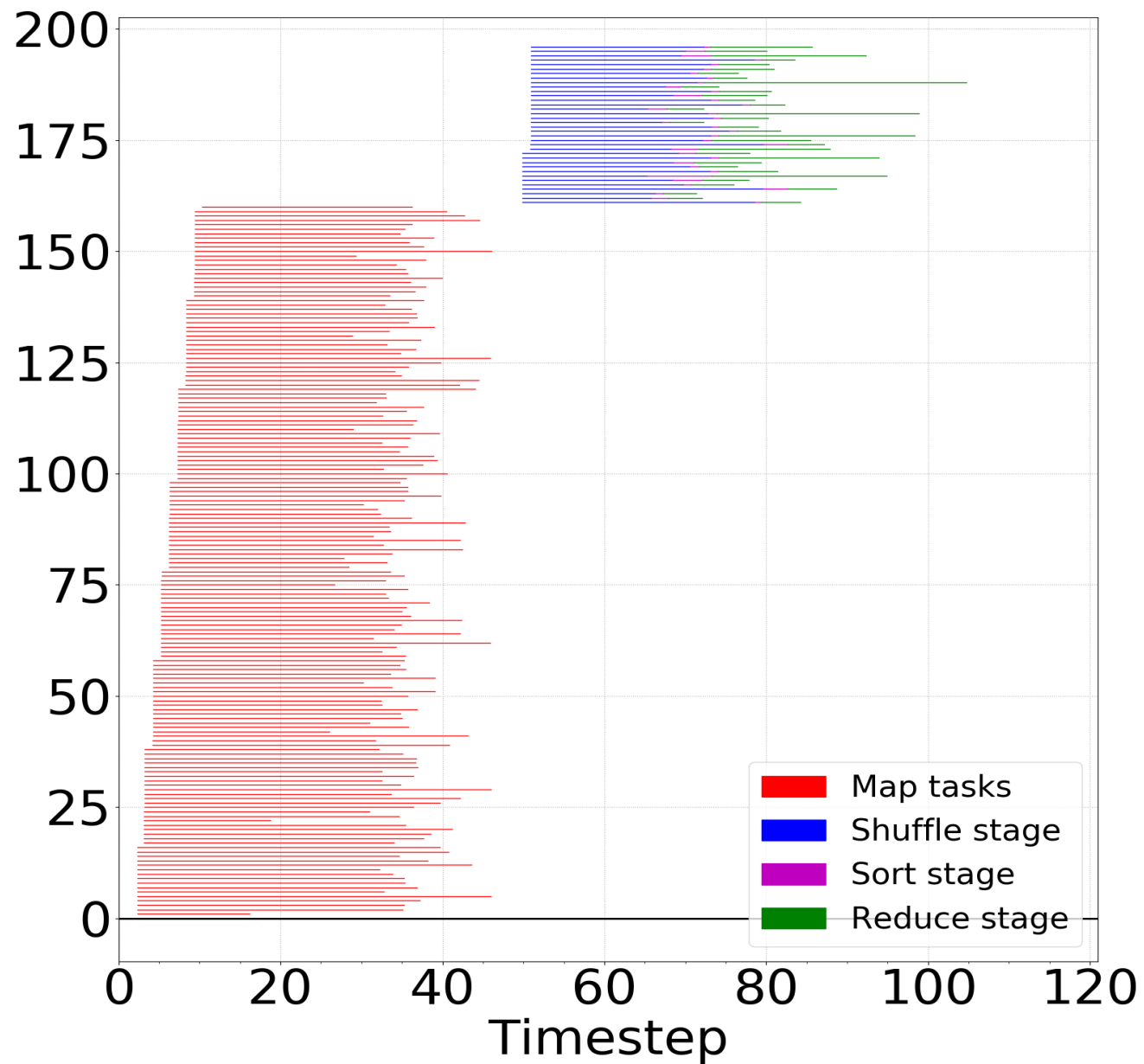
Non-overlapping  
Shuffle,  
HDD,  
10 Gbps



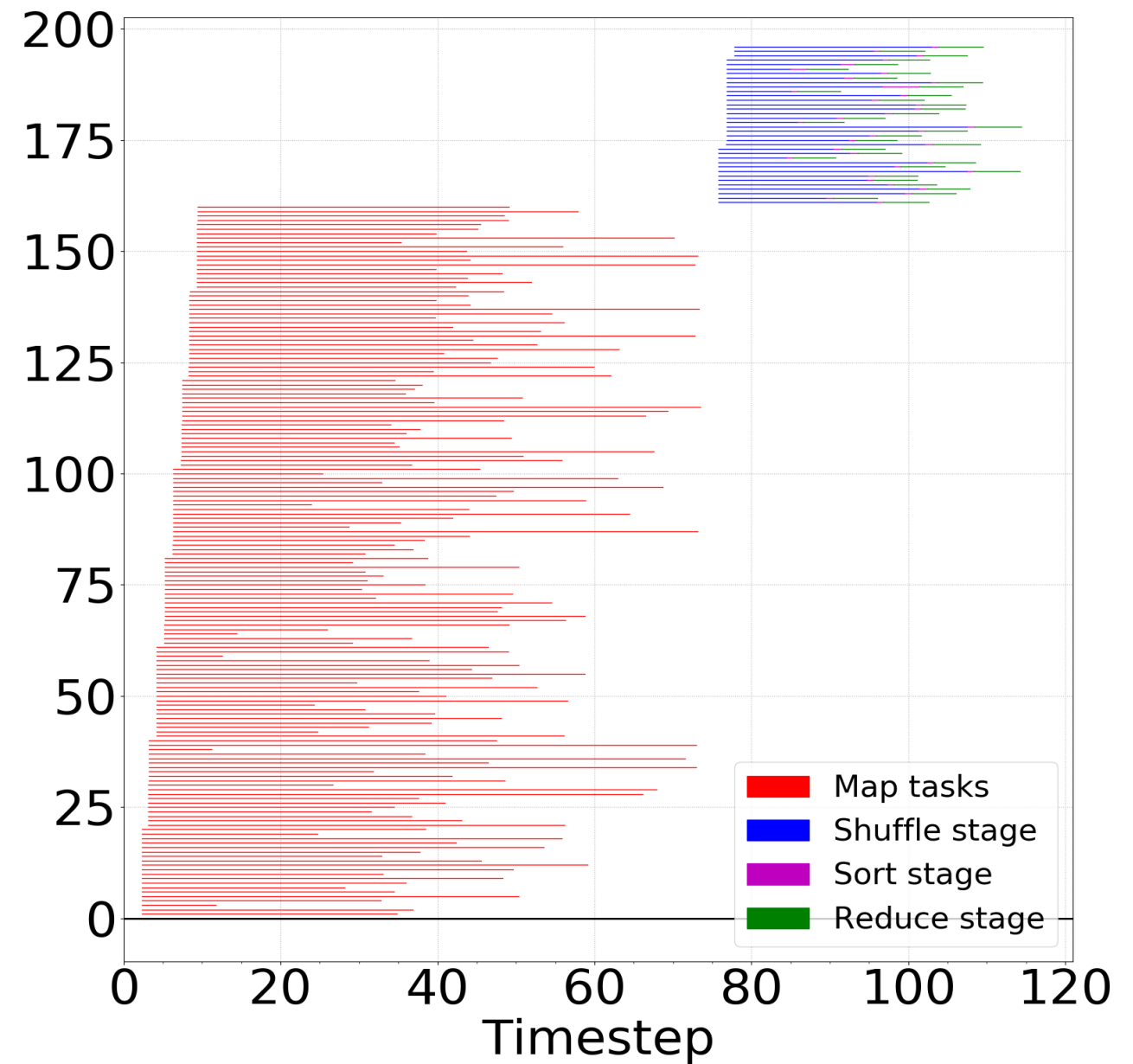


# Sort

## Tasks timeline - 40 GB



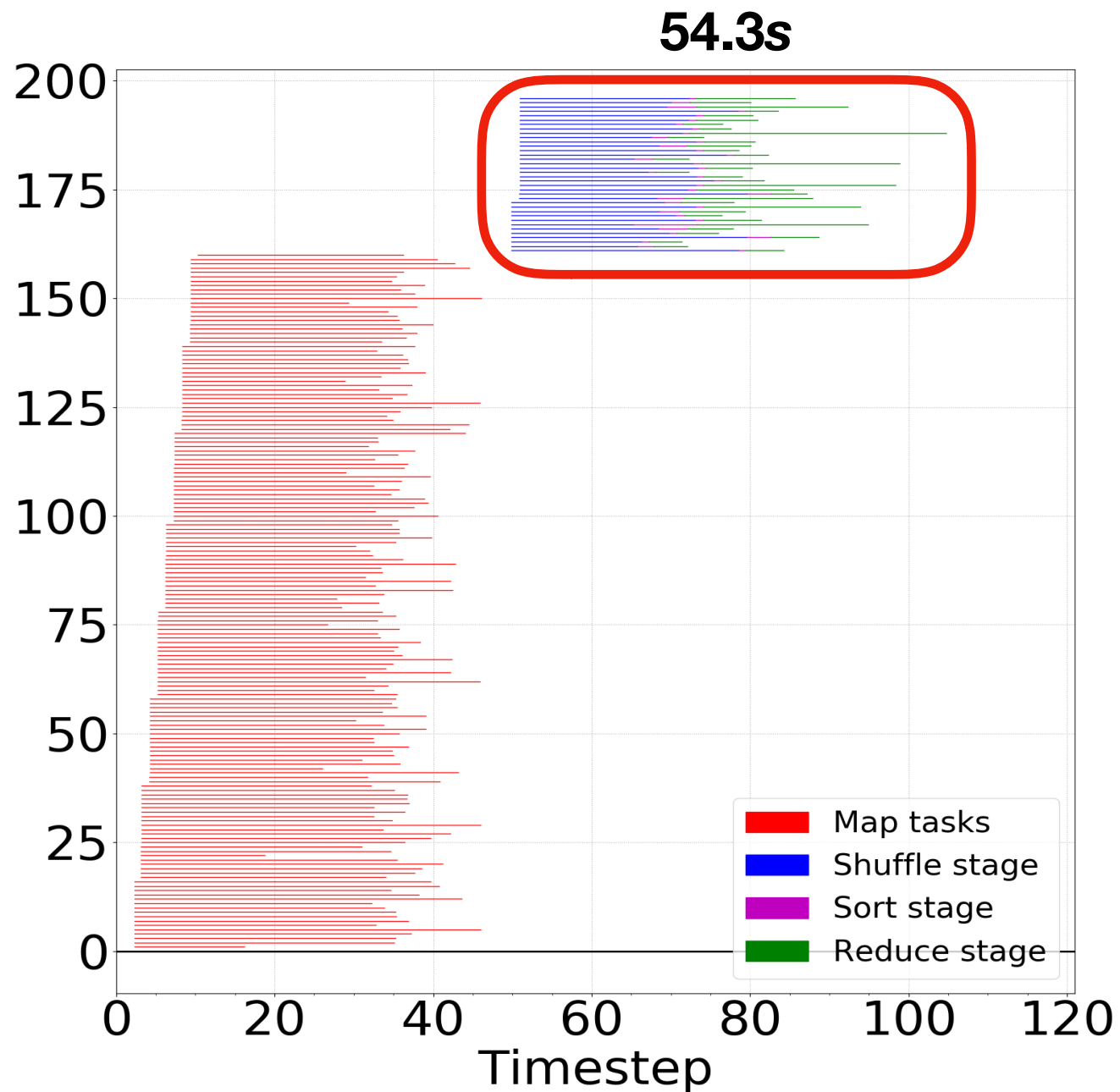
**REP**



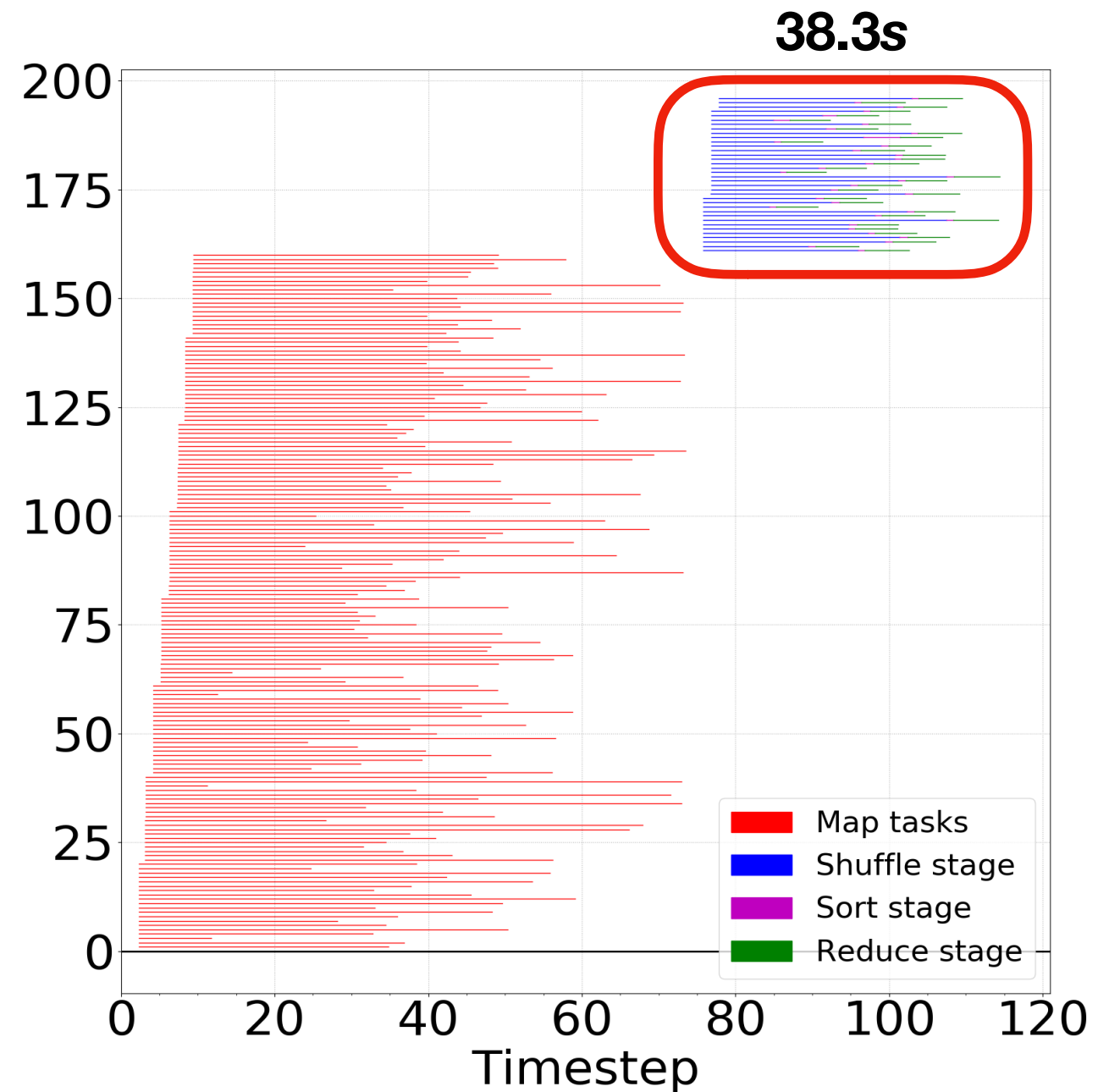
**EC**

# Sort

## Tasks timeline - 40 GB



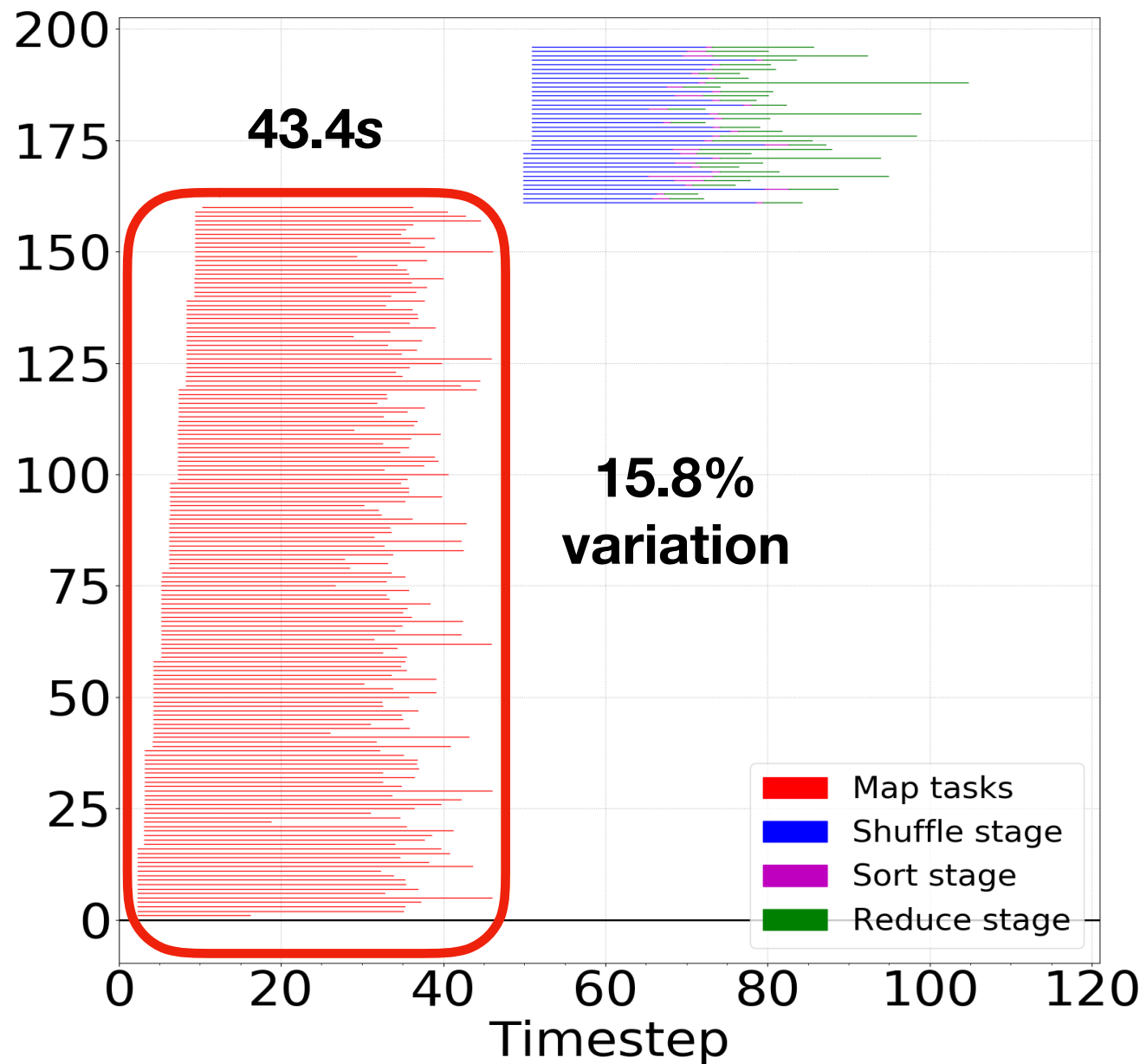
**REP**



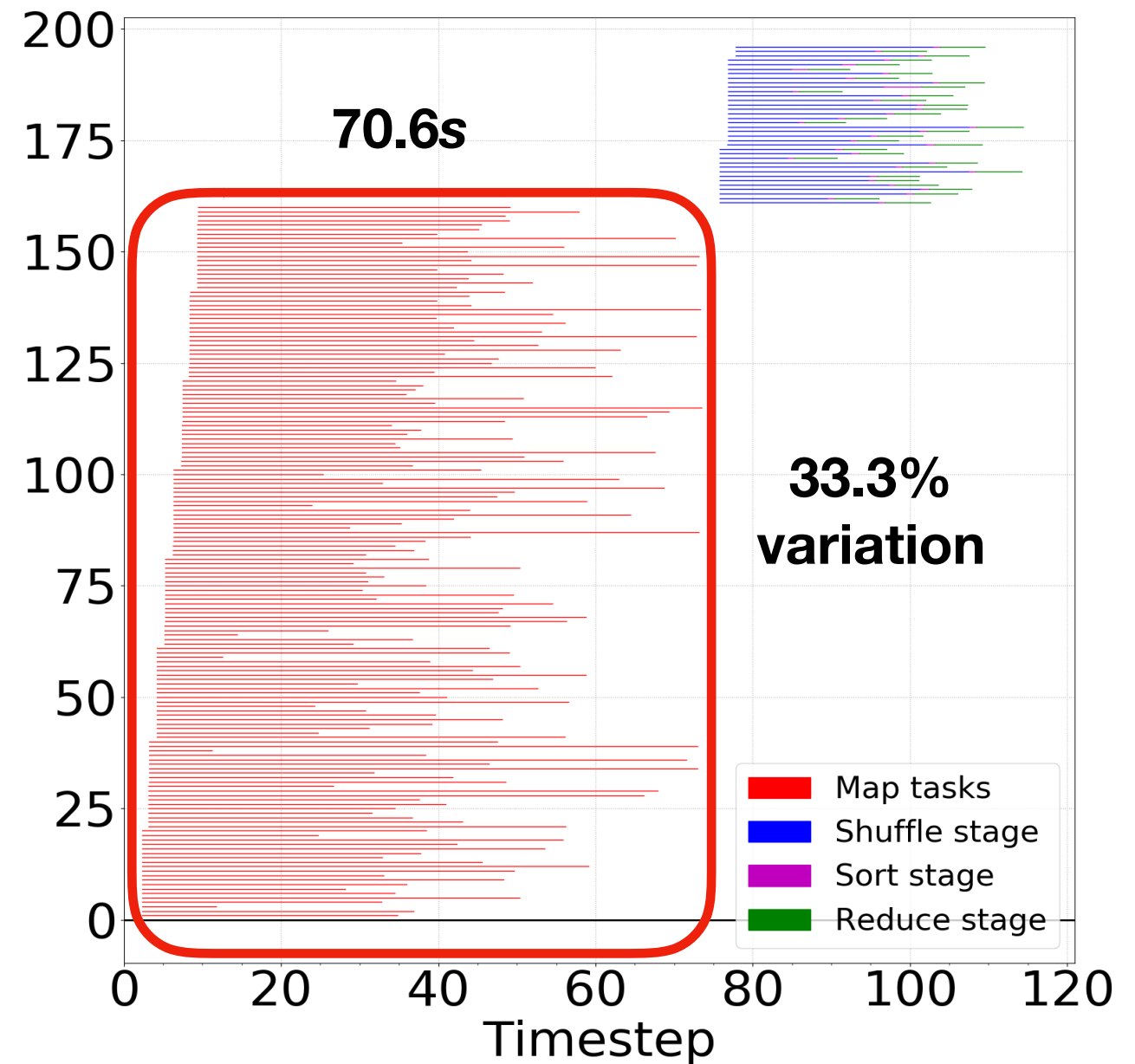
**EC**

# Sort

## Tasks timeline - 40 GB



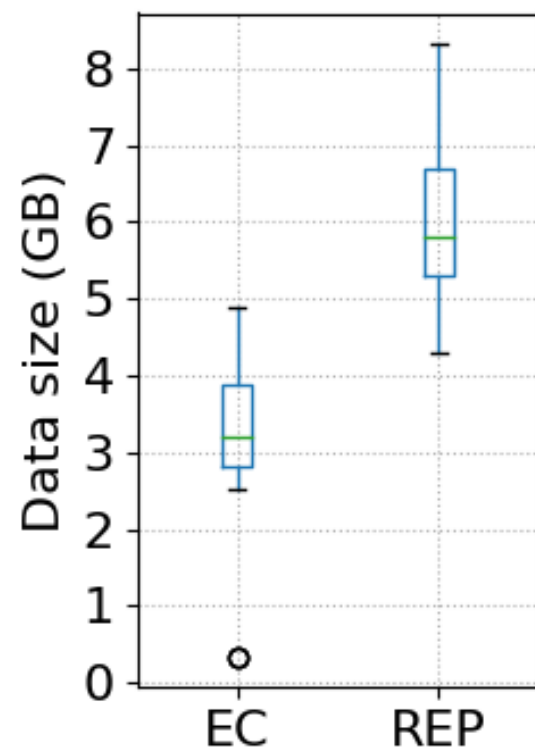
**REP**



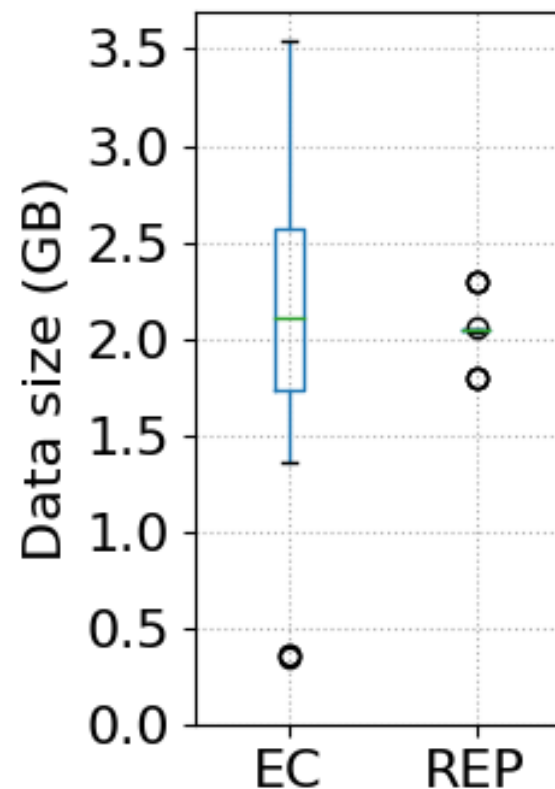
**EC**

# Sort

## Data read skew



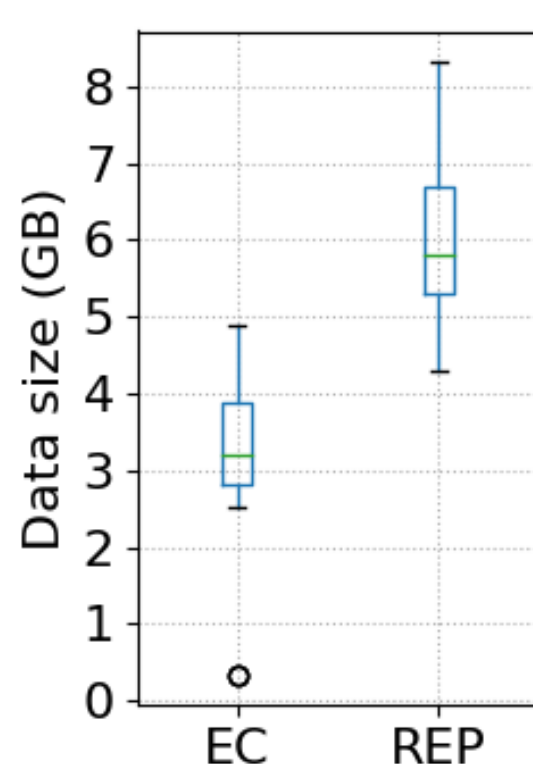
**Block distribution  
in HDFS**



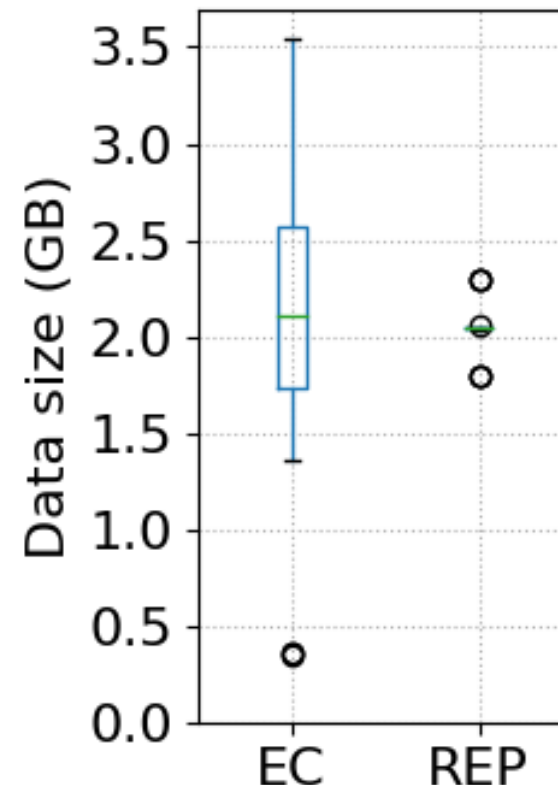
**Data read  
per machine**

# Sort

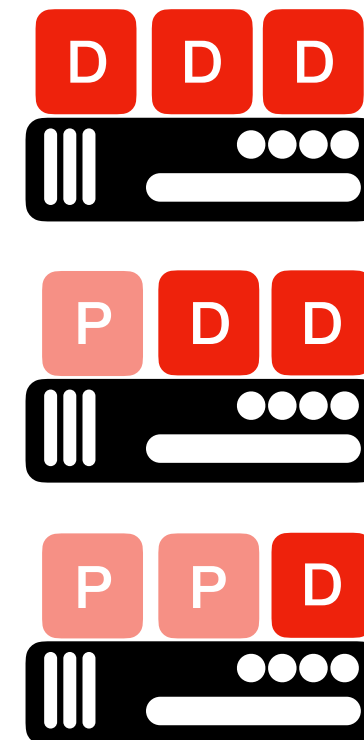
## Data read skew



**Block distribution  
in HDFS**



**Data read  
per machine**



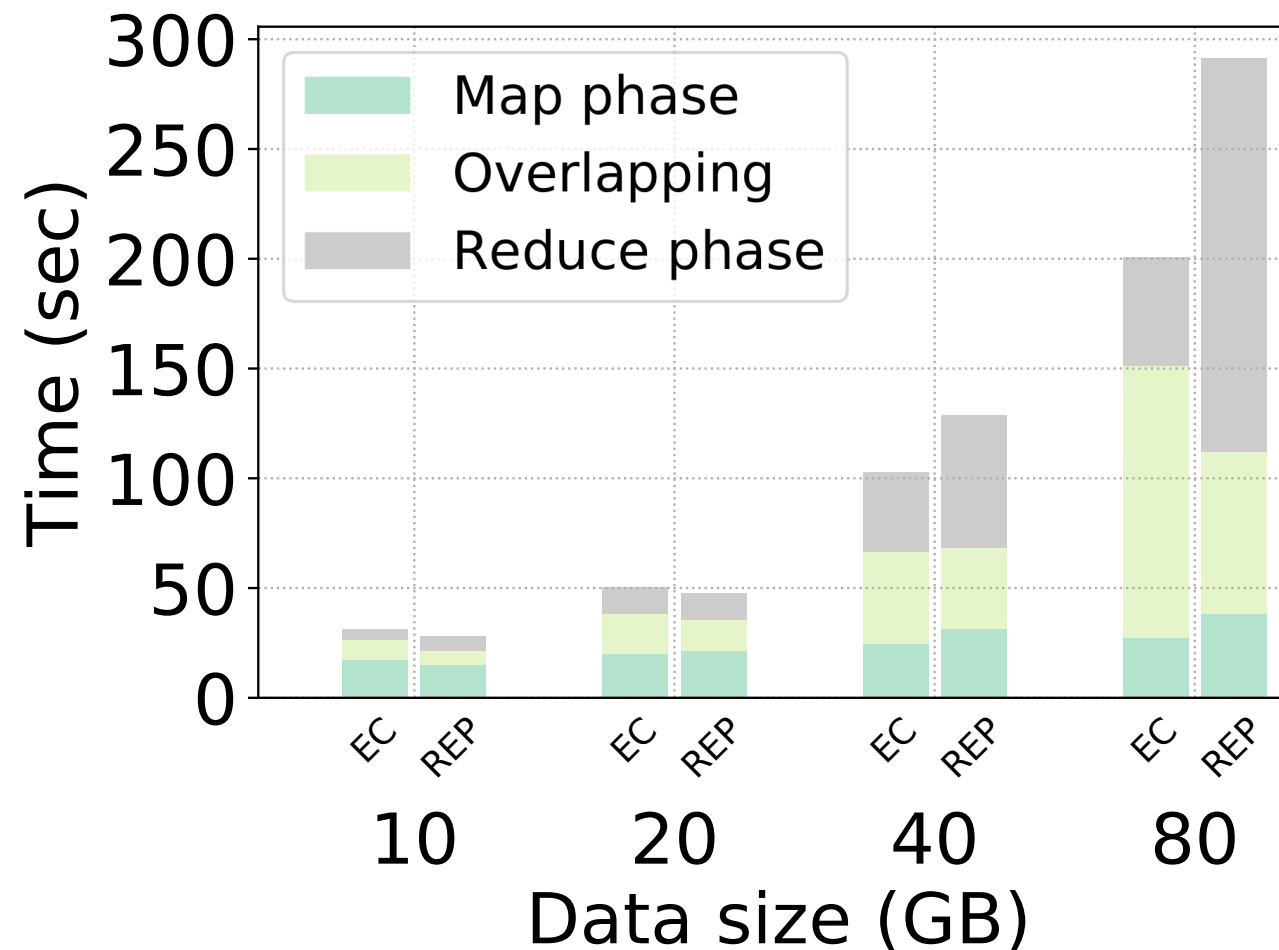
**Data/Parity**

Though they have different functionalities, original and parity chunks are treated the same when distributed across DN's. This results in a high variation in the data reads amongst the nodes.

# Sort

## Job execution time

Overlapping  
Shuffle,  
HDD,  
10 Gbps

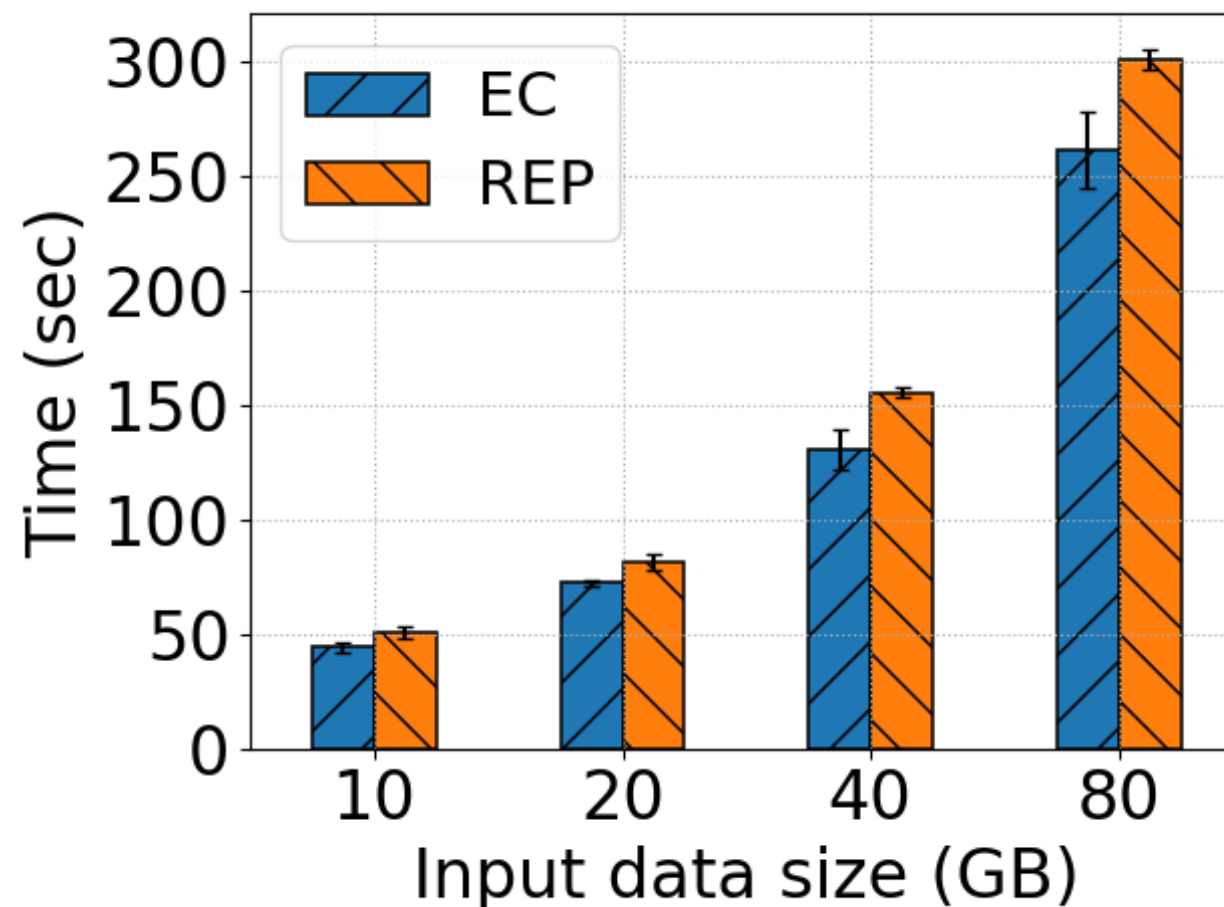


REP is more impacted by the overlapping shuffle as the contention increases on the disk level because of the simultaneous data read and write.

# Sort - Disk persistence

Job execution time

Non-overlapping  
Shuffle,  
HDD,  
10 Gbps

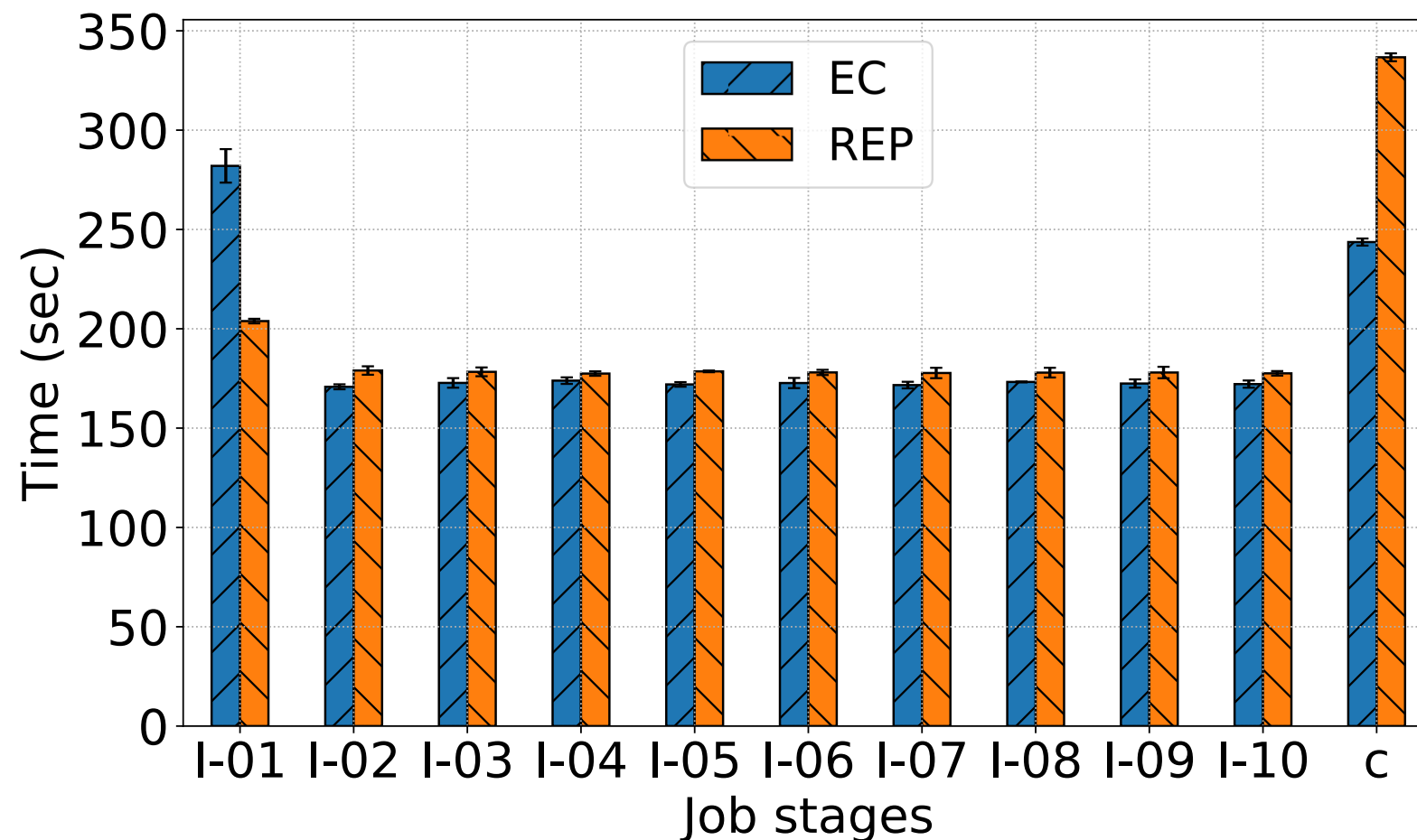


When output data are completely persisted to disk, jobs under EC are clearly faster than those under REP, at least during the reduce stage.

# K-means

## Job execution time

Overlapping  
Shuffle,  
HDD,  
10 Gbps



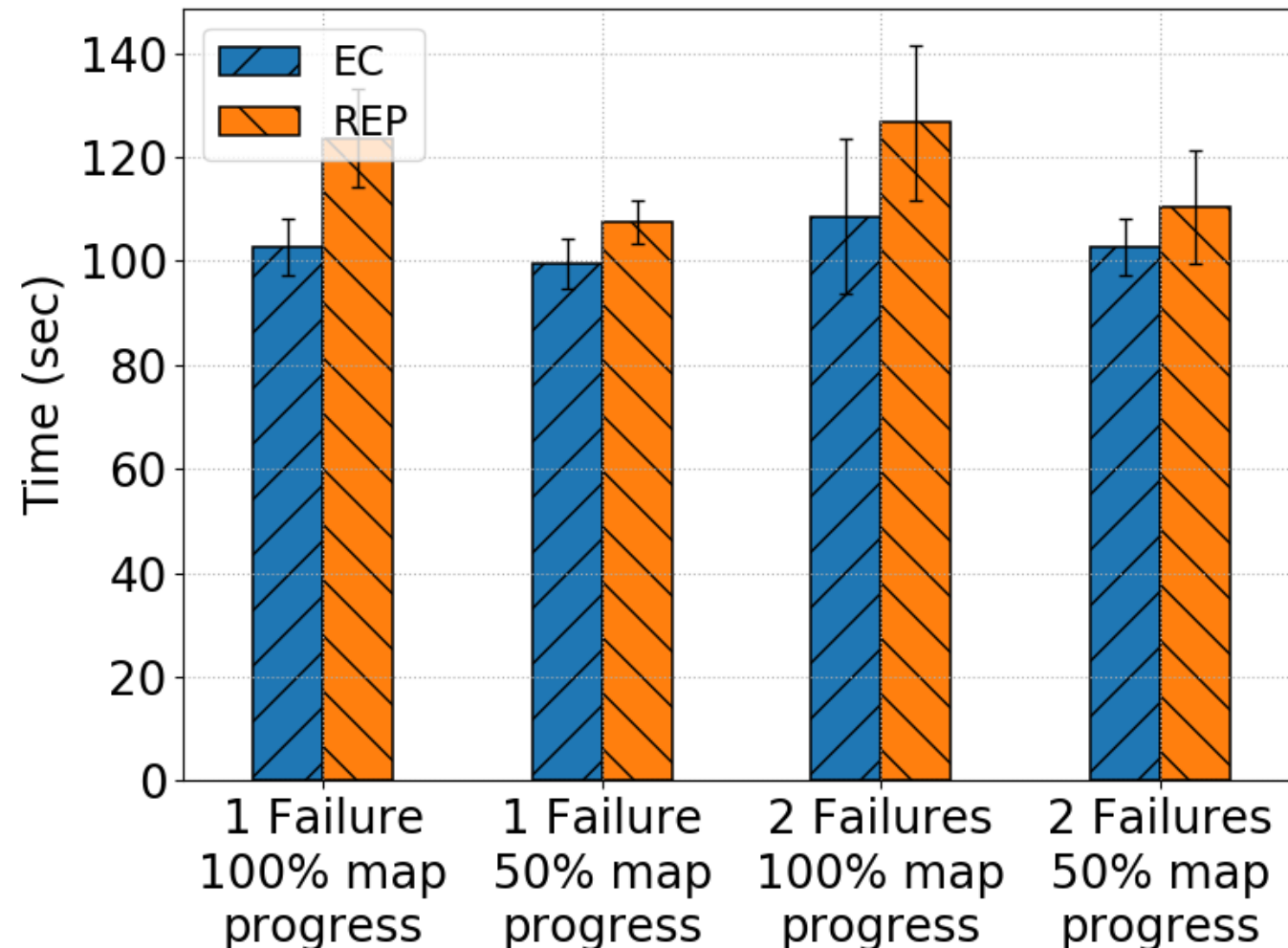
Iterative applications have similar performance under EC and REP as caching input data after the first iteration will move the bottleneck to the CPU for subsequent iterations.



# Sort - under failures

Job execution time - 40 GB

Non-overlapping  
Shuffle,  
HDD,  
10 Gbps



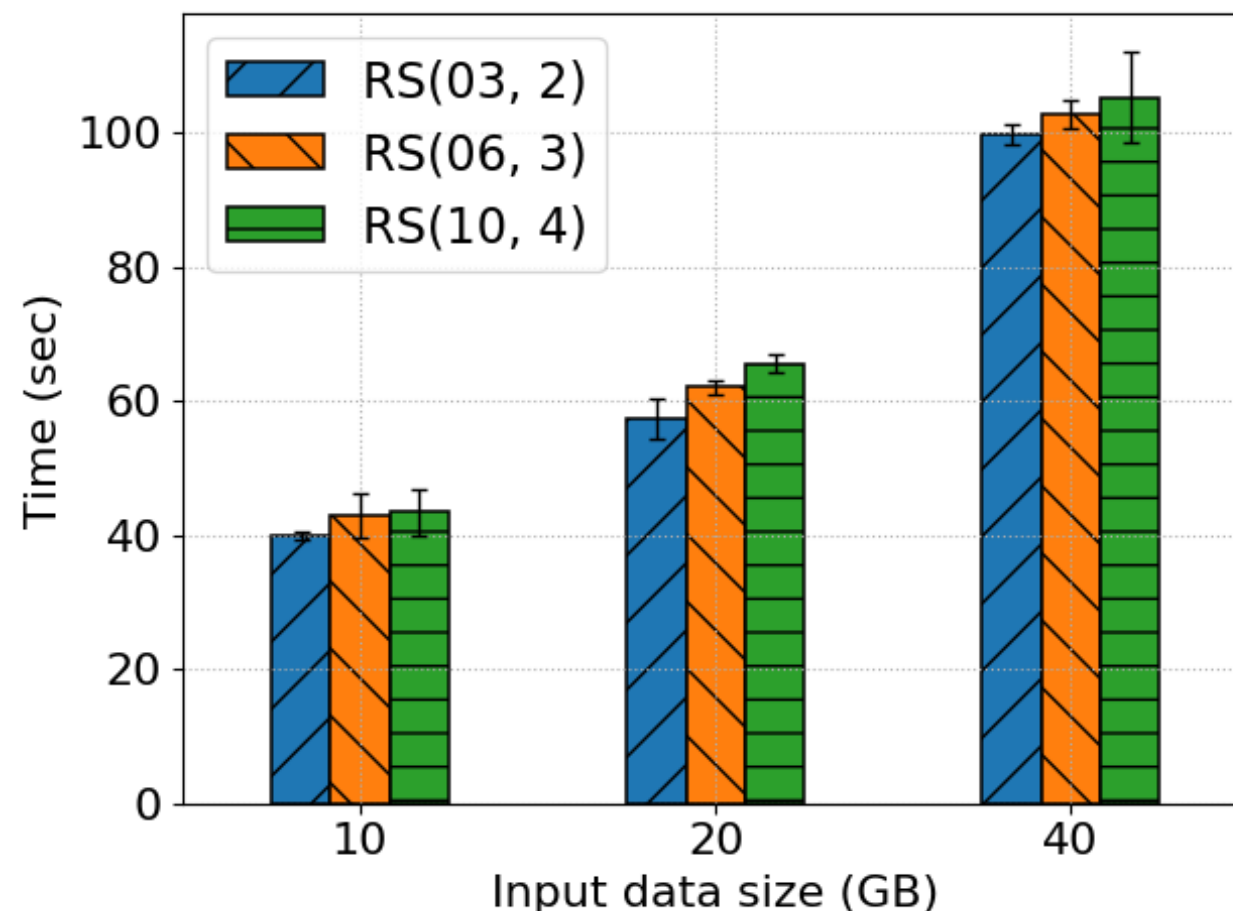
Degraded reads under EC with striped layout introduces negligible overhead (unlike contiguous layout<sup>1</sup>) and therefore the performance under EC is comparable to that under REP.

<sup>1</sup> Li et al., Degraded-First Scheduling for MapReduce in Erasure-Coded Storage Clusters, DSN, 2014

# Sort - RS scheme

## Job execution time

Non-overlapping  
Shuffle,  
HDD,  
10 Gbps

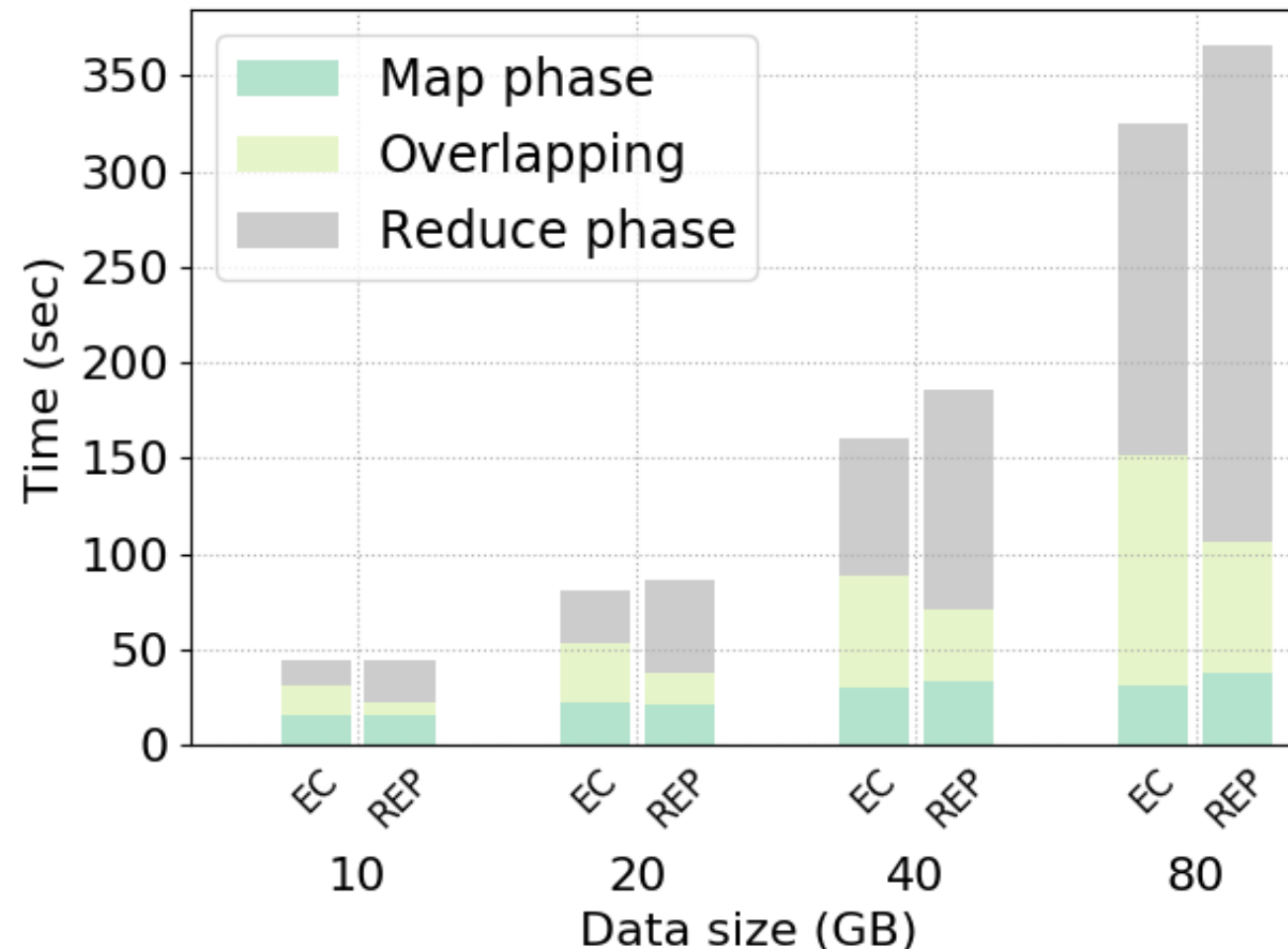


While increasing the stripe size can improve failure resiliency, it reduces local data accesses (map inputs) and increases the probability of data read imbalance.

# Sort - with slow network

## Job execution time

Overlapping  
Shuffle,  
HDD,  
1 Gbps

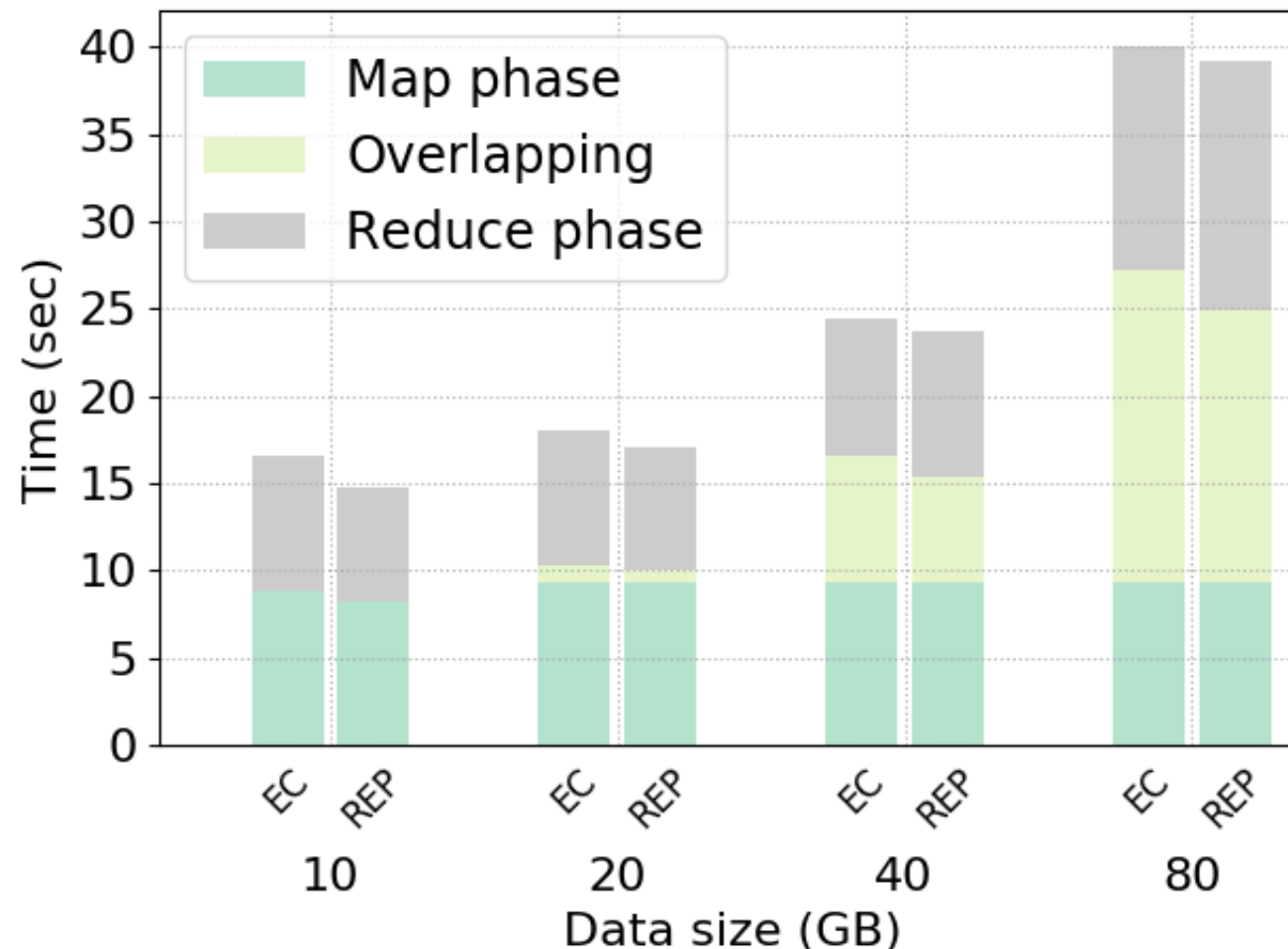


Reading the input data under EC is slightly affected when the network bandwidth is reduced. However, EC bring considerable advantage when the output size is big.

# Sort - in memory

## Job execution time

Overlapping  
Shuffle,  
MEM,  
10 Gbps



Using high-speed storage devices eliminate the stragglers caused by disk contention, therefore, EC brings the same performance as replication.

# Conclusion & Future work

- Experimental evaluation of data-intensive application under *Erasure Coding* with striped block layout
  - EC is a potential alternative to replication for data processing.
  - The current implementation of EC in HDFS can cause struggles that impact the job performances.
- As future work
  - Study how EC-aware block distribution can impact of jobs execution time.
  - Study how EC-aware scheduling can reduce the read skew between the data nodes.

# *Thank you !*

**Jad Darrous**

**Ph.D. candidate**

**STACK/AVALON team**

**Inria research institute**

**[jad.darrous@inria.fr](mailto:jad.darrous@inria.fr)**

**<http://perso.ens-lyon.fr/jad.darrous/>**

