

(Hadoop) MapReduce

DS 5110/CS 5501: Big Data Systems

Spring 2024

Lecture 4b

Yue Cheng



Some material taken/derived from:

- Wisconsin CS 320 by Tyler Caraza-Harter.

@ 2024 released for use under a [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.

HDFS demo...

Learning objectives

- Describe the role mappers and reducers have in MapReduce jobs
- Understand how MapReduce interacts with GFS (HDFS)

MapReduce

Today

(Hadoop)
MapReduce

BigTable (HBase)

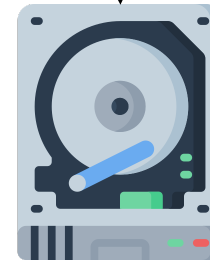
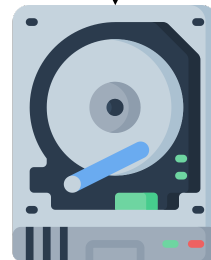
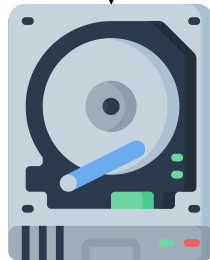
GFS (HDFS)

Worker
machine

Worker
machine

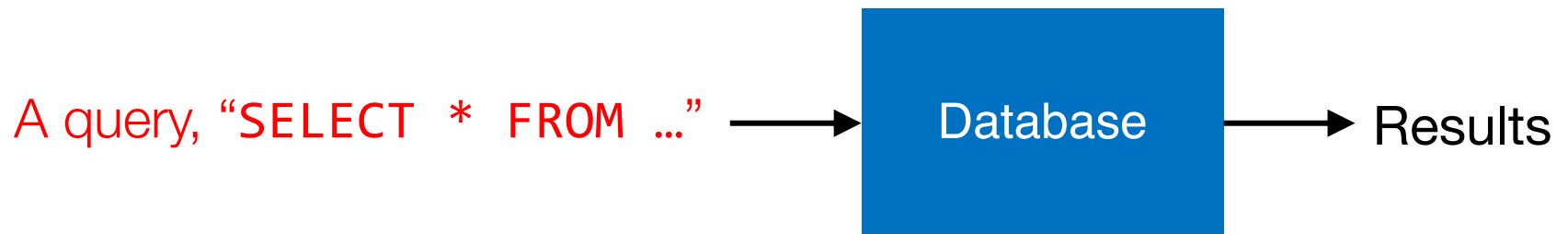
Worker
machine

Worker
machine

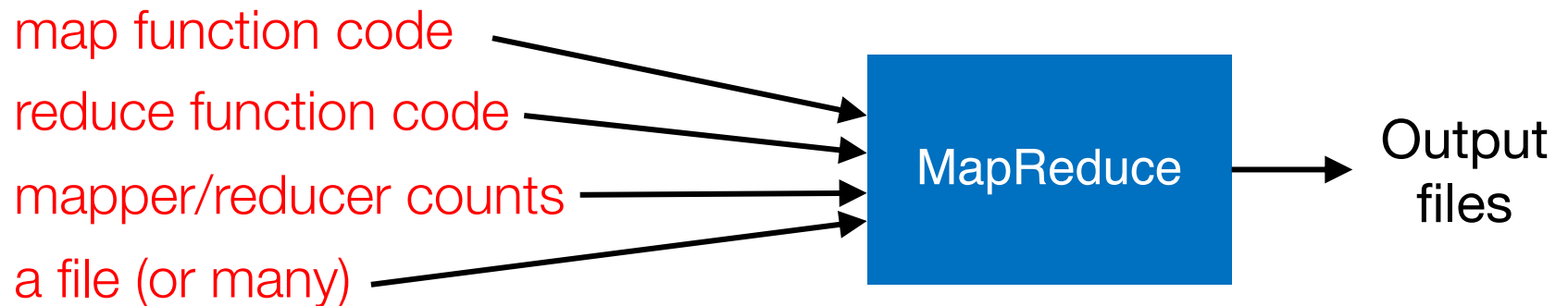


How does big data system answer questions?

SQL

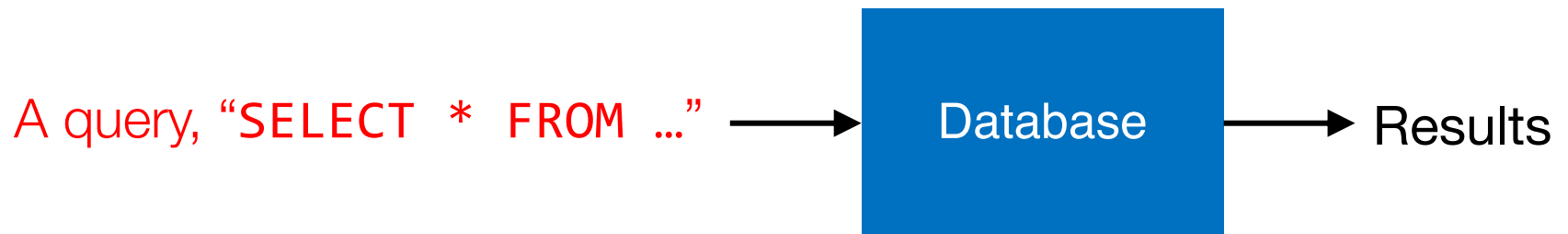


MapReduce

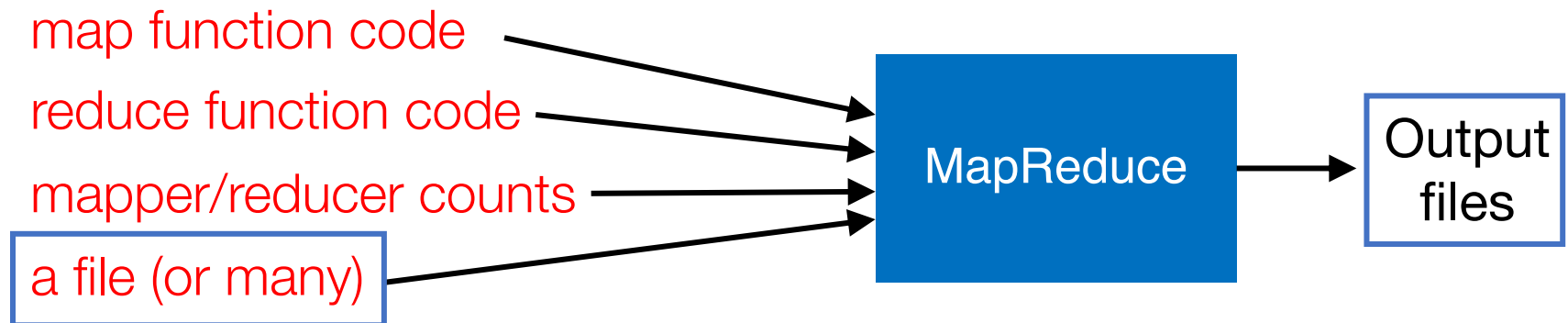


How does big data system answer questions?

SQL



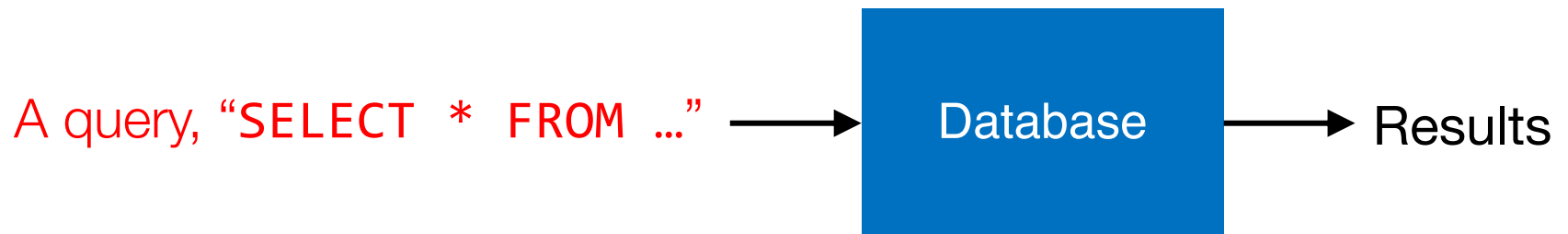
MapReduce



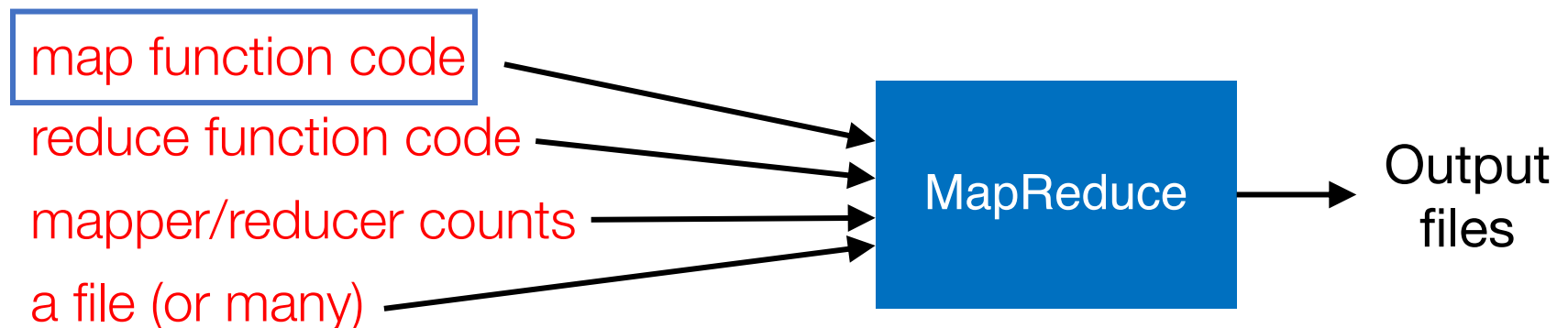
Input/output files are generally stored in HDFS

How does big data system answer questions?

SQL



MapReduce



Mappers by example: What are the colors of the squares?

input.csv (in HDFS)

color,	shape,	size
red,	circle,	3
red,	square,	5
blue,	oval,	1
green,	square,	3

```
def map(key, value):  
    ...
```

In SQL:

SELECT color FROM table WHERE shape = “square”

Mappers by example: What are the colors of the squares?

input.csv (in HDFS)

color,	shape,	size
red,	circle,	3
red,	square,	5
blue,	oval,	1
green,	square,	3

0

red,circle,3

```
def map(key, value):  
    ...
```

zero or more output
key/value pairs

Mappers by example: What are the colors of the squares?

input.csv (in HDFS)

color,	shape,	size
red,	circle,	3
red,	square,	5
blue,	oval,	1
green,	square,	3

1

red, square, 5

```
def map(key, value):  
    ...
```

zero or more output
key/value pairs

Mappers by example: What are the colors of the squares?

input.csv (in HDFS)

color	shape	size
red	circle	3
red	square	5
blue	oval	1
green	square	3

1 red, square, 5

```
def map(key, value):  
    if value.shape = square:  
        emit(key, value.color)
```

key value
1 red
3 green

Mappers by example: What are the colors of the squares?

input.csv (in HDFS)

color	shape	size
red	circle	3
red	square	5
blue	oval	1
green	square	3

What if the data is huge?

```
def map(key, value):  
    if value.shape = square:  
        emit(key, value.color)
```

1

red, square, 5

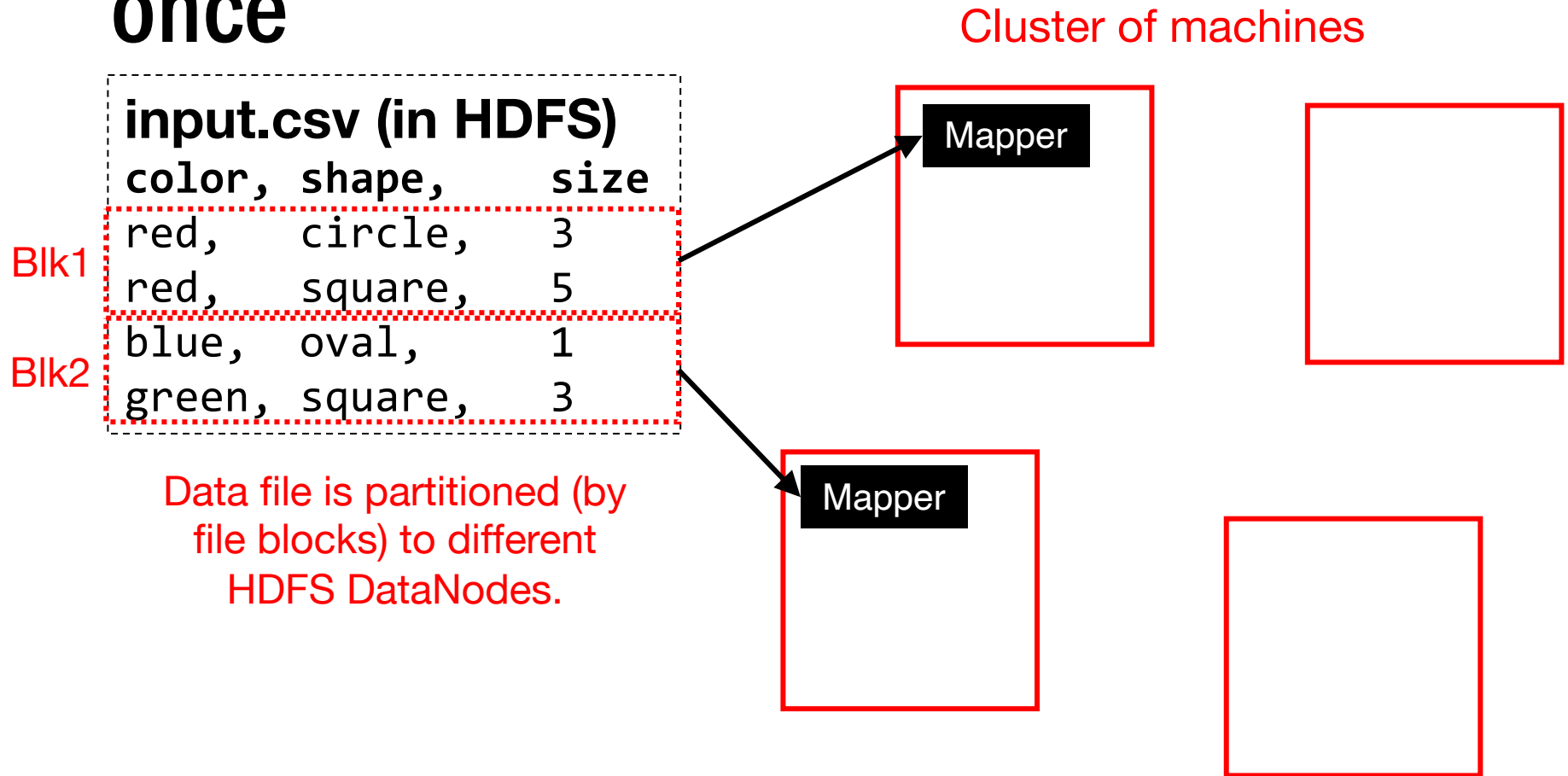
key

1
3

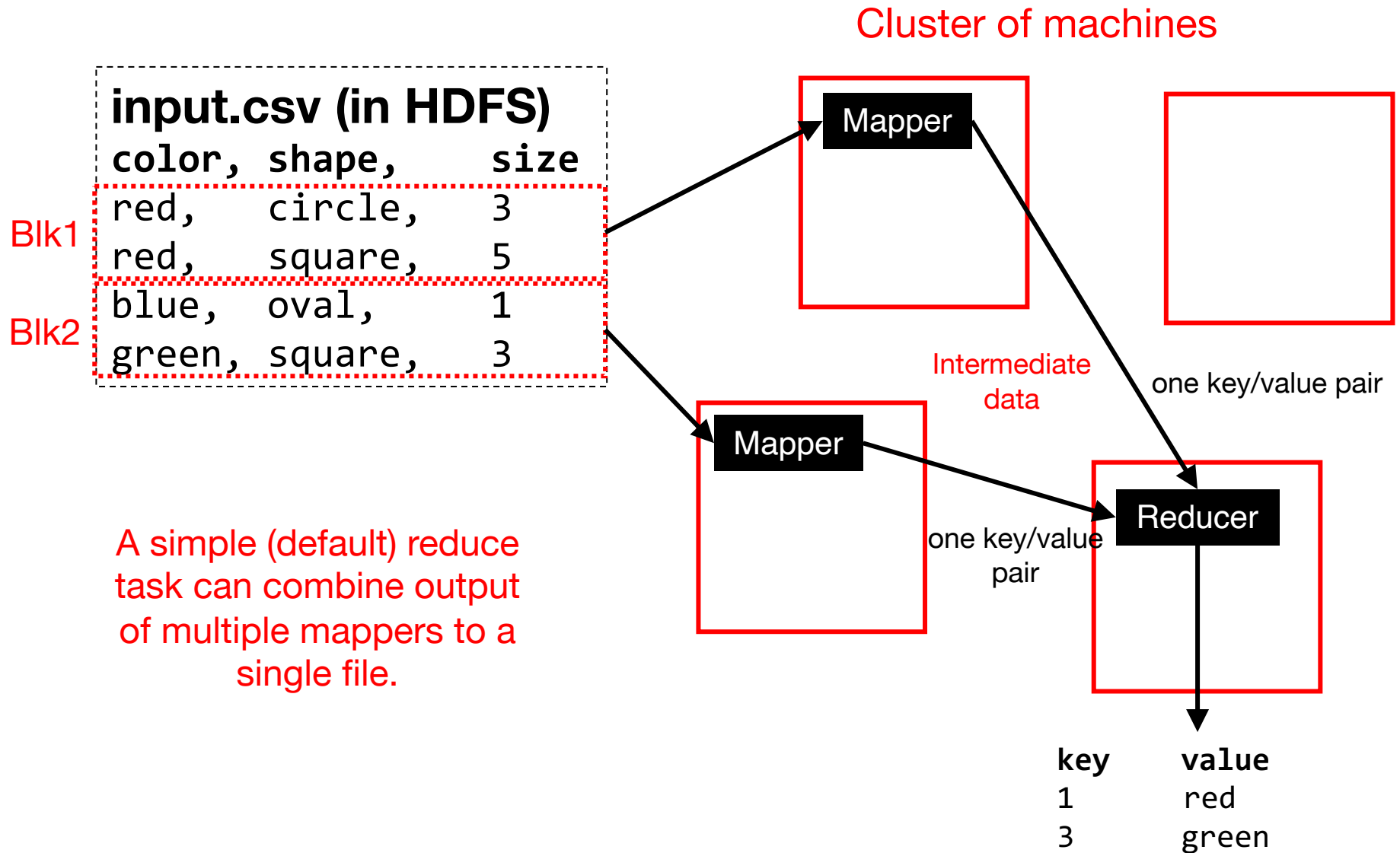
value

red
green

Mappers run on multiple machines at once



Reducers



Reducers

Reducers can output exactly their input,
OR have further computation.

```
def reduce(key, values):  
    for row in values:  
        emit(key, row)
```

Reducers

input.csv (in HDFS)

color	shape	size
red	circle	3
red	square	5
blue	oval	1
green	square	3

```
def map(key, value):  
    emit(value.color, value)
```

key	value
blue	blue, oval, 1
green	green, square, 3
red	red, circle, 3
red	red, square, 5

blue	blue, oval, 1
green	green, square, 3
red	red, circle, 3
red	red, square, 5

```
def reduce(key, values):  
    count = 0  
    for row in values:  
        count = count + 1  
    emit(key, count)
```

Intermediate data is **grouped**
and **sorted** by key.

Reduce will be called 3 times (once
for each group). The call could
happen in one reduce task (or be
split over many).

Reducers

input.csv (in HDFS)

color	shape	size
red	circle	3
red	square	5
blue	oval	1
green	square	3

```
def map(key, value):  
    emit(value.color, value)
```

key	value
blue	blue, oval, 1
green	green, square, 3
red	red, circle, 3
red	red, square, 5

```
def reduce(key, values):  
    count = 0  
    for row in values:  
        count = count + 1  
    emit(key, count)
```

Intermediate data is **grouped**
and **sorted** by key.

key	value
blue	1

Reducers

input.csv (in HDFS)

color	shape	size
red	circle	3
red	square	5
blue	oval	1
green	square	3

```
def map(key, value):  
    emit(value.color, value)
```

key	value
-----	-------

blue	blue, oval, 1
green	green, square, 3
red	red, circle, 3
red	red, square, 5

```
def reduce(key, values):  
    count = 0  
    for row in values:  
        count = count + 1  
    emit(key, count)
```

Intermediate data is **grouped**
and **sorted** by key.

key	value
blue	1
green	1

Reducers

input.csv (in HDFS)

color	shape	size
red	circle	3
red	square	5
blue	oval	1
green	square	3

```
def map(key, value):  
    emit(value.color, value)
```

key	value
-----	-------

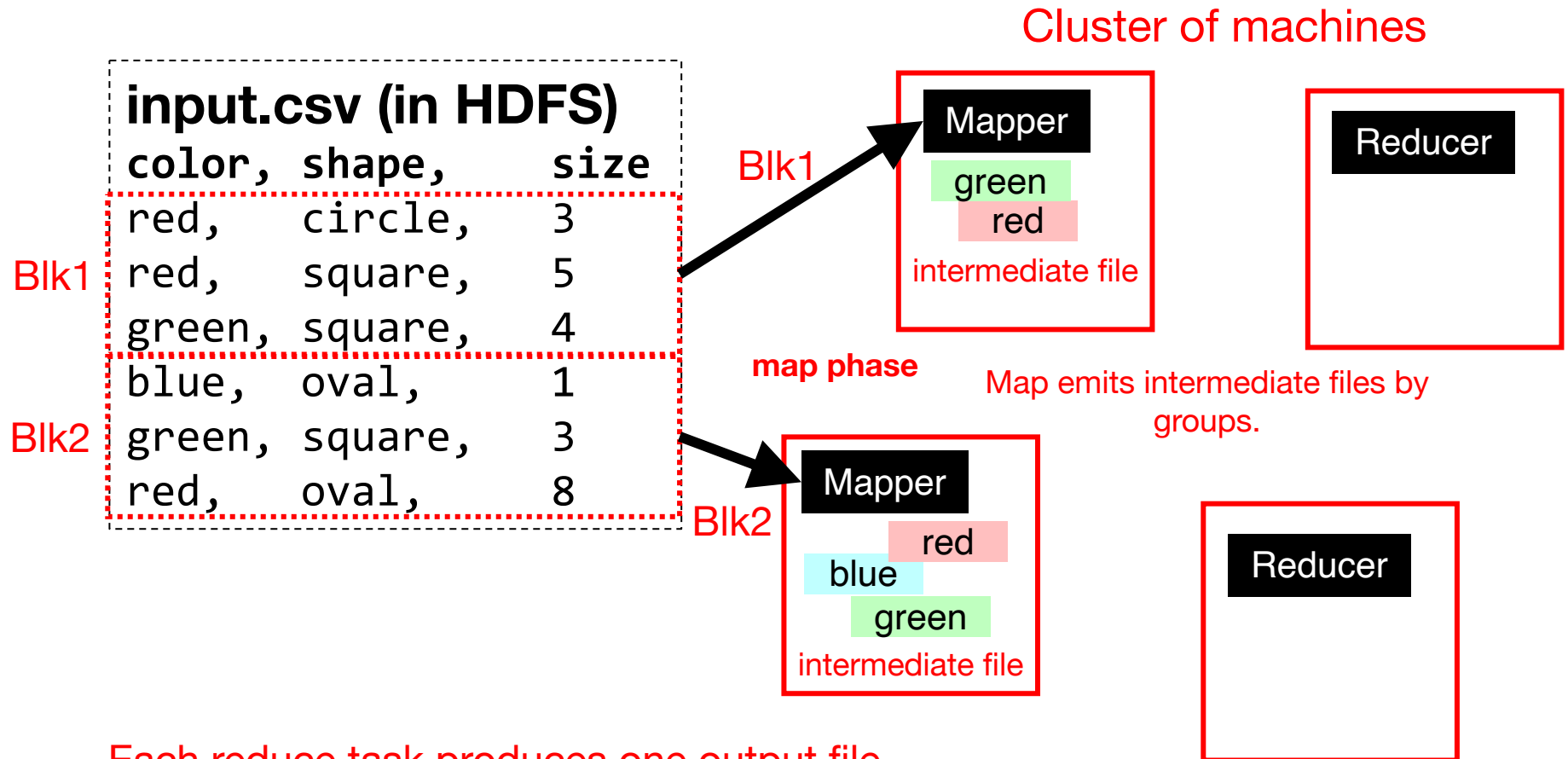
blue	blue, oval, 1
green	green, square, 3
red	red, circle, 3
red	red, square, 5

```
def reduce(key, values):  
    count = 0  
    for row in values:  
        count = count + 1  
    emit(key, count)
```

Intermediate data is **grouped**
and **sorted** by key.

key	value
blue	1
green	1
red	2

Multiple reducers (for big intermediate data)

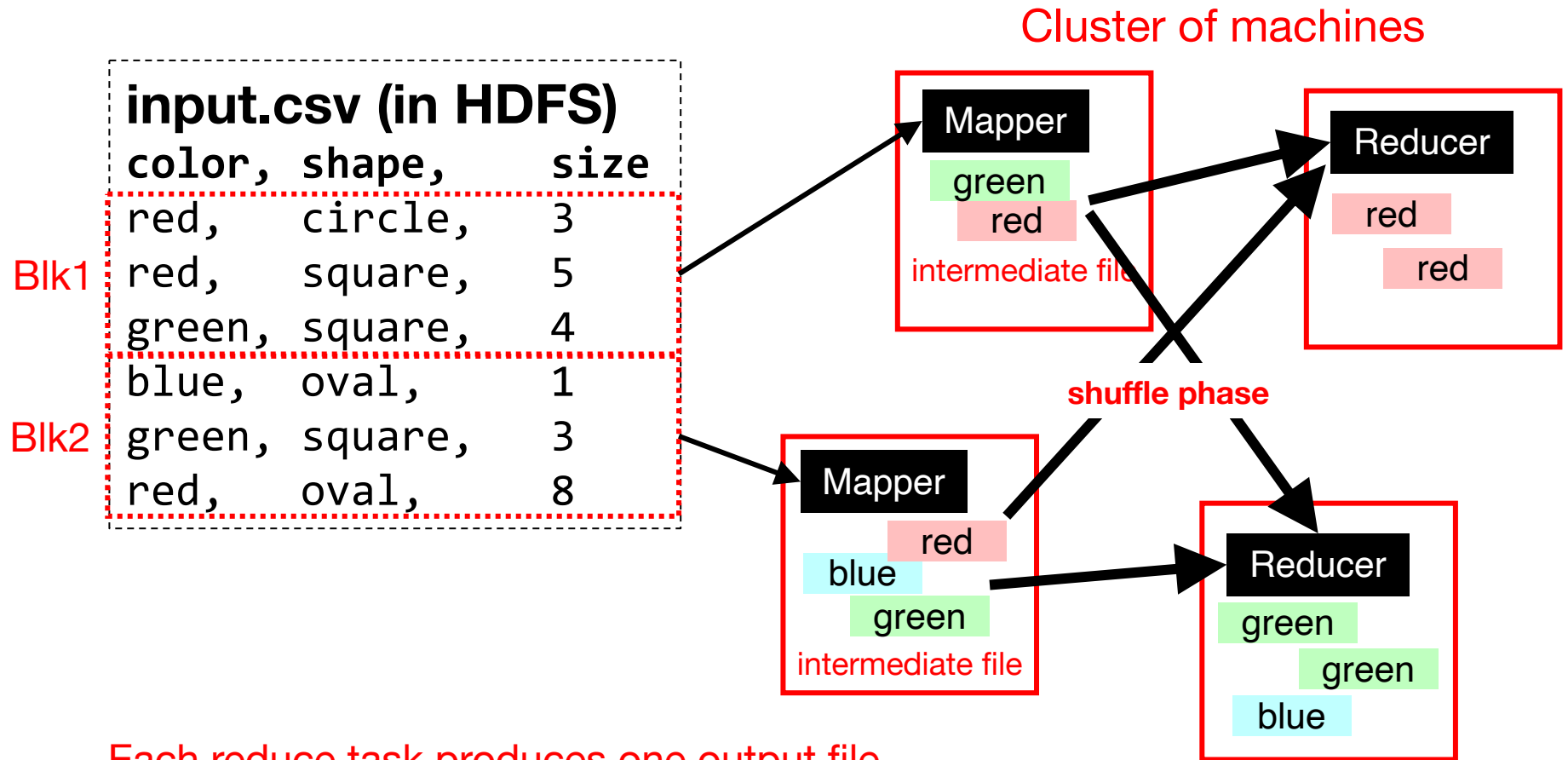


Each reduce task produces one output file.

A reduce task might take multiple keys.

All intermediate rows with the same key go to the same reducer.

Multiple reducers (for big intermediate data)



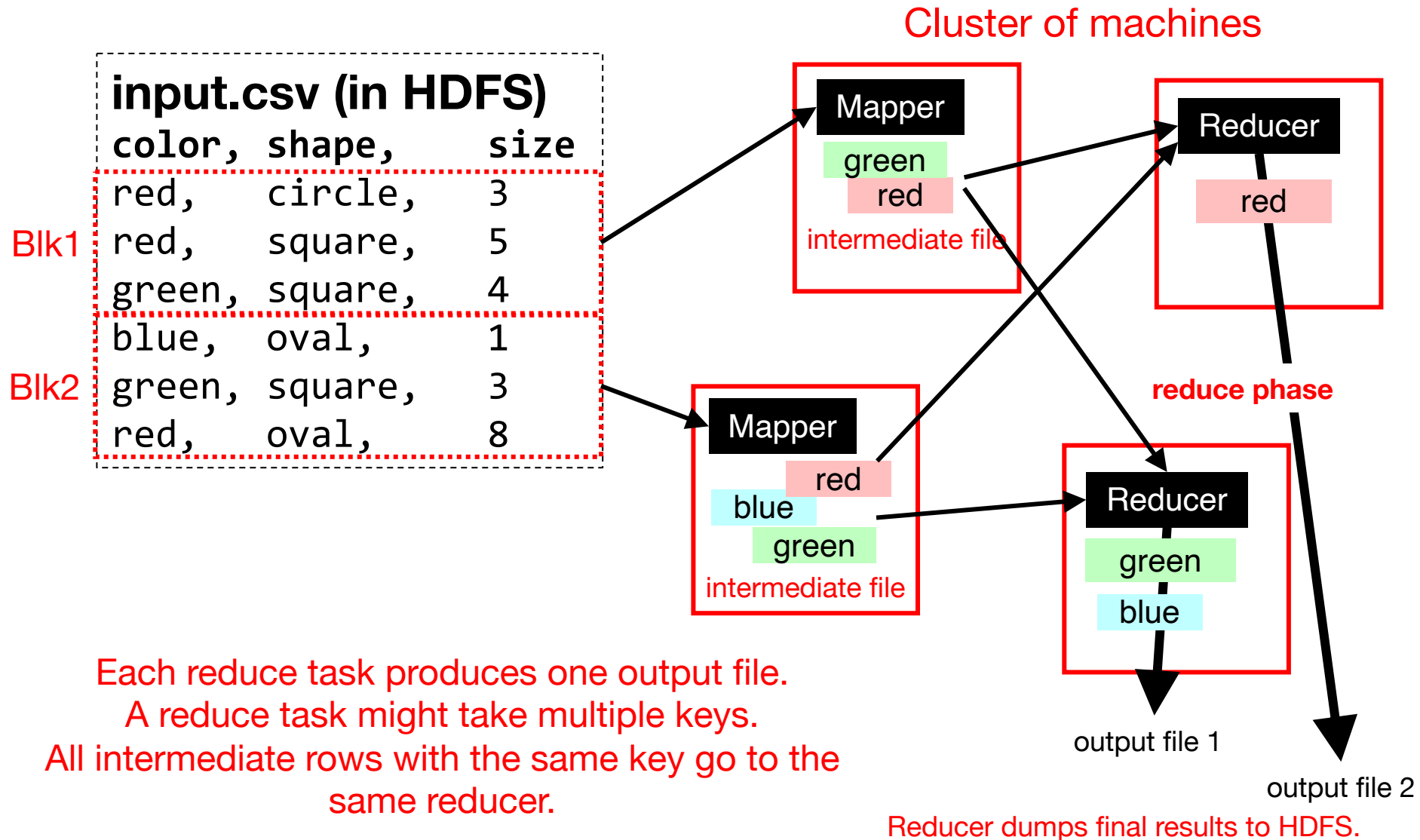
Each reduce task produces one output file.

A reduce task might take multiple keys.

All intermediate rows with the same key go to the same reducer.

Reducer collects all intermediate files of its assigned keys (groups).

Multiple reducers (for big intermediate data)

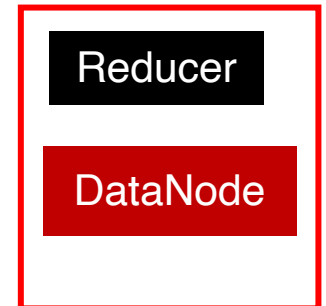
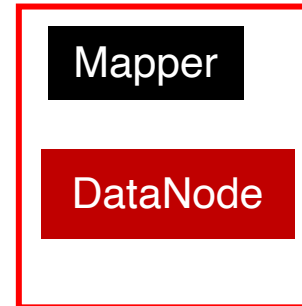


Data locality: Avoid network transfer

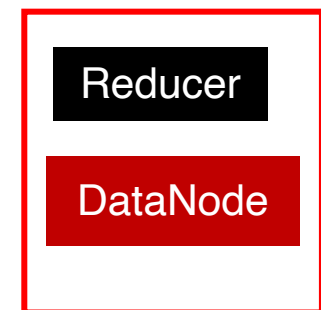
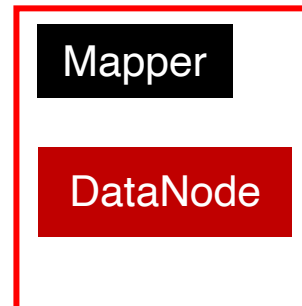
Cluster of machines

Run on same machines

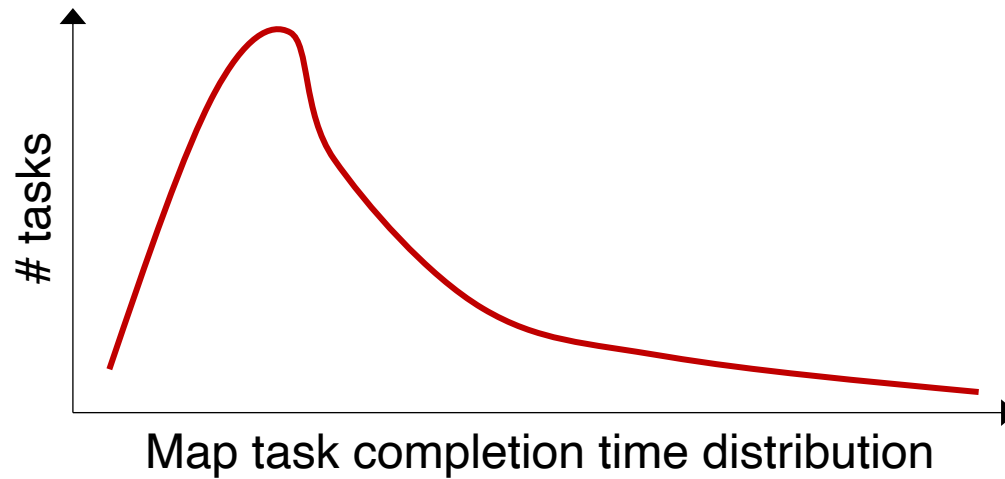
- Layered subsystems
- MapReduce executor
- HDFS DataNode



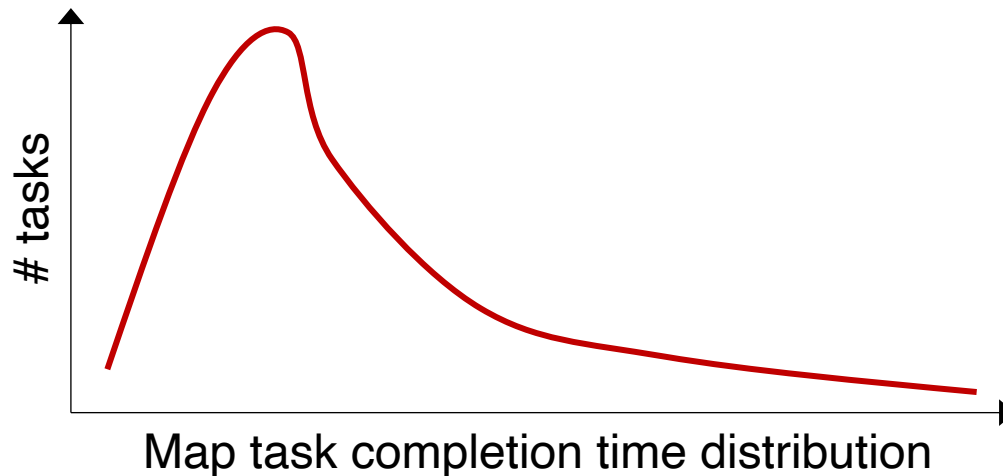
Try to run mappers on machine where DataNode has needed data. Uses local disk but not network.



Stragglers



Stragglers



- **Tail execution time** means some executors (always) finish late (**recall tail latency**)

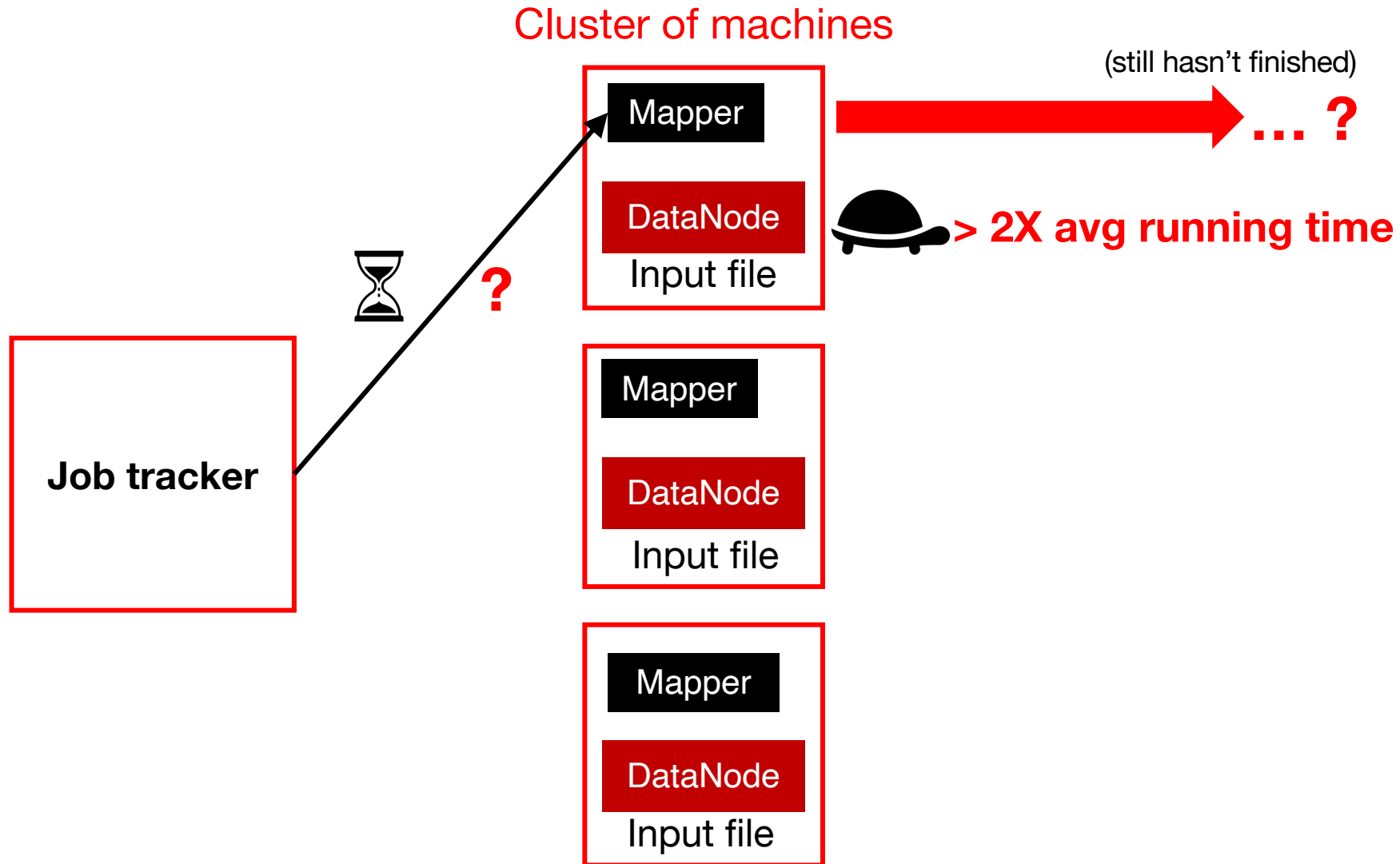
Q: How can MapReduce work around this?

- Hint: its approach to **fault-tolerance** provides the right tool

Resilience against stragglers?

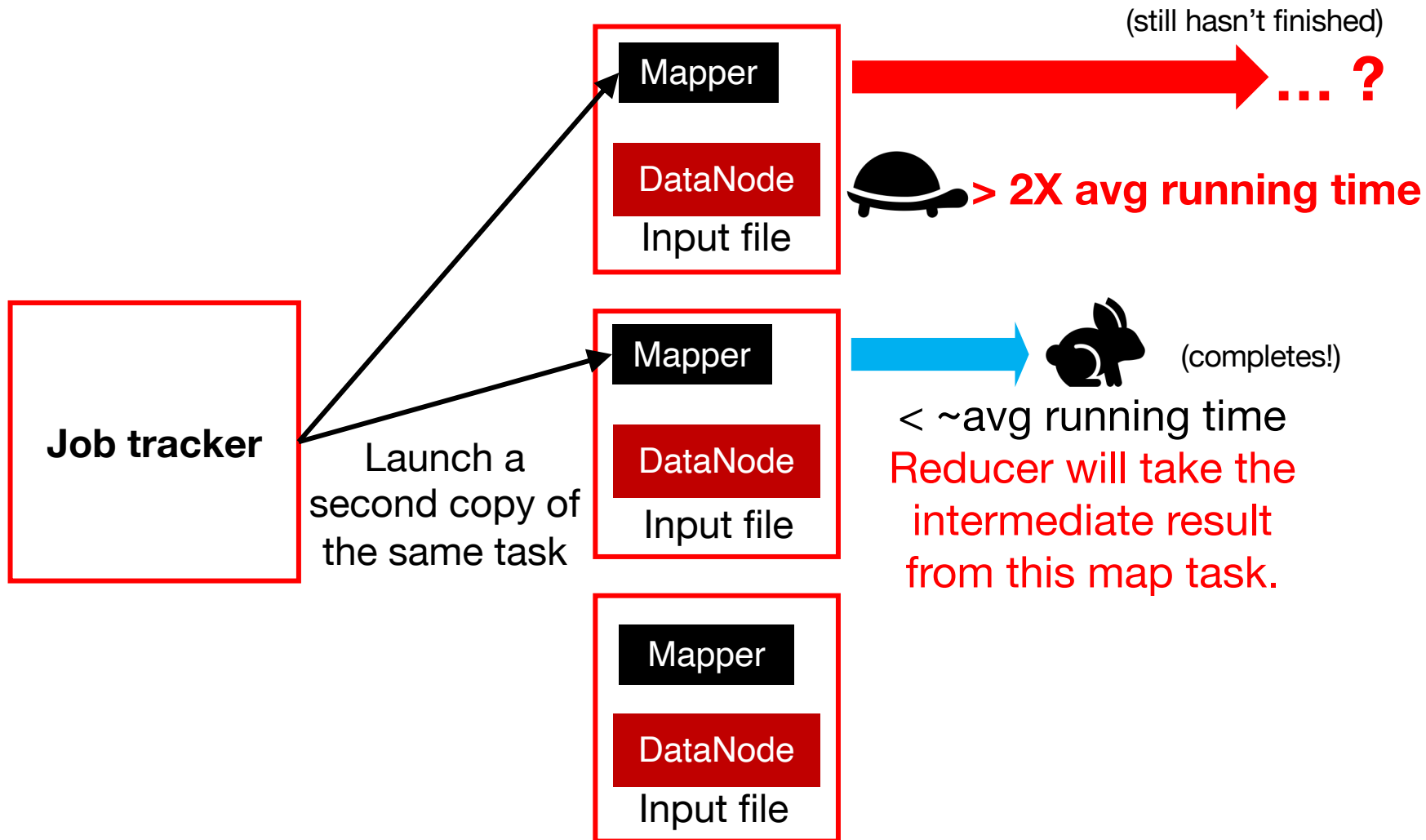
- If a task is going slowly (i.e., **straggler**):
 - Launch second copy of task (**backup task**) on another node
 - Take the output of whichever finishes first

Resilience against stragglers



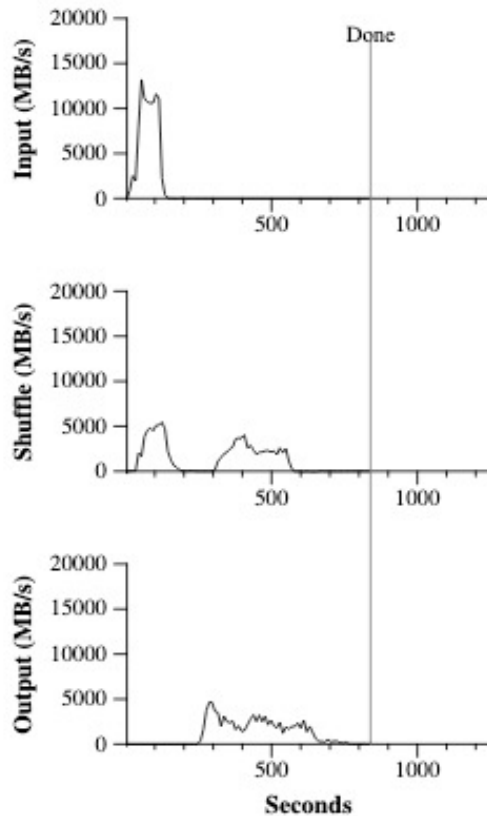
Resilience against stragglers

Cluster of machines

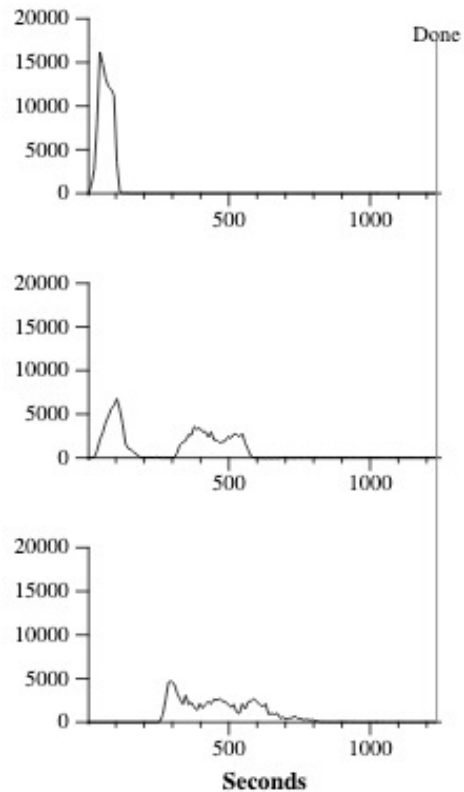


Would backup tasks cause correctness issue in MapReduce?

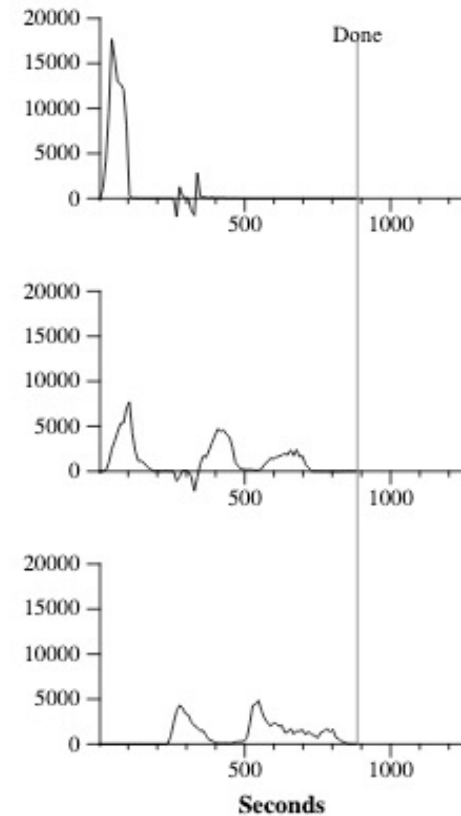
Discussion: MapReduce eval (paper)



(a) Normal execution



(b) No backup tasks



(c) 200 tasks killed