

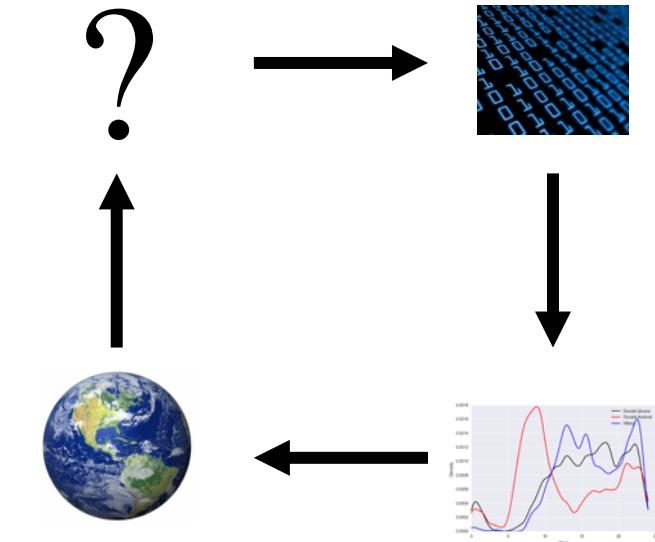
Big Data Analytics

Map-Reduce and Spark

Slides by:

Joseph E. Gonzalez

jegonzal@cs.berkeley.edu



Data in the Organization

A little bit of buzzword bingo!

Inventory



It is tempting to think of data as being organized in a single source-of-truth database.



Sales
(Asia)



Sales
(US)



Inventory

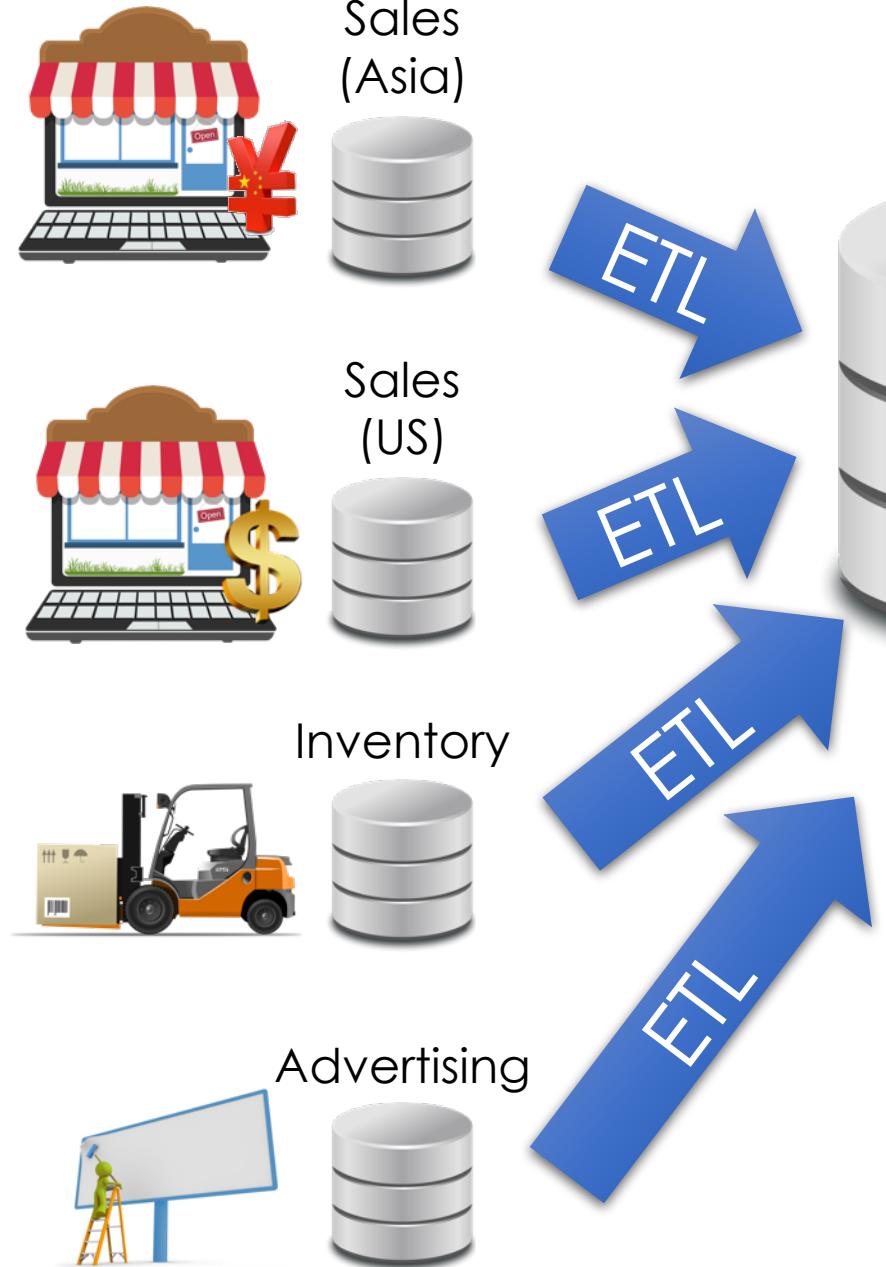


Advertising

Operational Data Stores

- Capture **the now**
- Many different databases across an organization
- Mission critical ... be careful!
 - Serving live ongoing business operations
 - Managing inventory
- Different formats (e.g., currency)
 - Different schemas (acquisitions ...)
- Live systems often don't maintain history

We would like a consolidated, clean, historical snapshot of the data.



Data Warehouse

Collects and organizes historical data from multiple sources

Data is periodically **ETL**ed into the data warehouse:

- **Extracted** from remote sources
- **Transformed** to standard schemas
- **Loaded** into the (typically) relational (SQL) data system

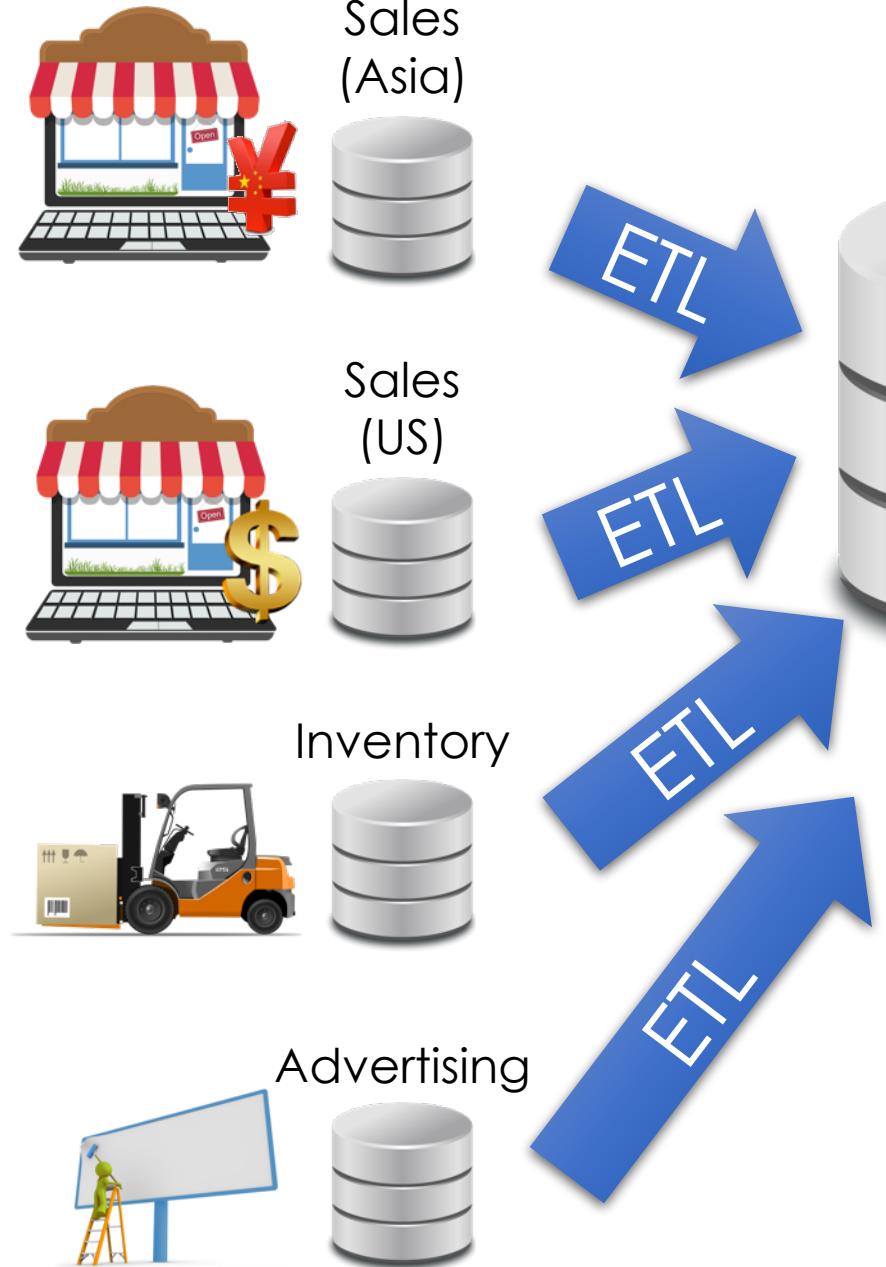
Extract → Transform → Load (ETL)

Extract & Load: provides a snapshot of operational data

- Historical snapshot
- Data in a single system
- Isolates analytics queries (e.g., Deep Learning) from business critical services (e.g., processing user purchase)
- Easy!

Transform: clean and prepare data for analytics in a unified representation

- **Difficult** → often requires specialized code and tools
- Different schemas, encodings, granularities



Data Warehouse

Collects and organizes historical data from multiple sources

How is data organized in the Data Warehouse?

Example Sales Data

pname	category	price	qty	date	day	city	state	country	
Corn	Food	25	25	3/30/16	Wed.	Omaha	NE	USA	
Corn	Food	25	8	3/31/16	Thu.	Omaha	NE	USA	
Corn	Food	25	15	4/1/16	Fri.	Omaha	NE	USA	
➤ Big table: many columns and rows	Galaxy 1	Phones	18	30	1/30/16	Wed.	Omaha	NE	USA
➤ Substantial redundancy → expensive to store and access	Galaxy 1	Phones	18	20	3/31/16	Thu.	Omaha	NE	USA
➤ Could we organize the data more efficiently?	Galaxy 1	Phones	18	50	4/1/16	Fri.	Omaha	NE	USA
Peanuts	Food	2	45	3/31/16	Wed.	Omaha	NE	USA	
					Thu.	Seoul		Korea	

Multidimensional Data Model

Sales **Fact Table**

pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
12	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35
11	2	2	22
11	3	2	10
12	1	2	26

Locations

locid	city	state	country
1	Omaha	Nebraska	USA
2	Seoul		Korea
5	Richmond	Virginia	USA

Products

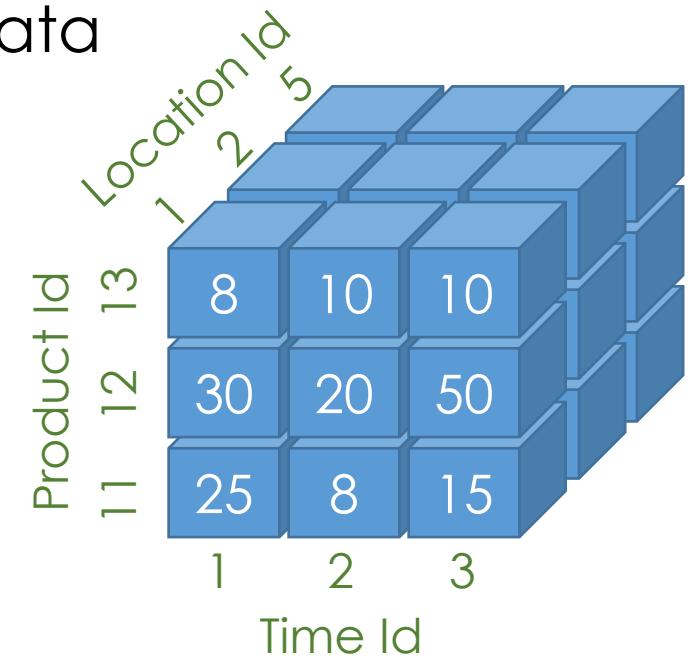
pid	pname	category	price
11	Corn	Food	25
12	Galaxy 1	Phones	18
13	Peanuts	Food	2

Time

timeid	Date	Day
1	3/30/16	Wed.
2	3/31/16	Thu.
3	4/1/16	Fri.

Dimension Tables

➤ Multidimensional “Cube” of data



Multidimensional Data Model

Sales **Fact Table**

pid	timeid	locid	sales
11	1	1	25
11	2	1	8
11	3	1	15
12	1	1	30
12	2	1	20
12	3	1	50
12	1	1	8
13	2	1	10
13	3	1	10
11	1	2	35
11	2	2	22
11	3	2	10
12	1	2	26

Locations

locid	city	state	country
1	Omaha	Nebraska	USA
2	Seoul		Korea
5	Richmond	Virginia	USA

Products

pid	pname	category	price
11	Corn	Food	25
12	Galaxy 1	Phones	18
13	Peanuts	Food	2

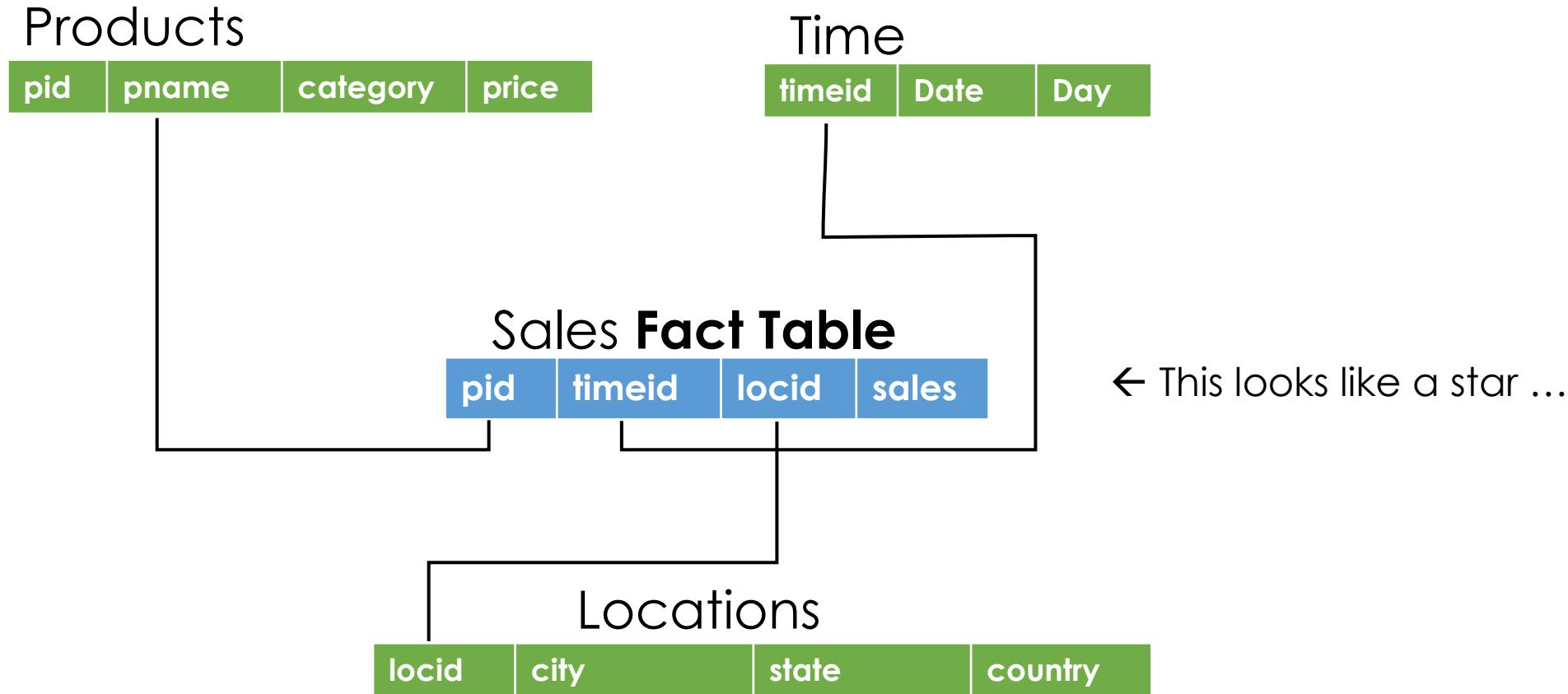
Time

timeid	Date	Day
1	3/30/16	Wed.
2	3/31/16	Thu.
3	4/1/16	Fri.

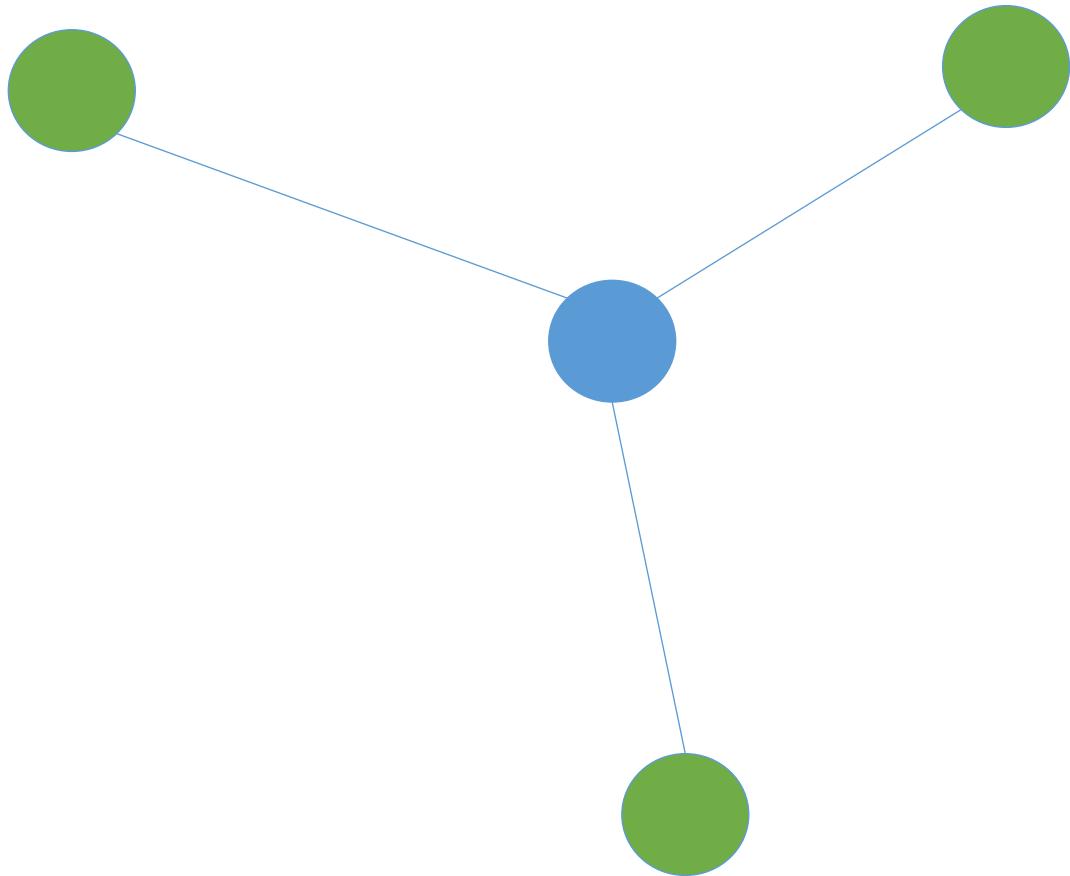
Dimension Tables

- Normalized Representation
- Fact Table
 - minimizes redundant info.
 - Reduces data errors
- Dimensions
 - easy to manage and summarize
 - Rename: Galaxy1 → Phablet
- How do we do analysis?
 - **Joins!!!!**

The Star Schema

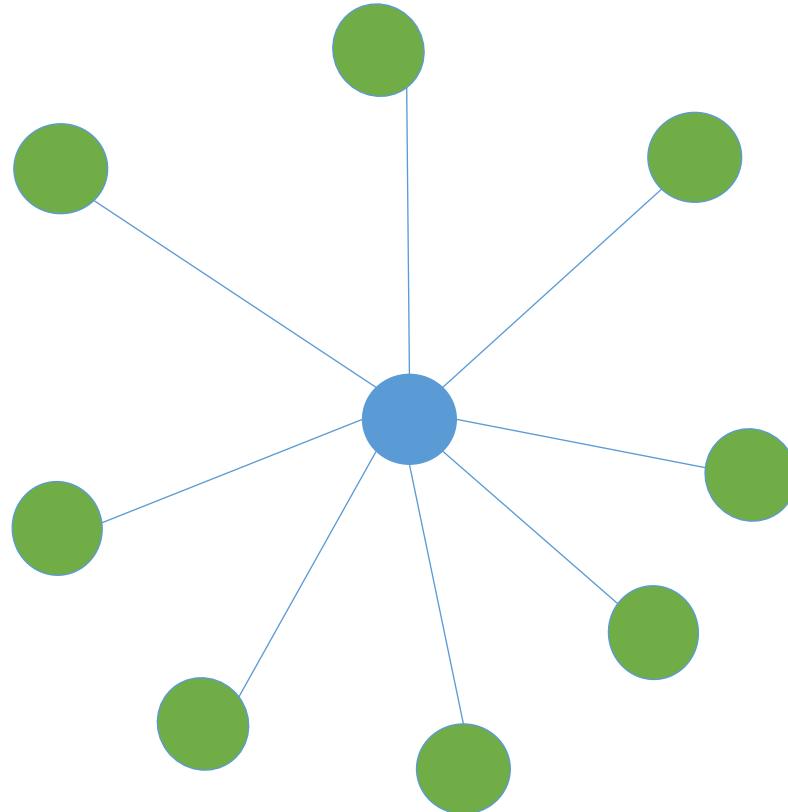


The Star Schema



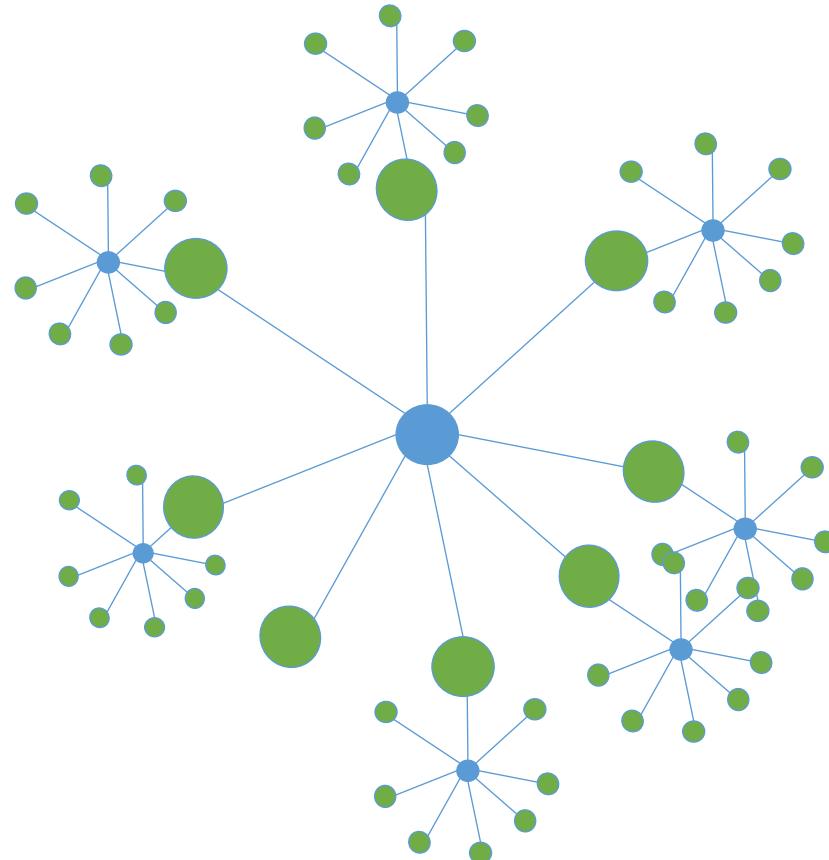
← This looks like a star ...

The Star Schema

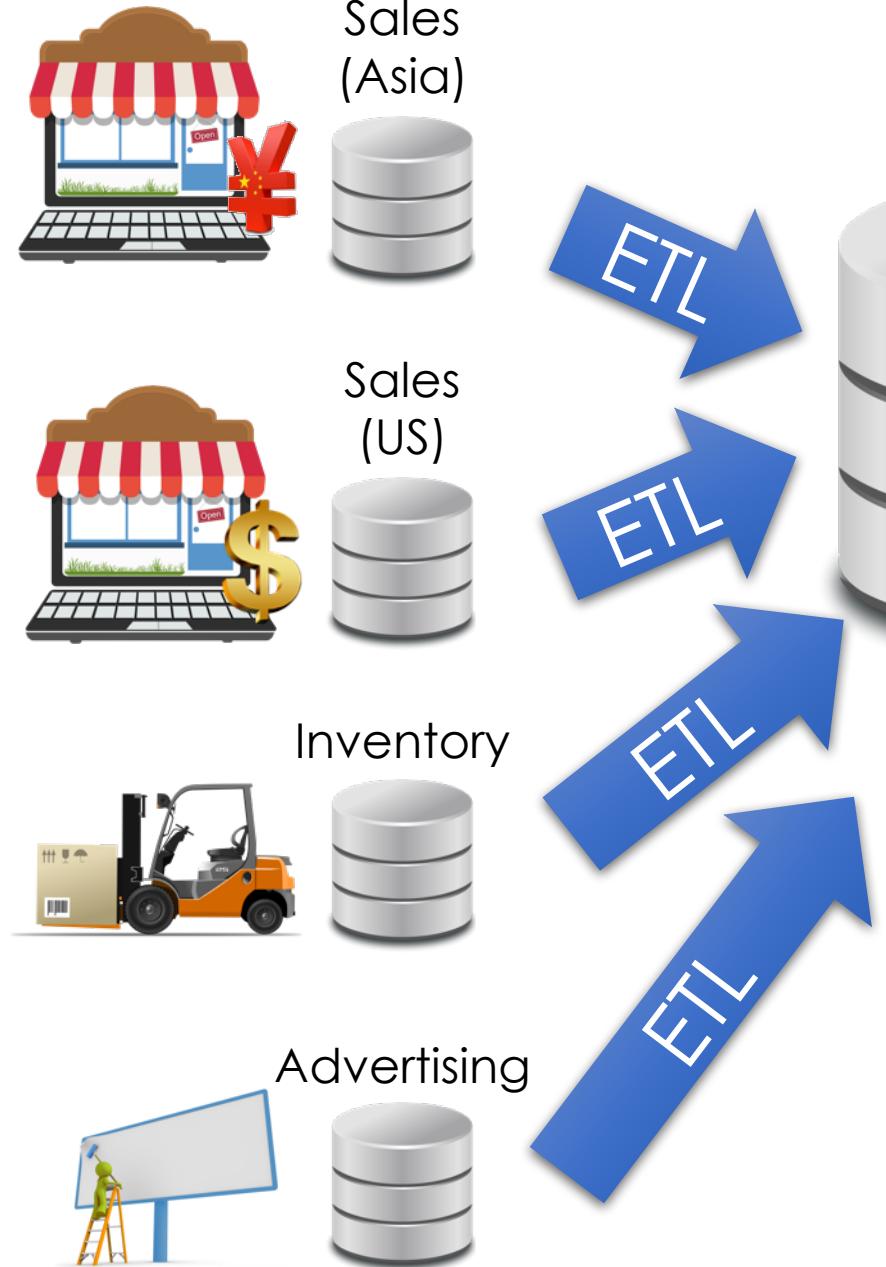


← This looks like a star ...

The Snowflake Schema



← This looks like a snowflake ...?



Data Warehouse

Collects and organizes historical data from multiple sources

How is data organized in the Data Warehouse?

Star Schemas



Data Warehouse

Collects and organizes historical data from multiple sources

- How do we deal with semi-structured and unstructured data?
- Do we really want to force a schema on load?



Sales
(Asia)

How do we **clean** and **organize** this data?

Depends on use ...



Text/Log Data



ETL?

Photos & Videos



Data Warehouse

Collects and organizes historical data from multiple sources

How do we **load** and **process** this data in a relational system?

➤ How do we deal with semi-structured data?

Depends on use ...
Can be difficult ...
Requires thought ...



Data Lake*

Store a copy of all the data

- in one place
- in its original “natural” form

Enable data consumers to choose how to transform and use data.

- Schema on Read

Enabled by new Tools:
Map-Reduce & Distributed Filesystems

What could go wrong?

*Still being defined...[Buzzword Disclaimer]

The Dark Side of Data Lakes



- Cultural shift: *Curate* → *Save Everything!*
 - Noise begins to dominate signal
- Limited data governance and planning
 - **Example:** `hdfs://important/joseph_big_file3.csv_with_json`
 - **What** does it contain?
 - **When** and **who** created it?
- No cleaning and verification → lots of dirty data
- New tools are more complex and old tools no longer work

Enter the data scientist

A Brighter Future for Data Lakes

Enter the data scientist

- Data scientists bring new skills
 - Distributed data processing and cleaning
 - Machine learning, computer vision, and statistical sampling
- Technologies are improving
 - SQL over large files
 - Self describing file formats & catalog managers
- Organizations are evolving
 - Tracking data usage and file permissions
 - Data engineers and managers



Distributed File Systems

Storing very large files

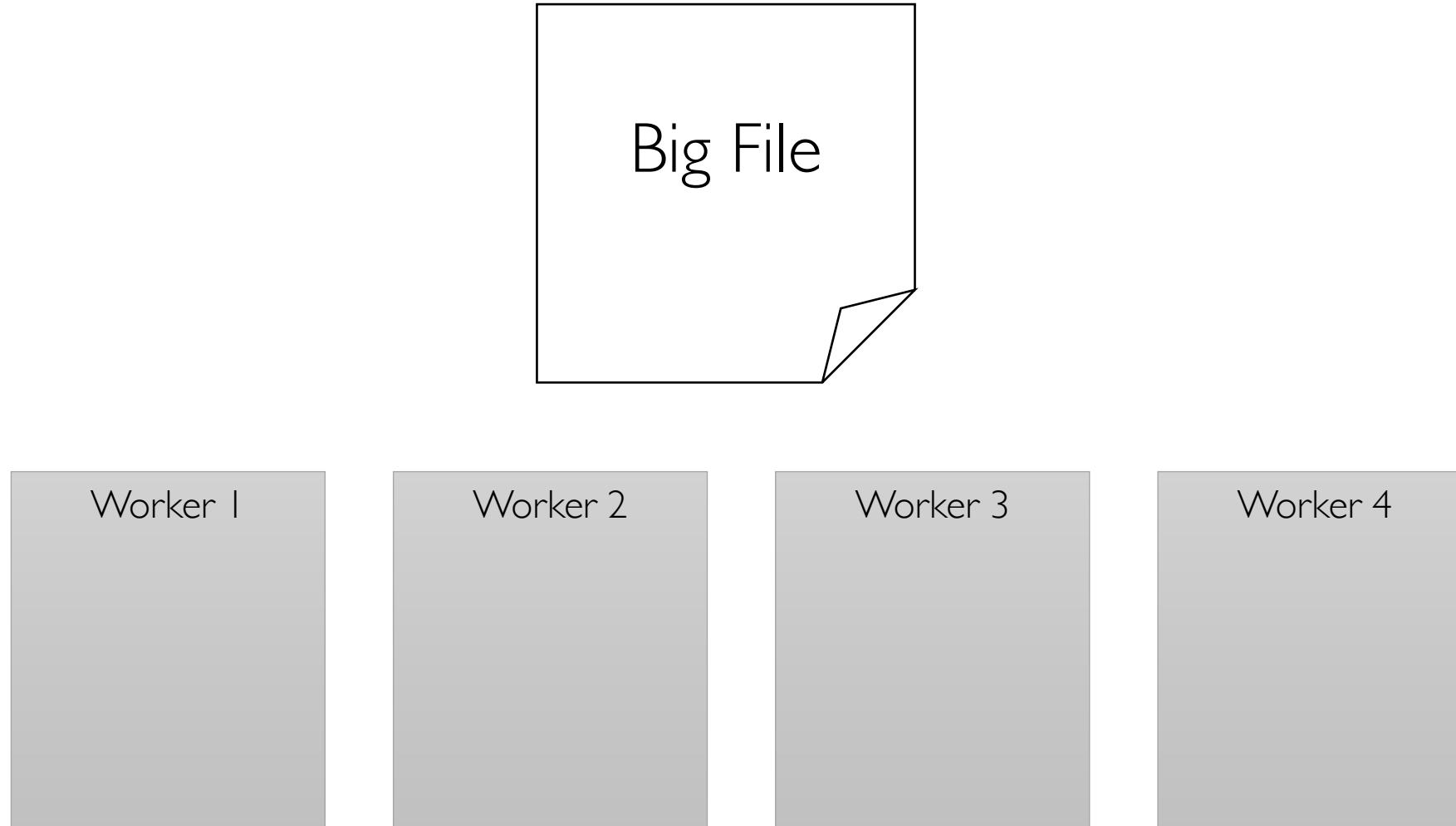
Fault Tolerant Distributed File Systems (e.g., HDFS)



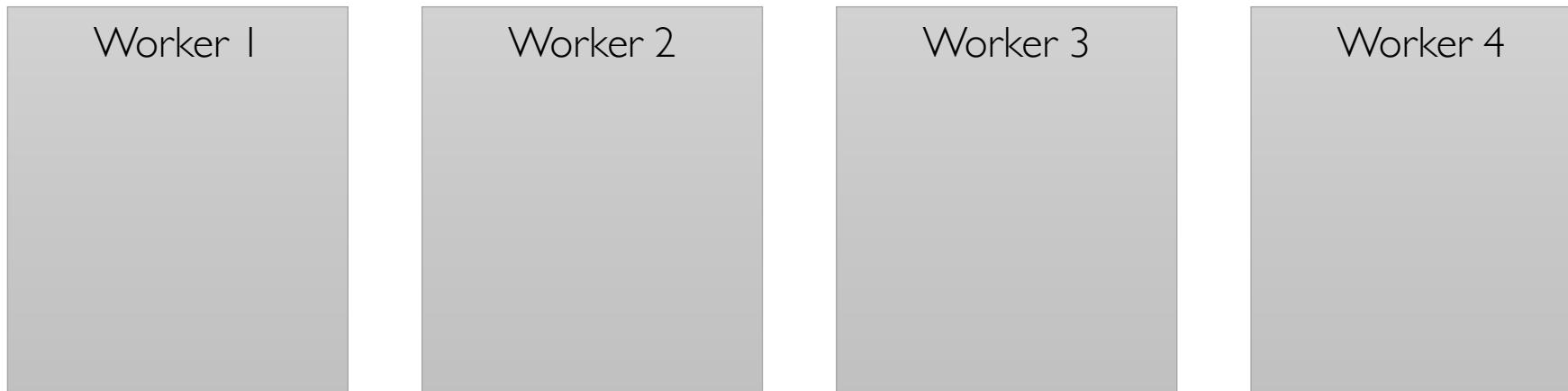
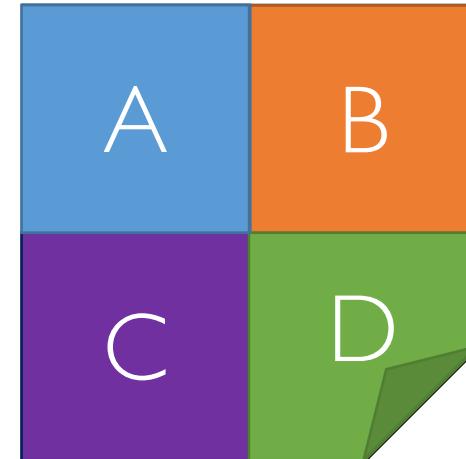
Big File

How do we **store** and **access** very
large files across **cheap**
commodity devices ?

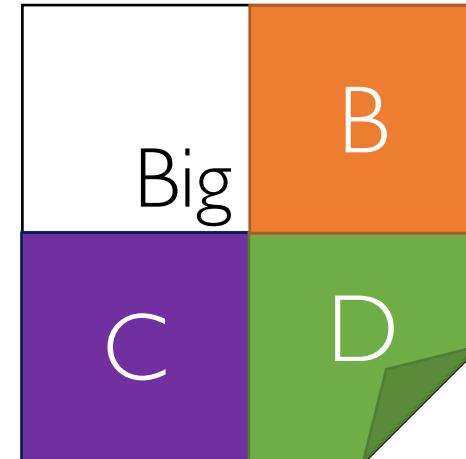
Fault Tolerant Distributed File Systems (e.g., HDFS)



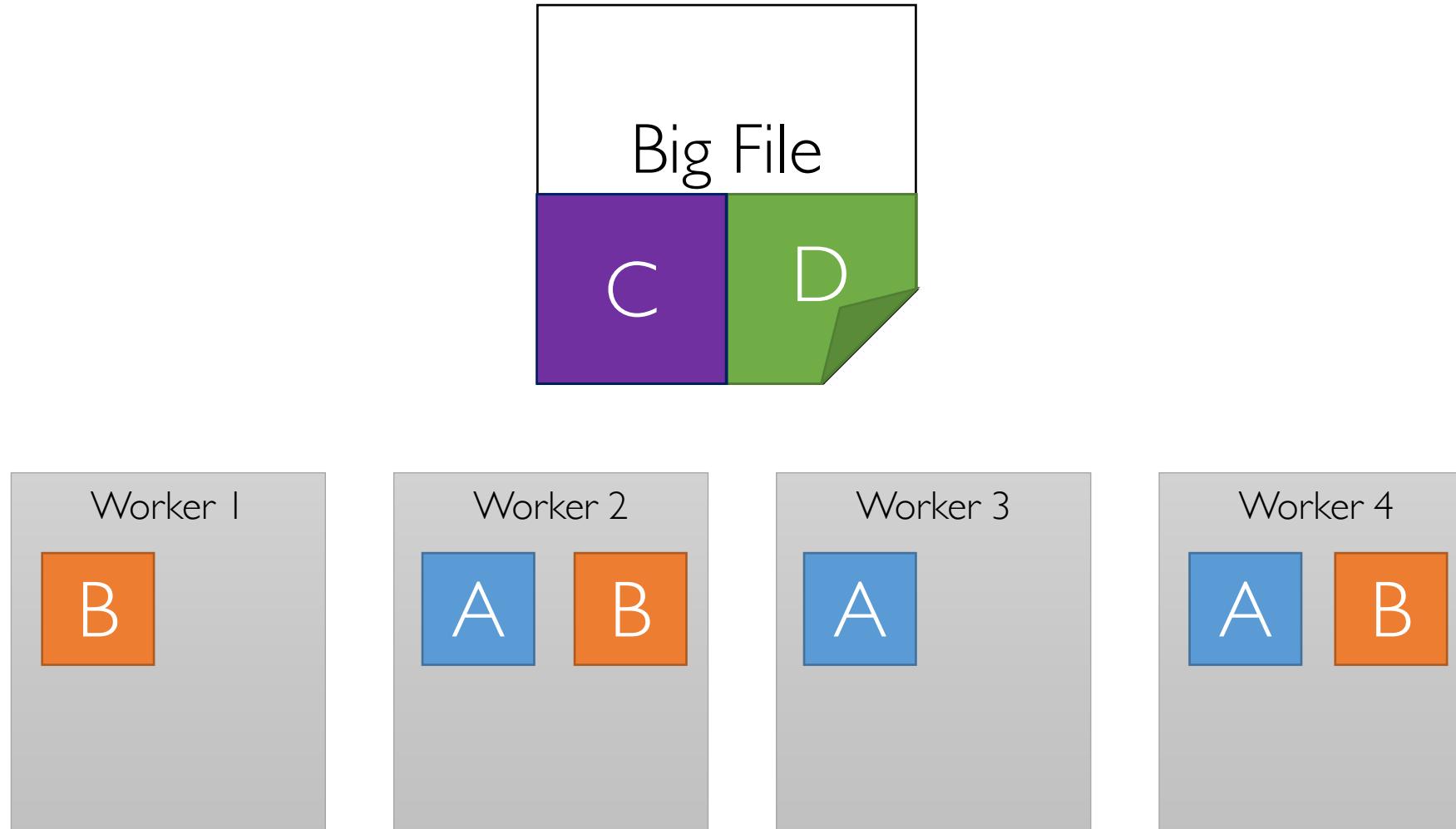
Fault Tolerant Distributed File Systems (e.g., HDFS)



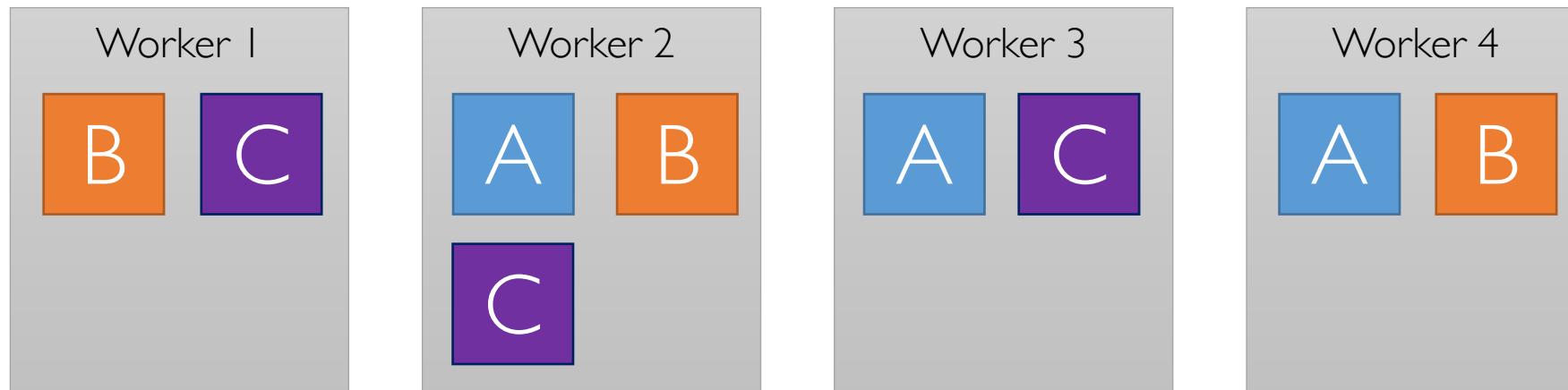
Fault Tolerant Distributed File Systems (e.g., HDFS)



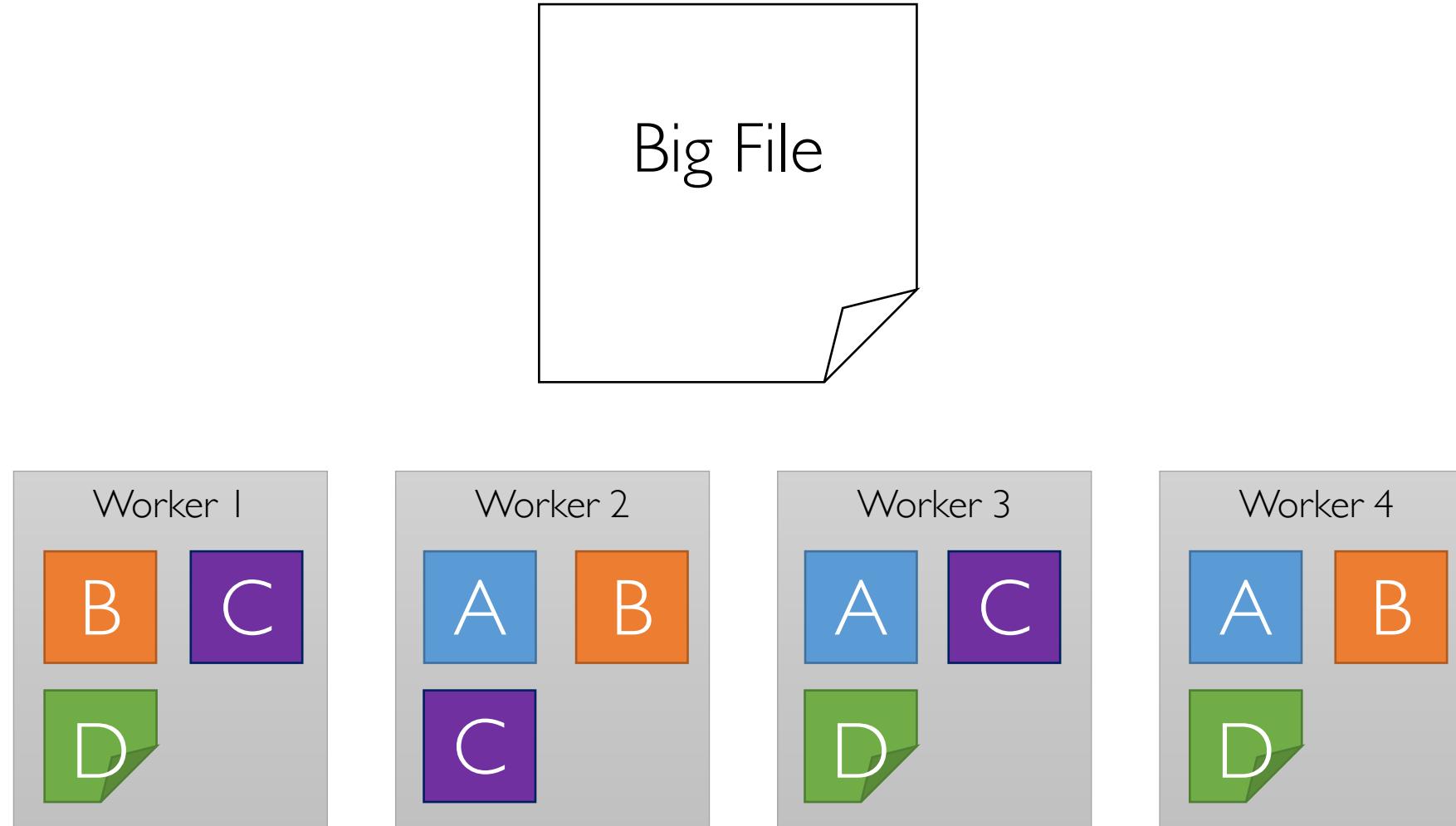
Fault Tolerant Distributed File Systems (e.g., HDFS)



Fault Tolerant Distributed File Systems (e.g., HDFS)



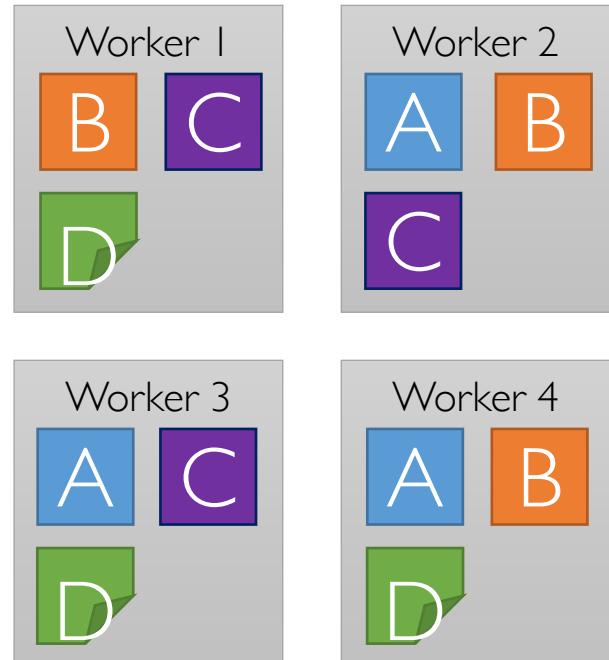
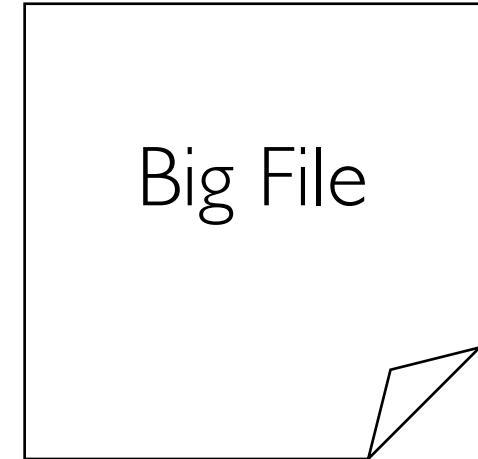
Fault Tolerant Distributed File Systems (e.g., HDFS)



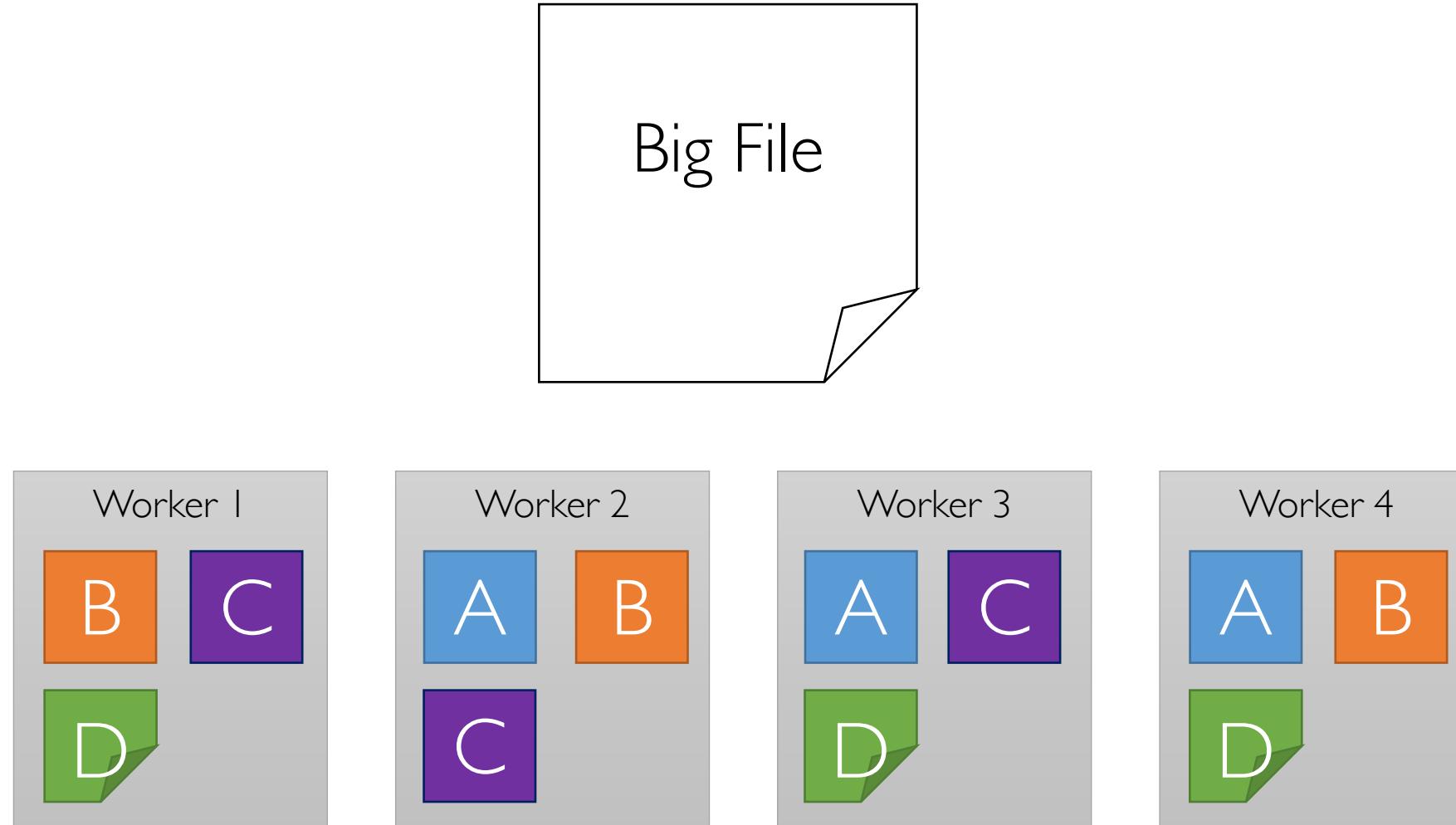
Fault Tolerant Distributed File Systems (e.g., HDFS)

- Split large files over multiple machines
 - Easily support very massive files spanning
- Read parts of file in parallel
 - Fast reads of large files
- Often built using cheap commodity storage devices

Cheap commodity storage devices will fail!

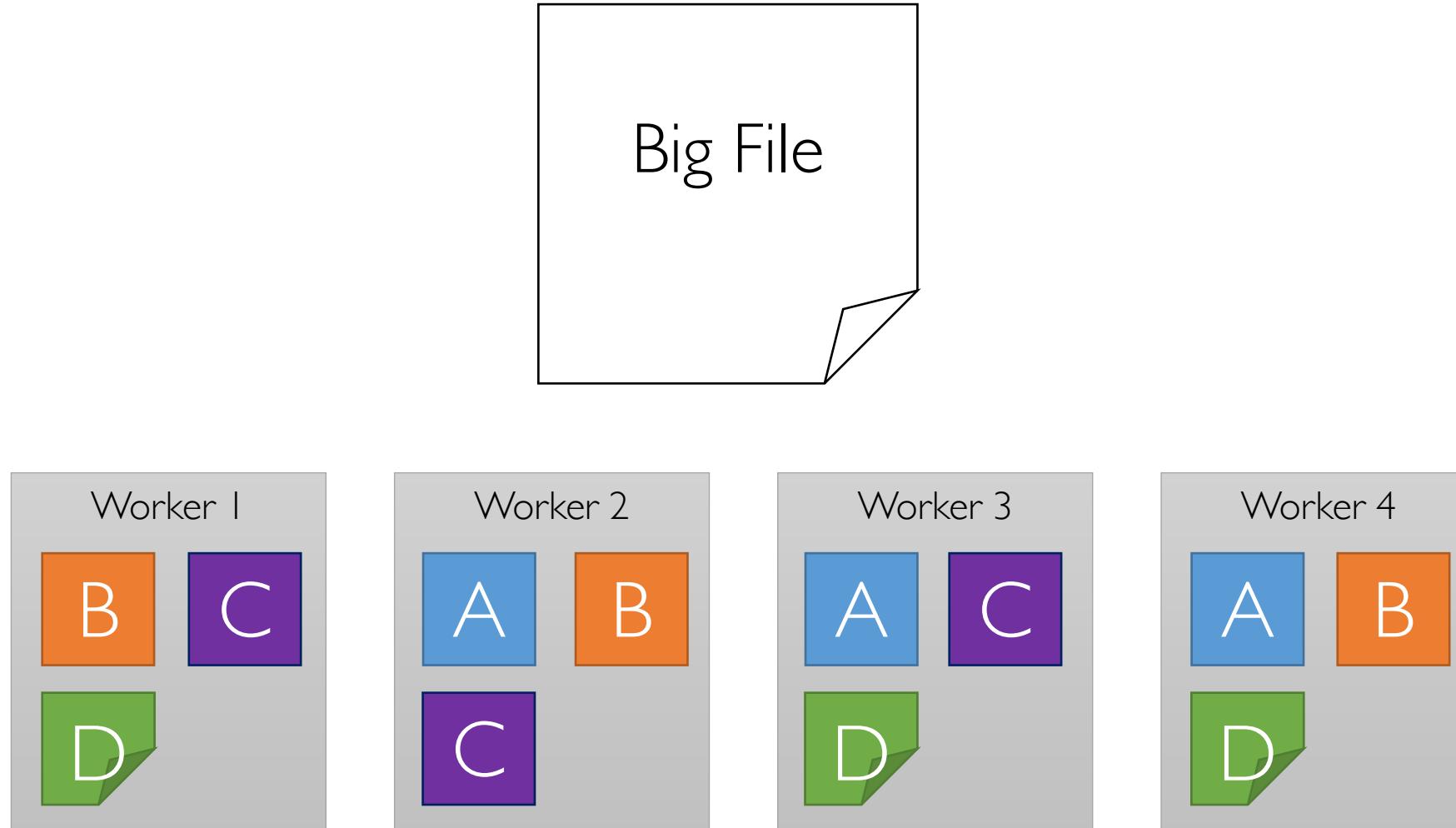


Fault Tolerant Distributed File Systems (e.g., HDFS)



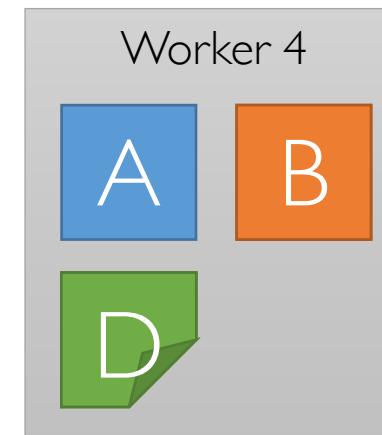
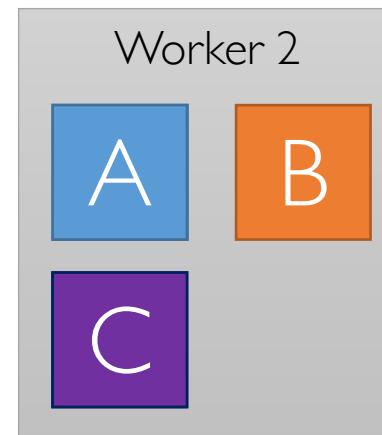
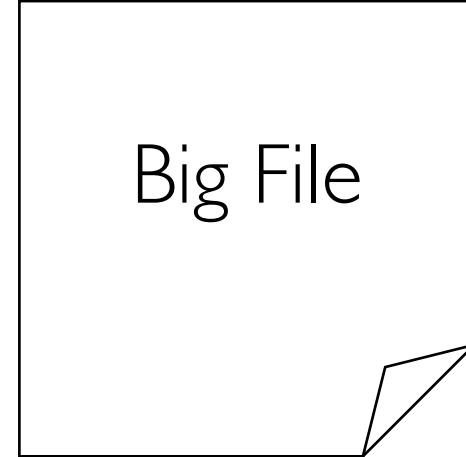
Fault Tolerant Distributed File Systems

Failure Event



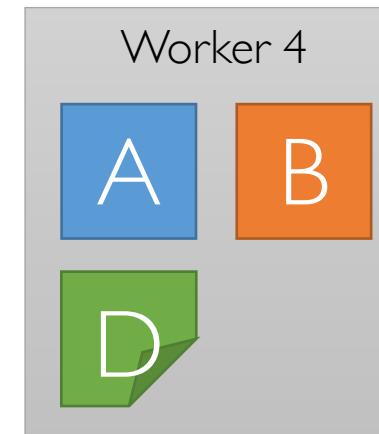
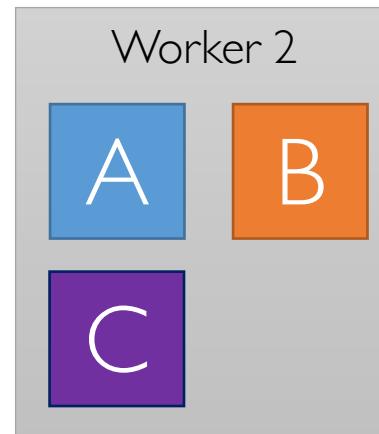
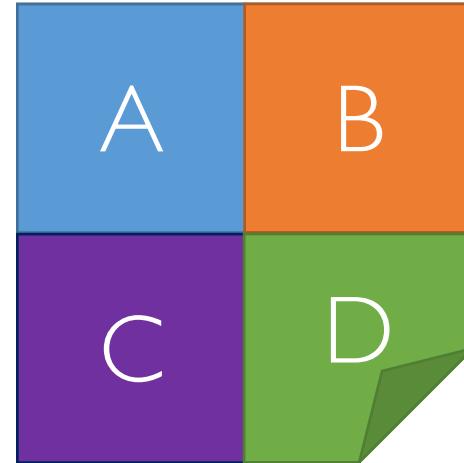
Fault Tolerant Distributed File Systems

Failure Event



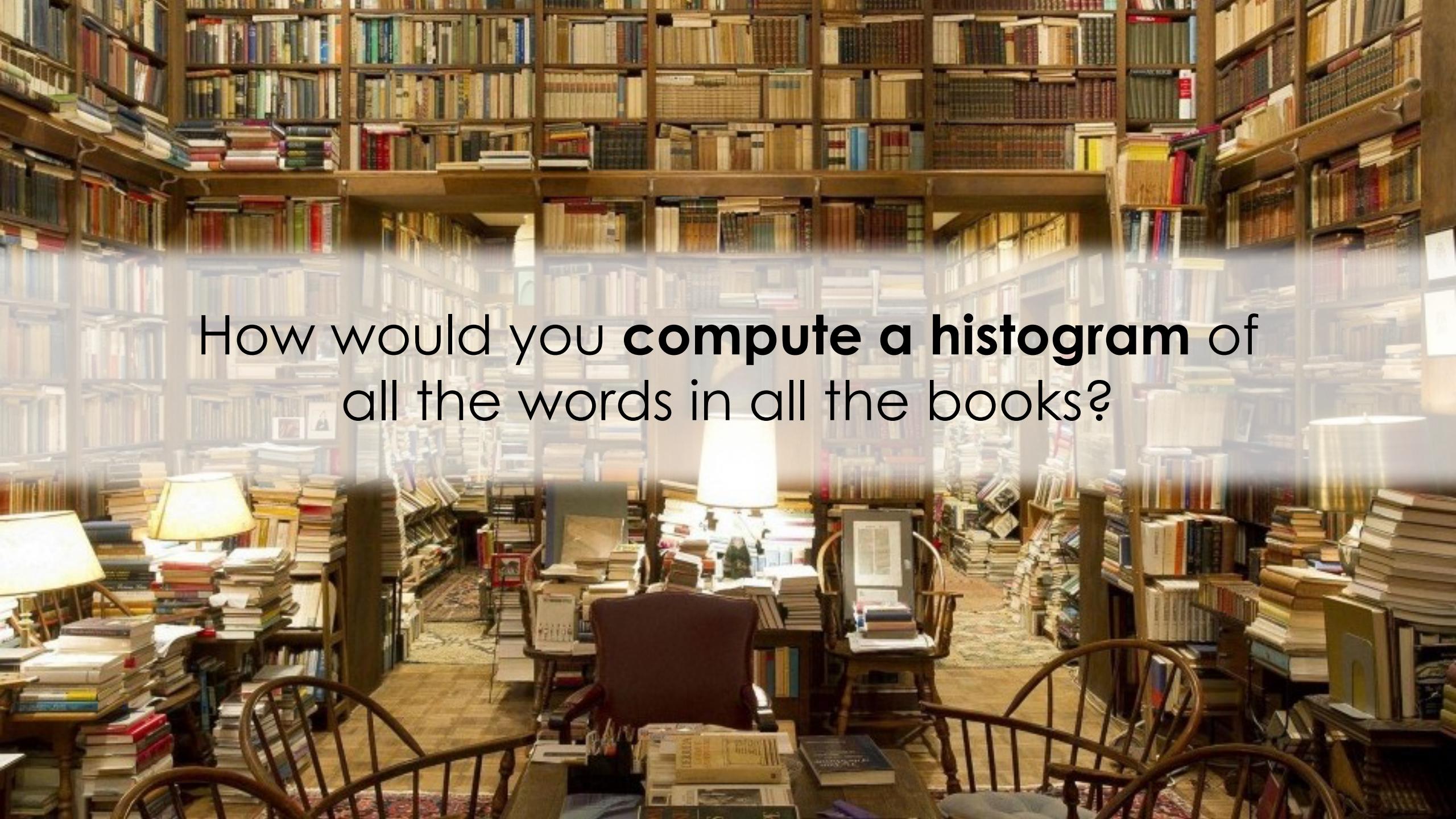
Fault Tolerant Distributed File Systems

Failure Event



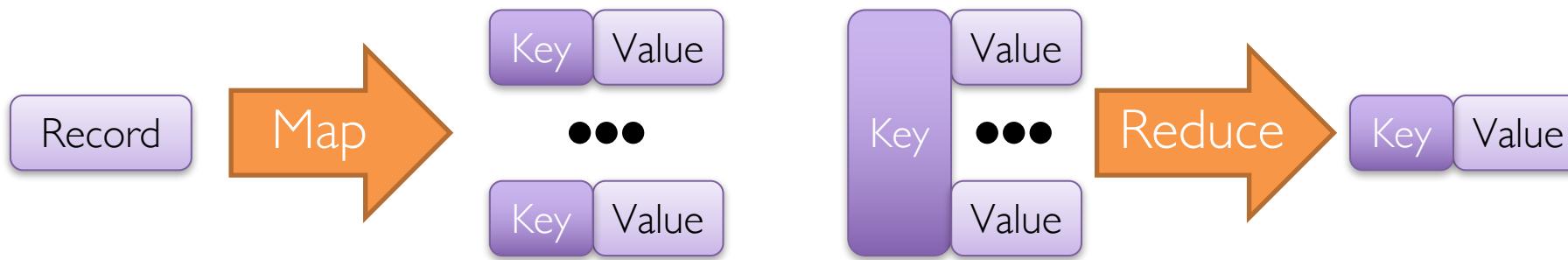
Map-Reduce Distributed Aggregation

Computing are very large files



How would you **compute a histogram** of all the words in all the books?

The Map Reduce Abstraction



Example: *Word-Count*

```
Map(docRecord) {  
    for (word in docRecord) {  
        emit (word, 1)  
    }  
}
```

Key Value

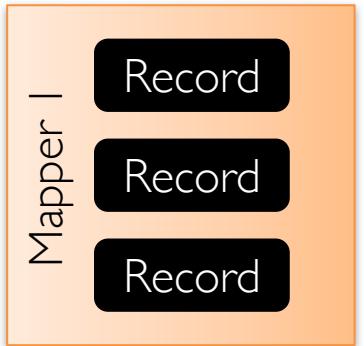
```
Reduce(word, counts) {  
    emit (word, SUM(counts))  
}
```

Map: Idempotent
Reduce: Commutative and Associative

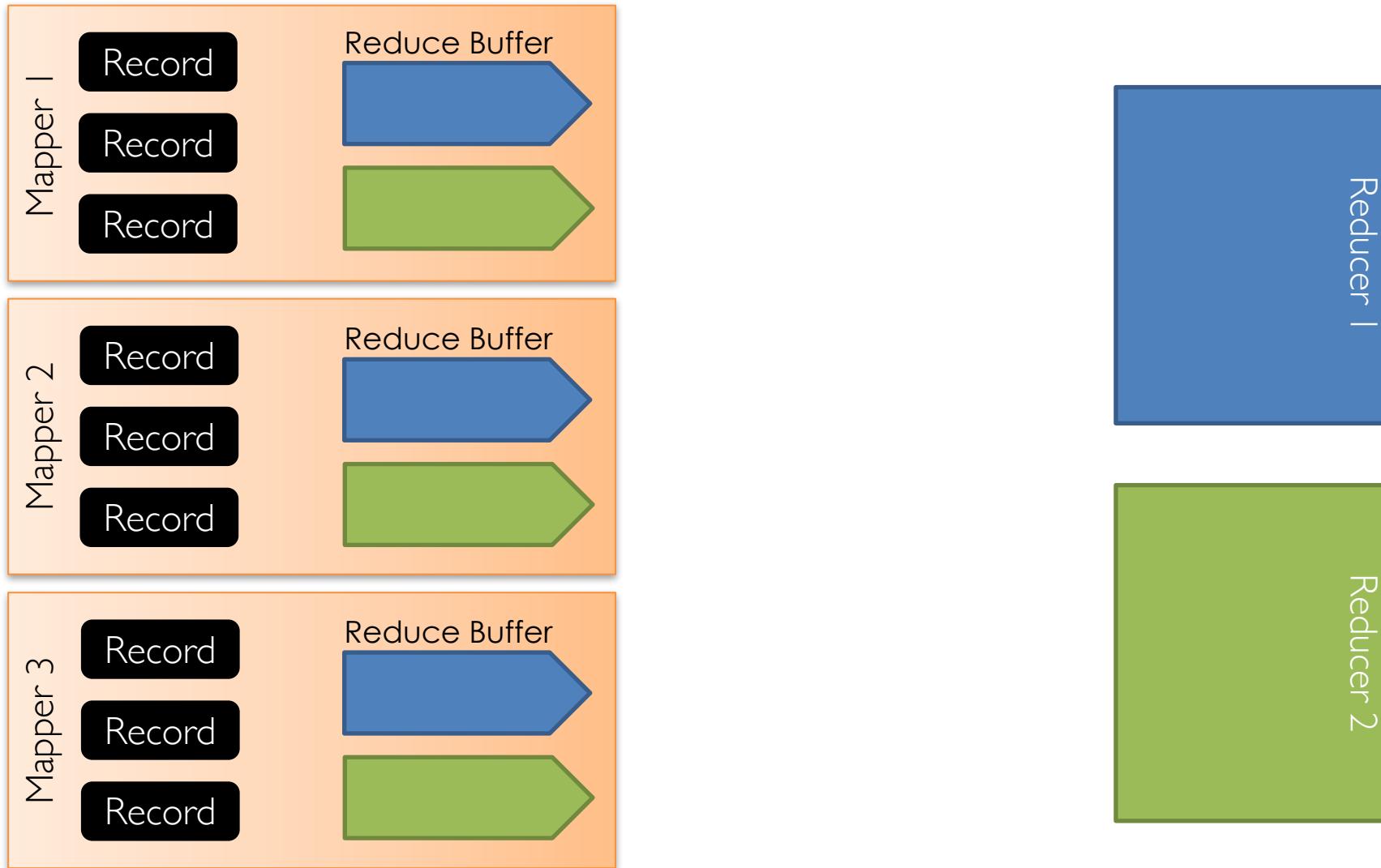
The Map Reduce System



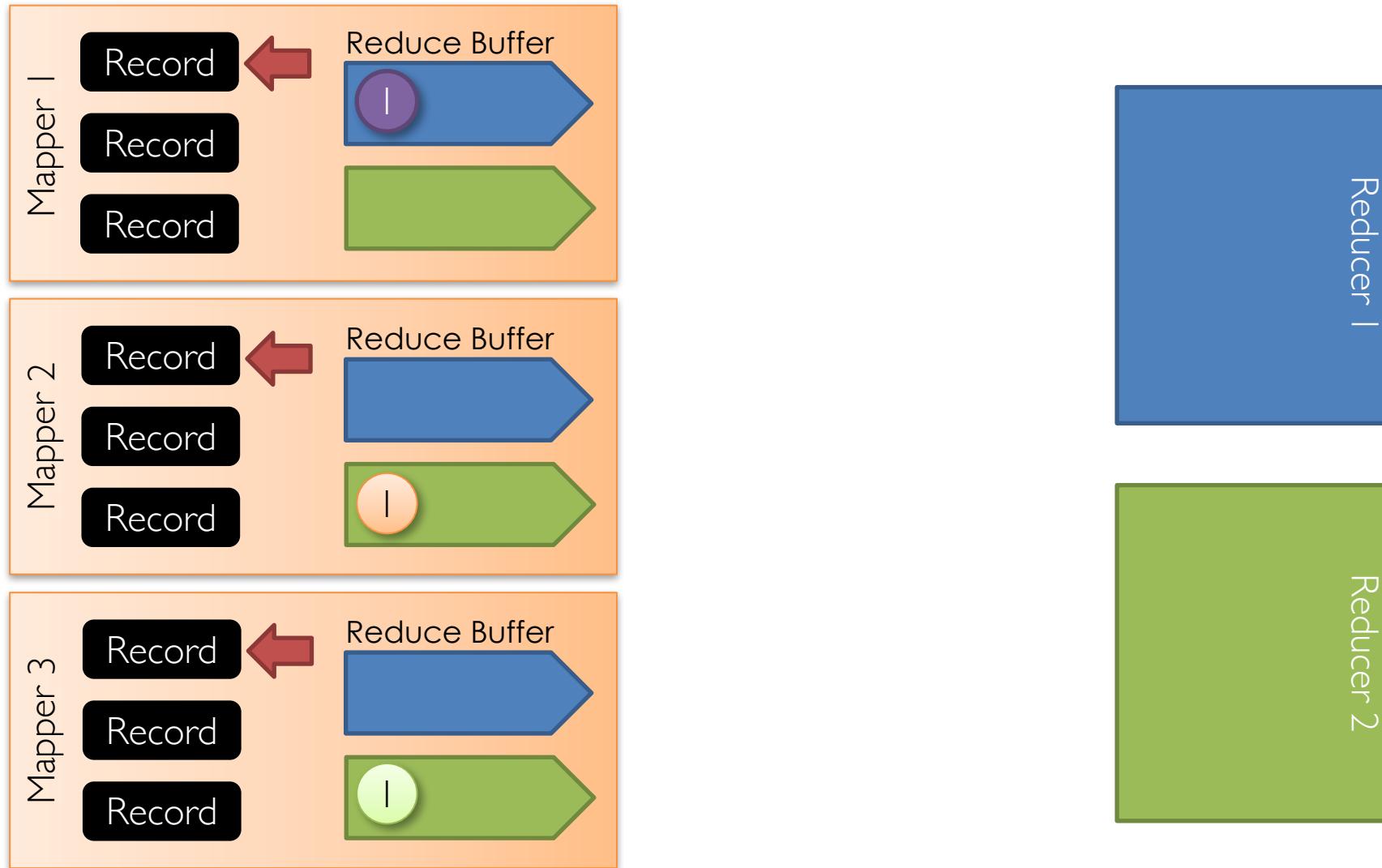
The Map Reduce System



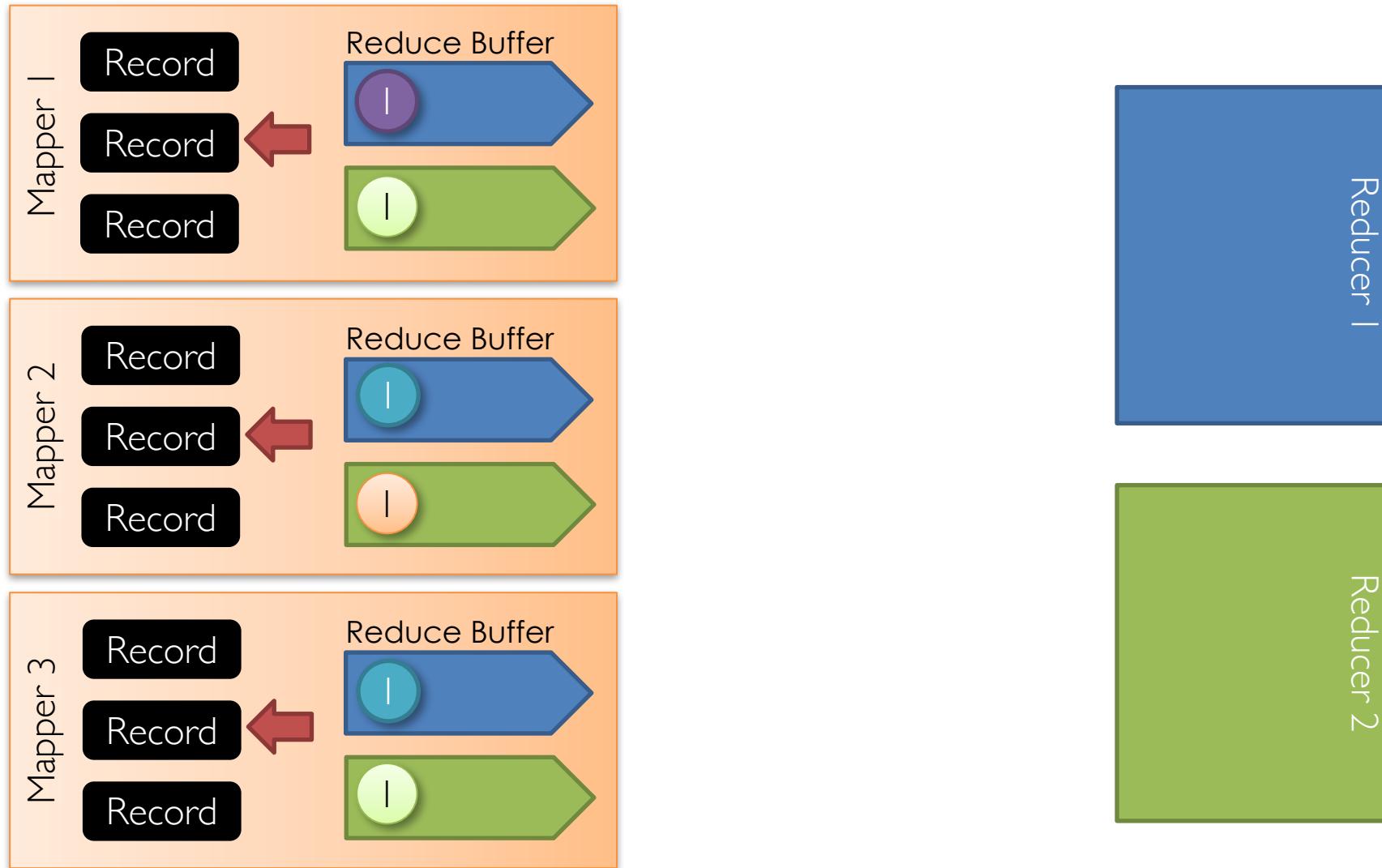
The Map Reduce System



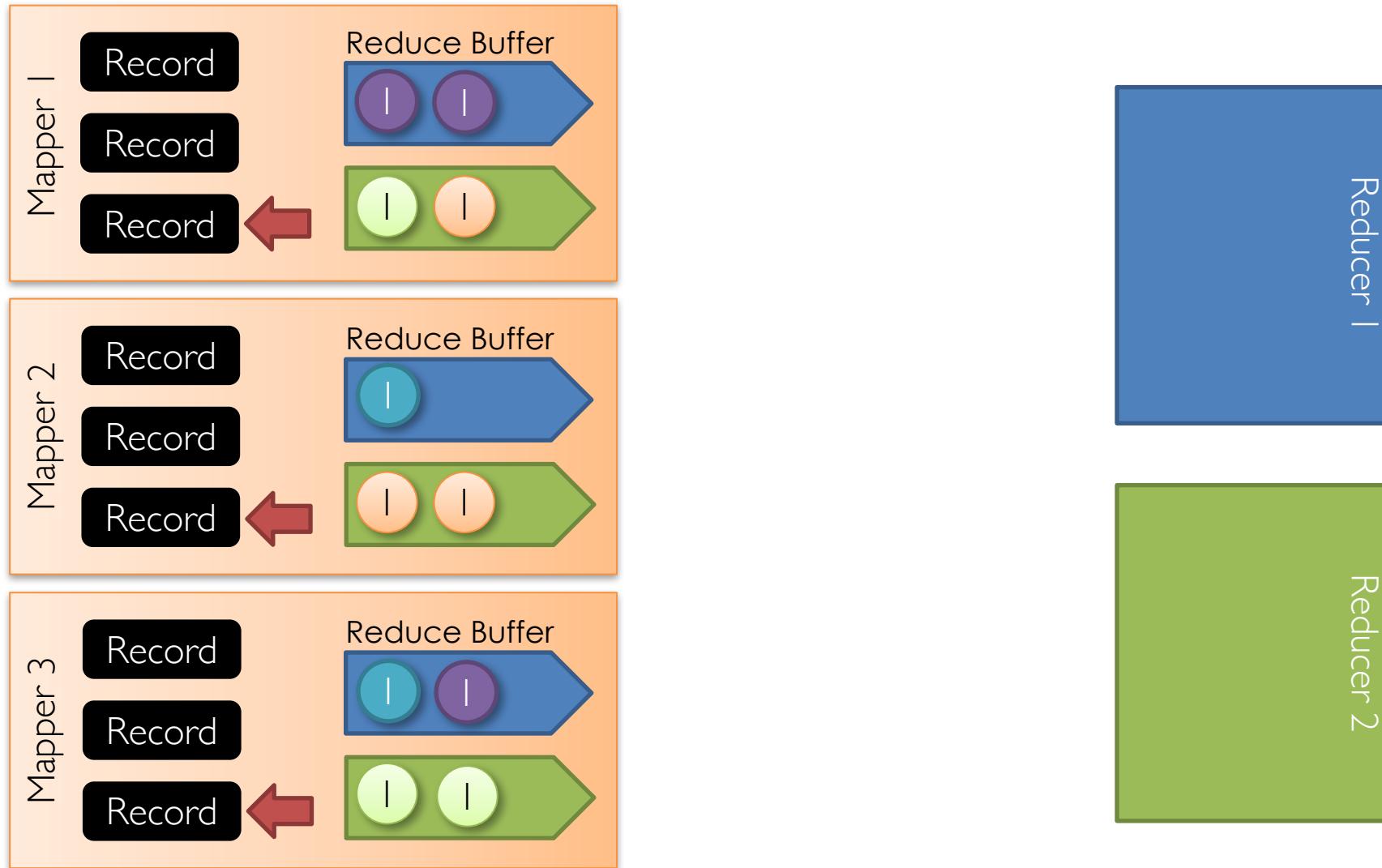
The Map Reduce System



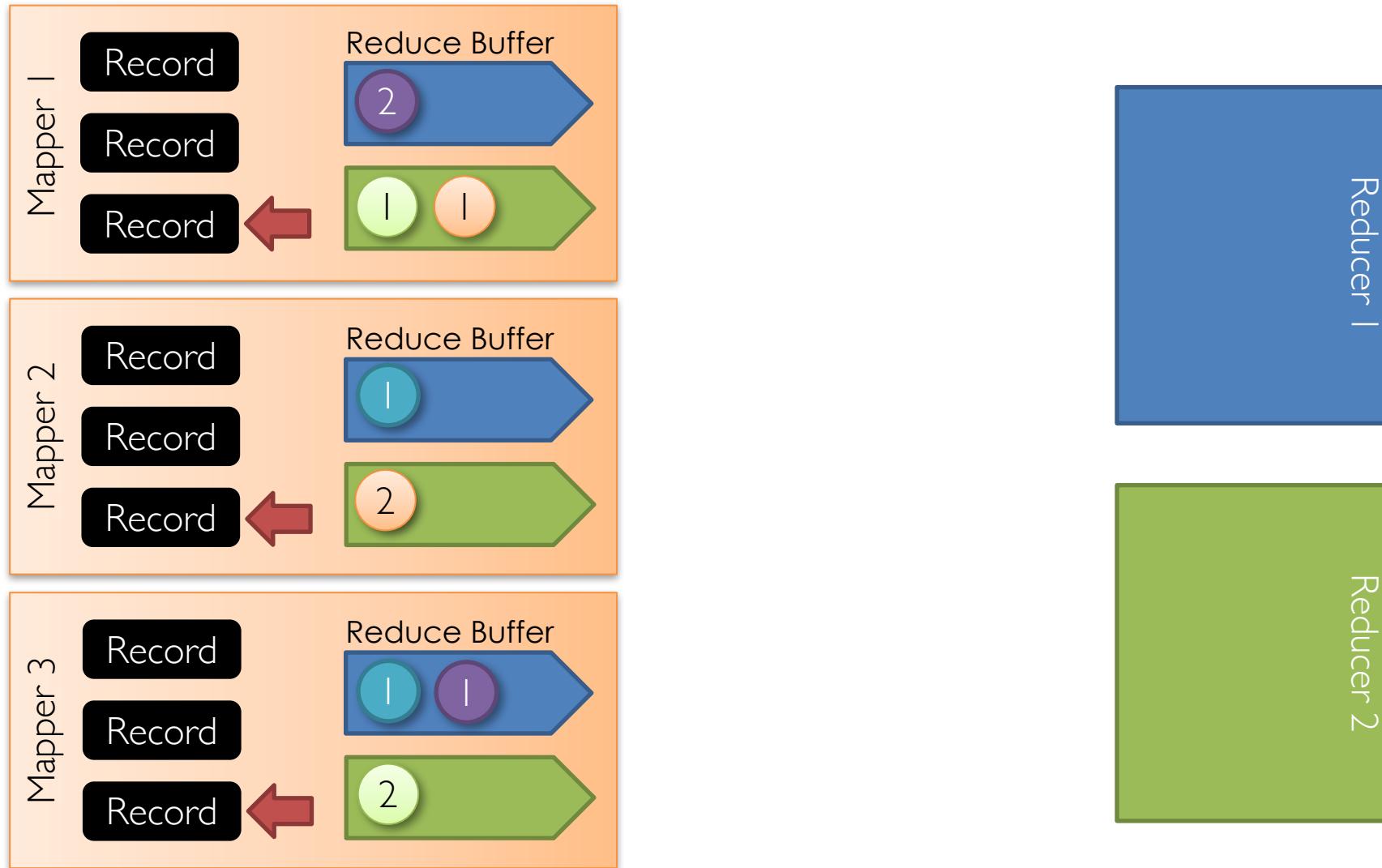
The Map Reduce System



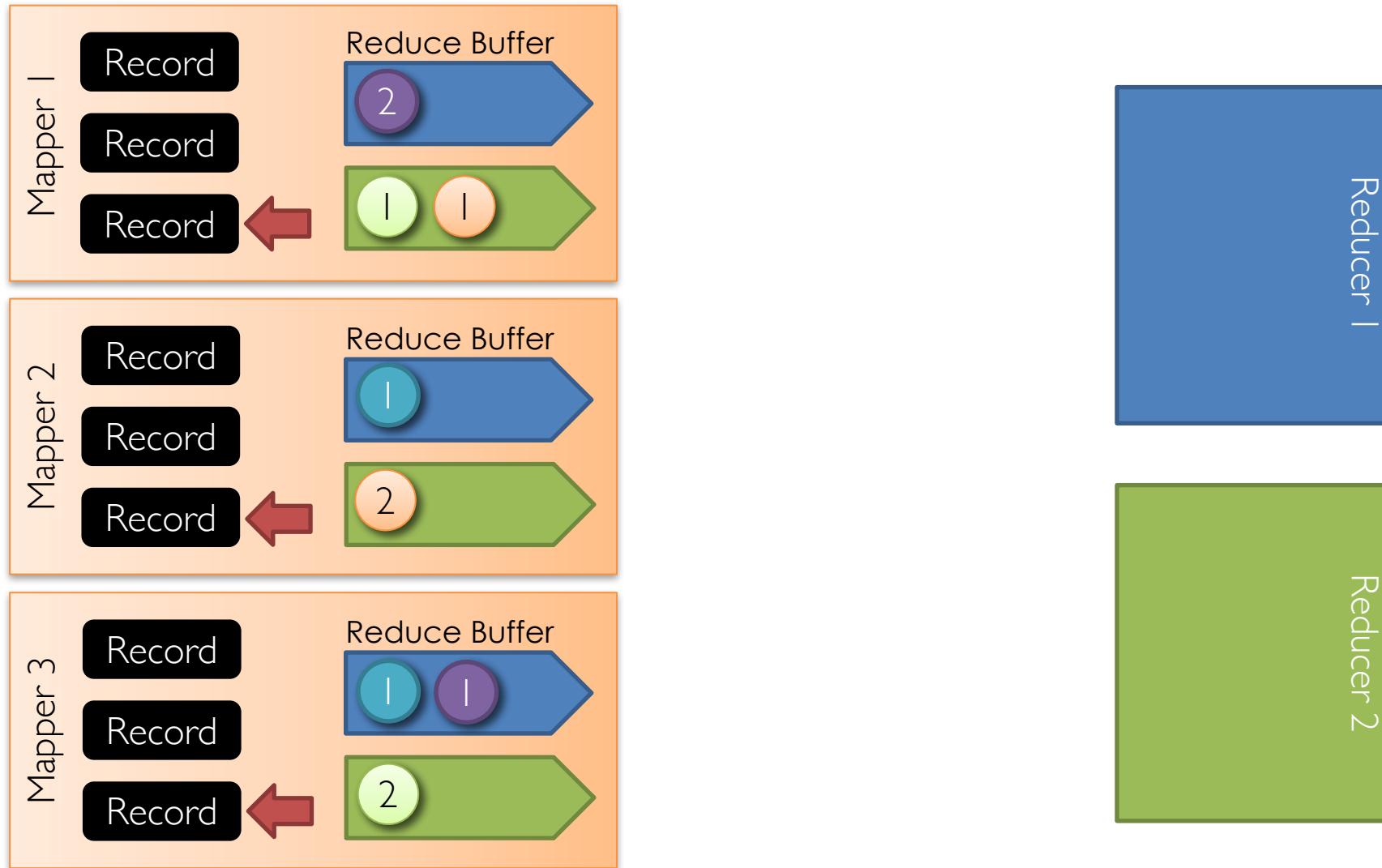
The Map Reduce System



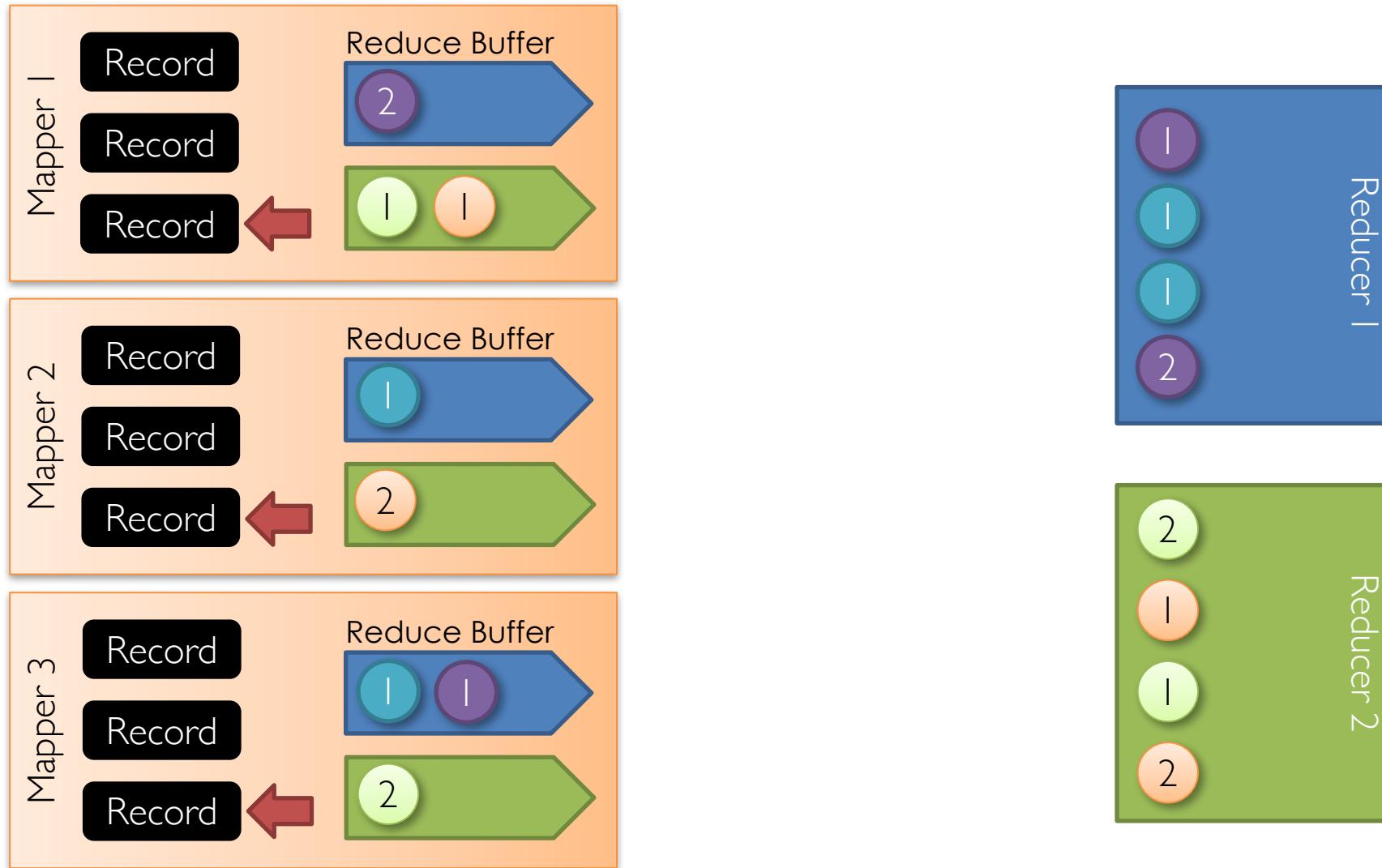
The Map Reduce System



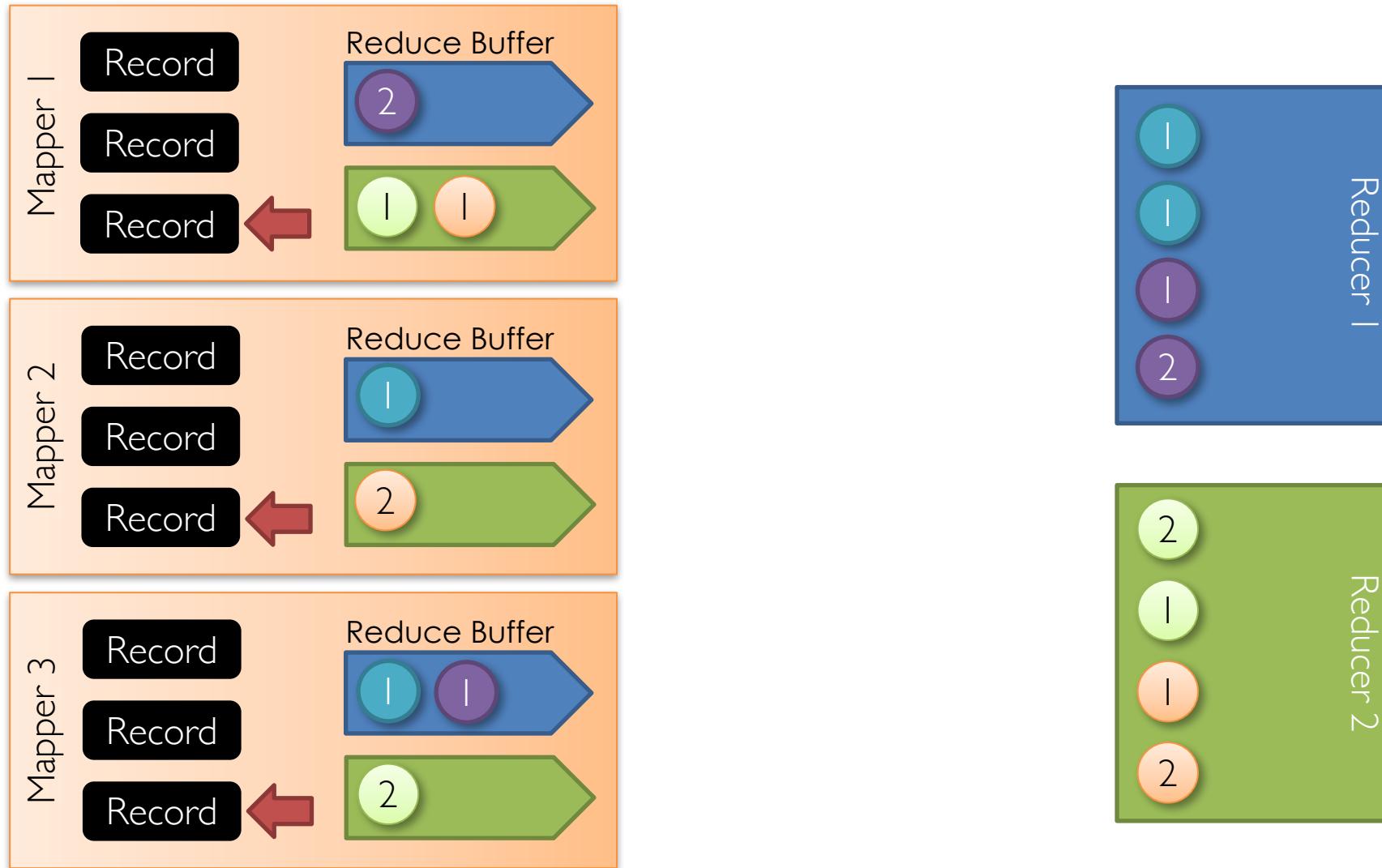
The Map Reduce System



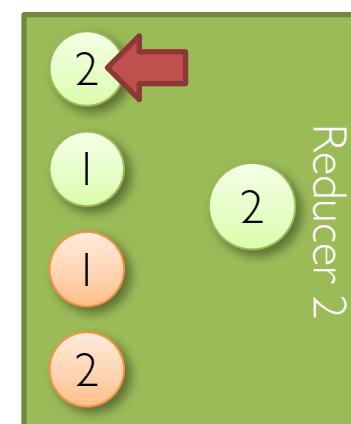
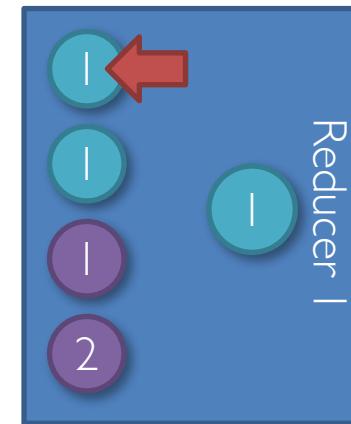
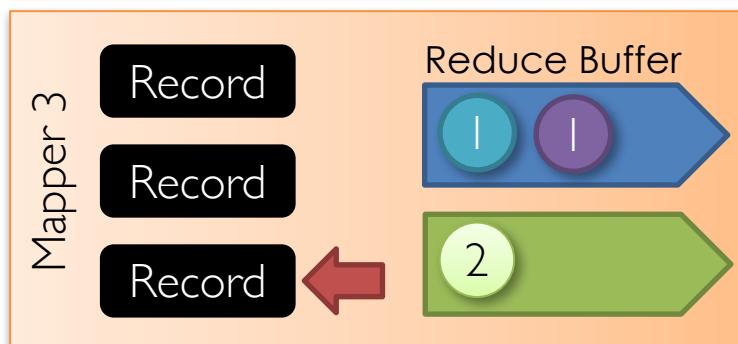
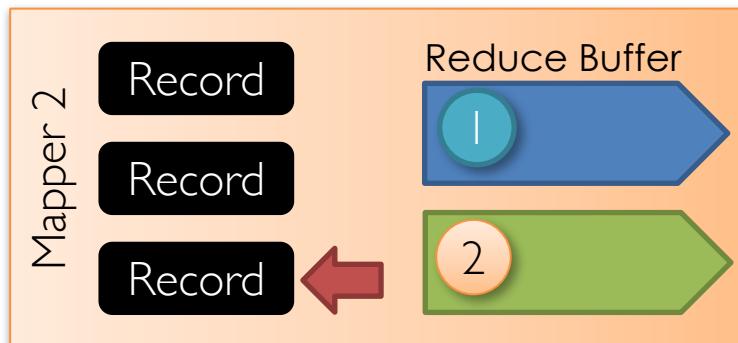
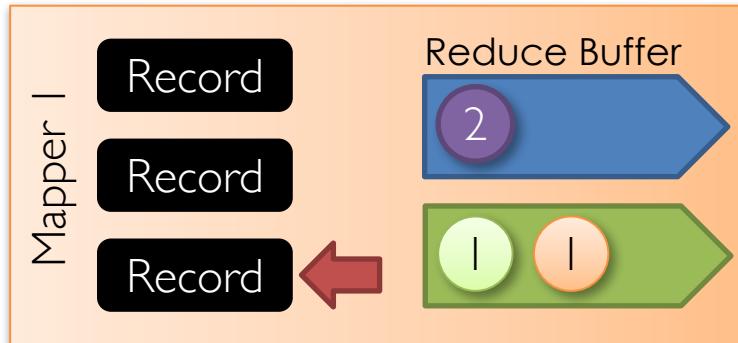
The Map Reduce System



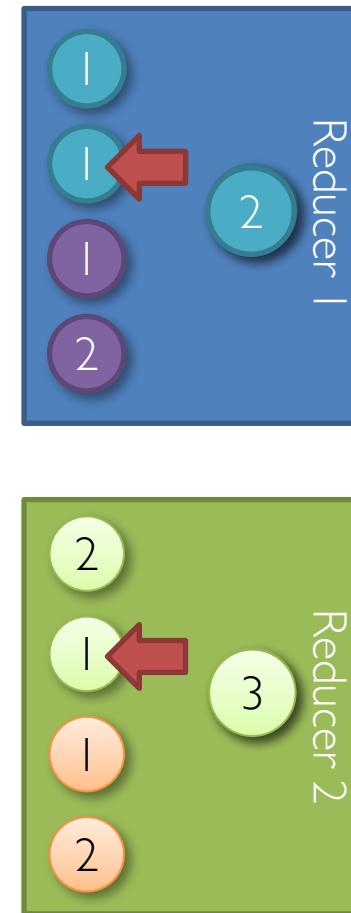
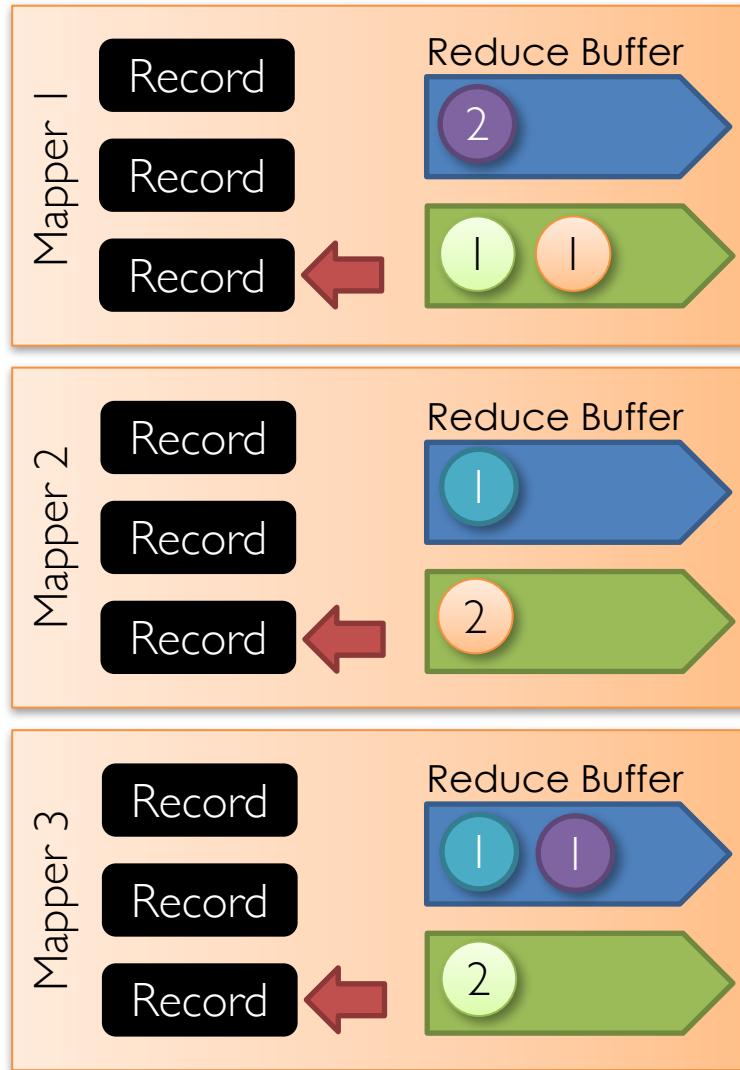
The Map Reduce System



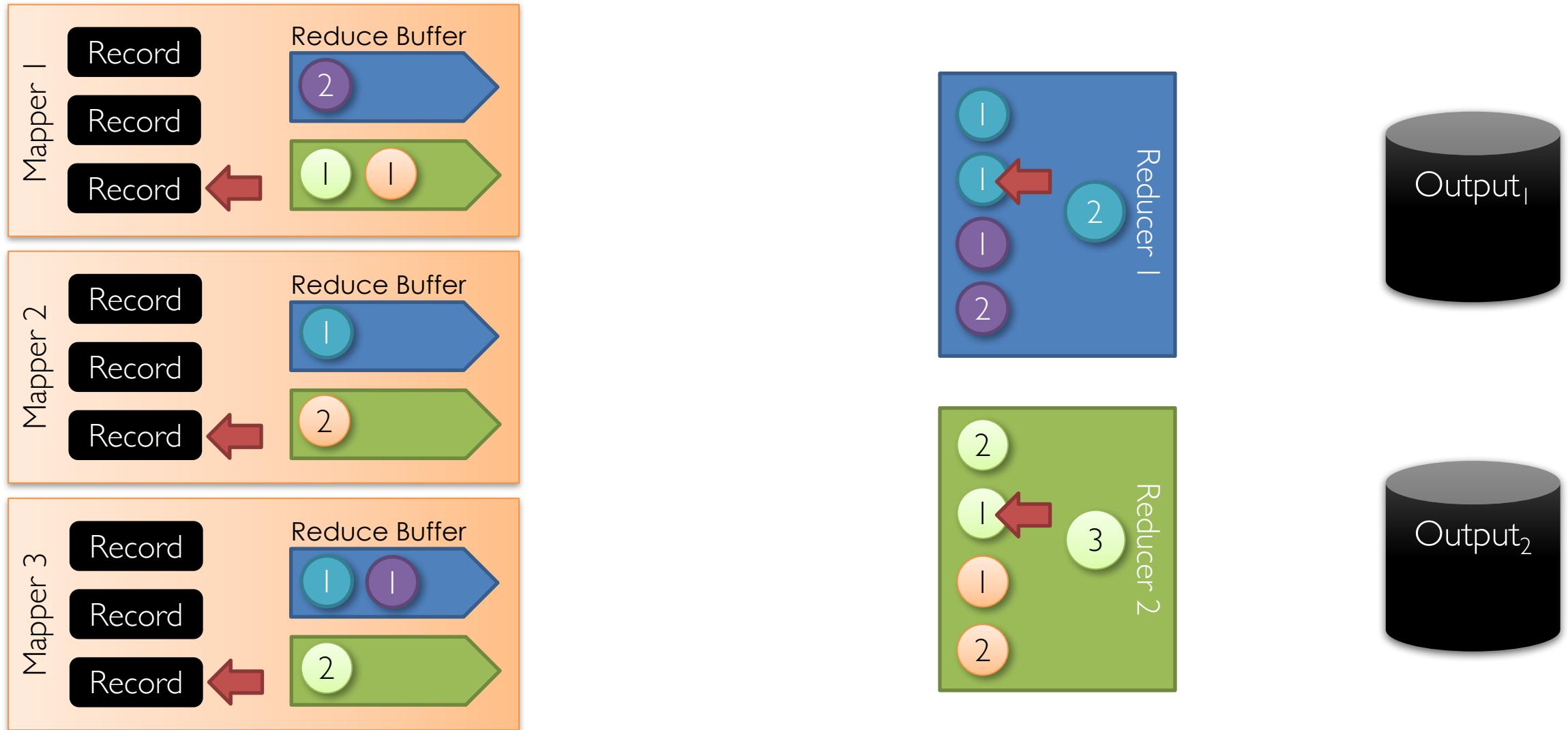
The Map Reduce System



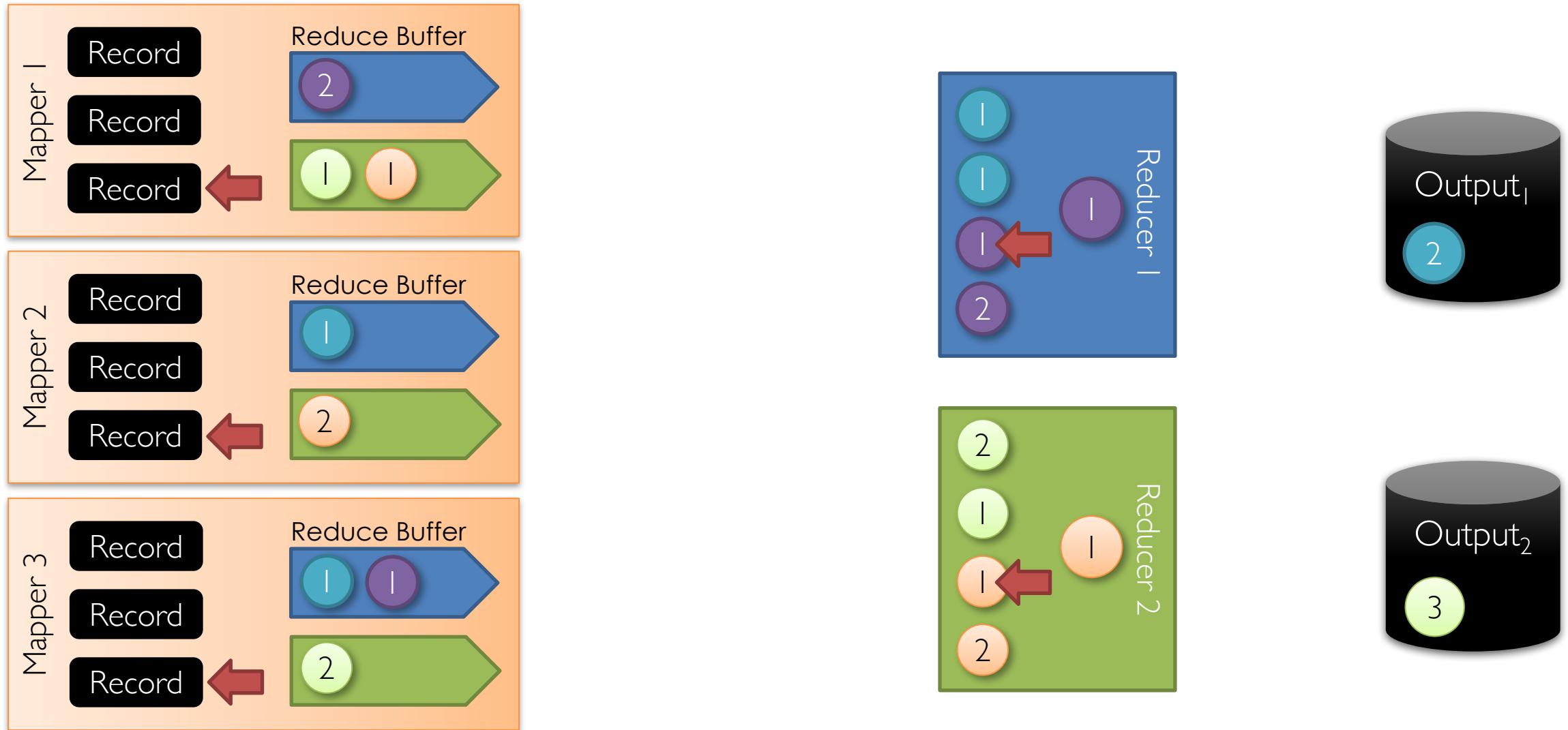
The Map Reduce System



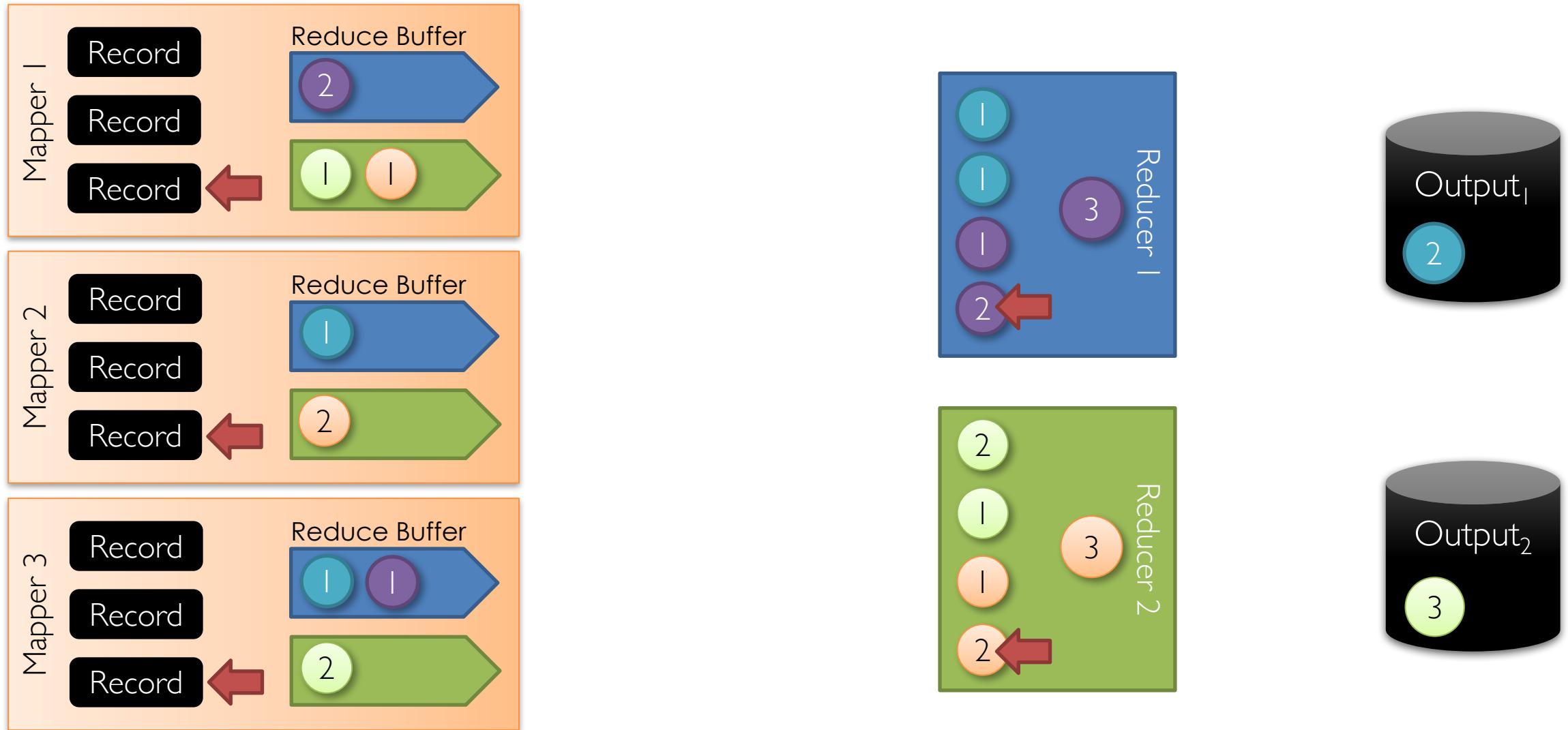
The Map Reduce System



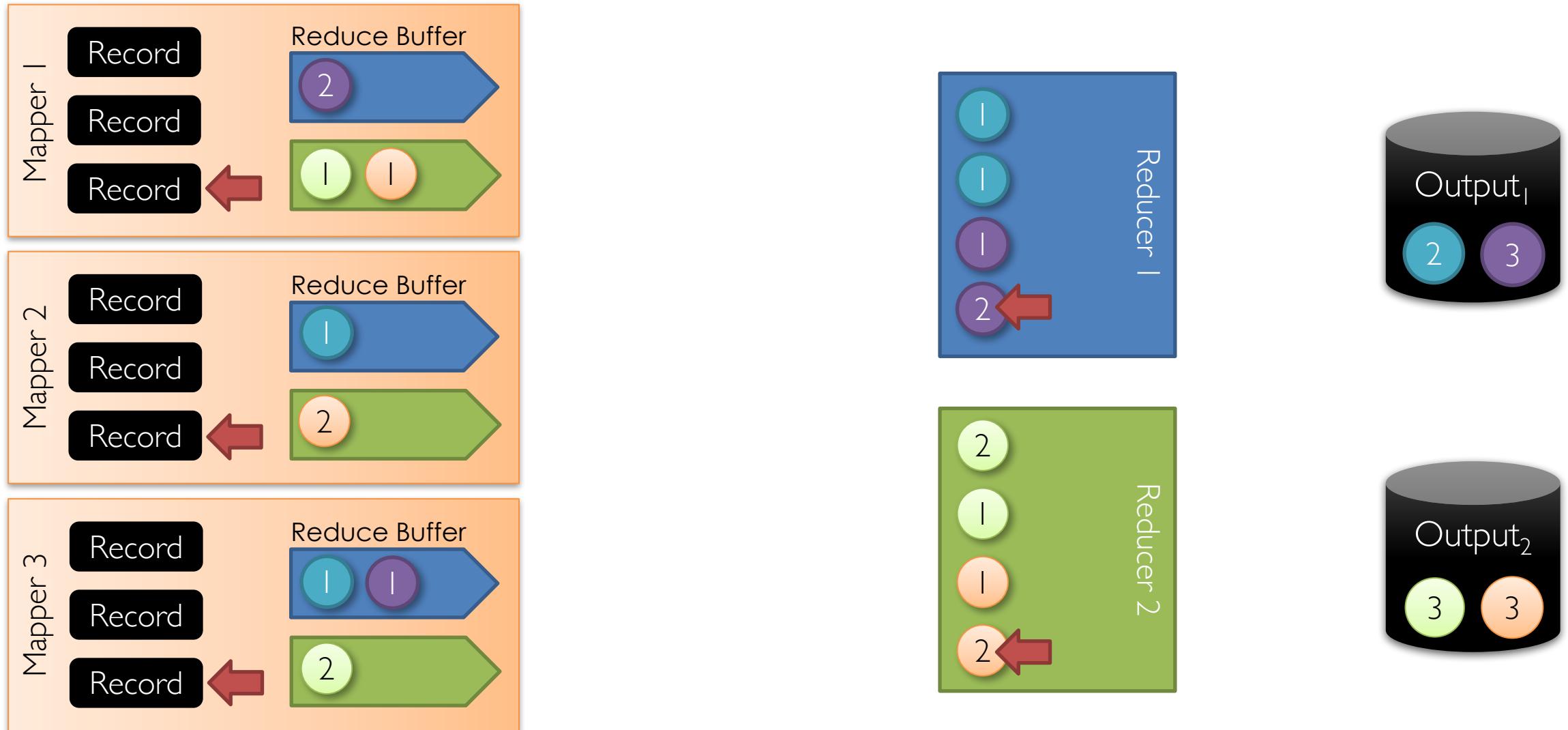
The Map Reduce System



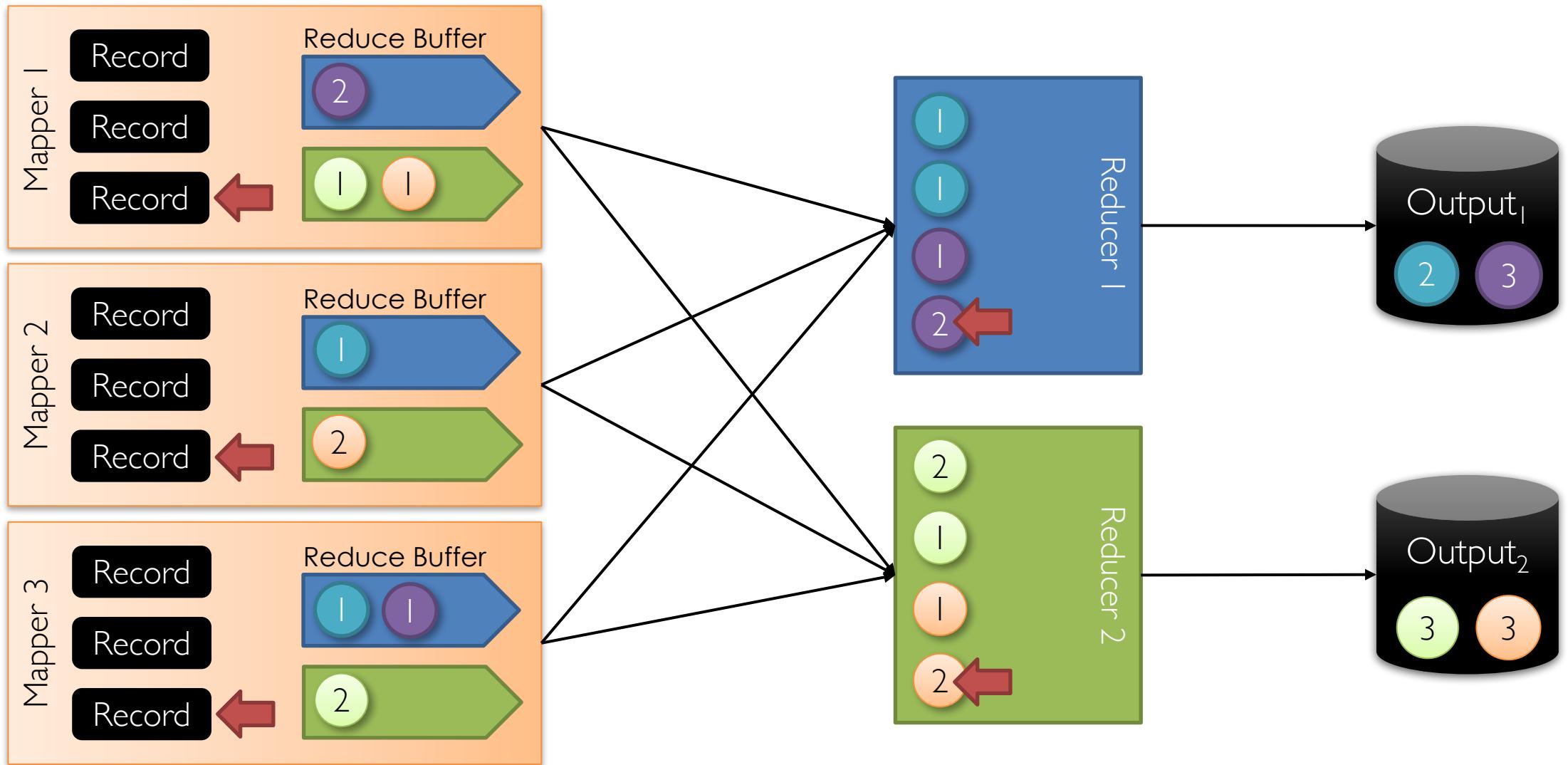
The Map Reduce System



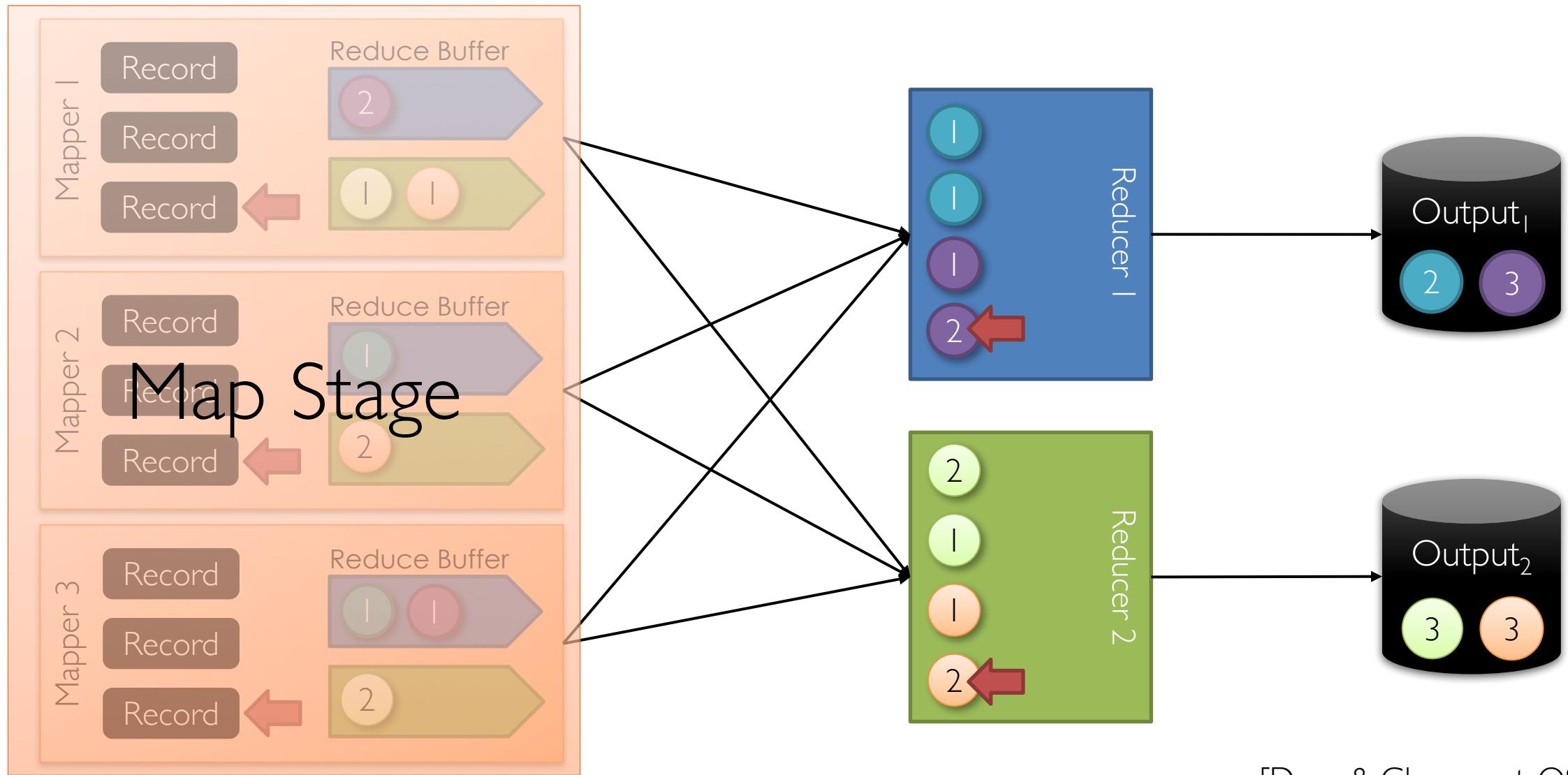
The Map Reduce System



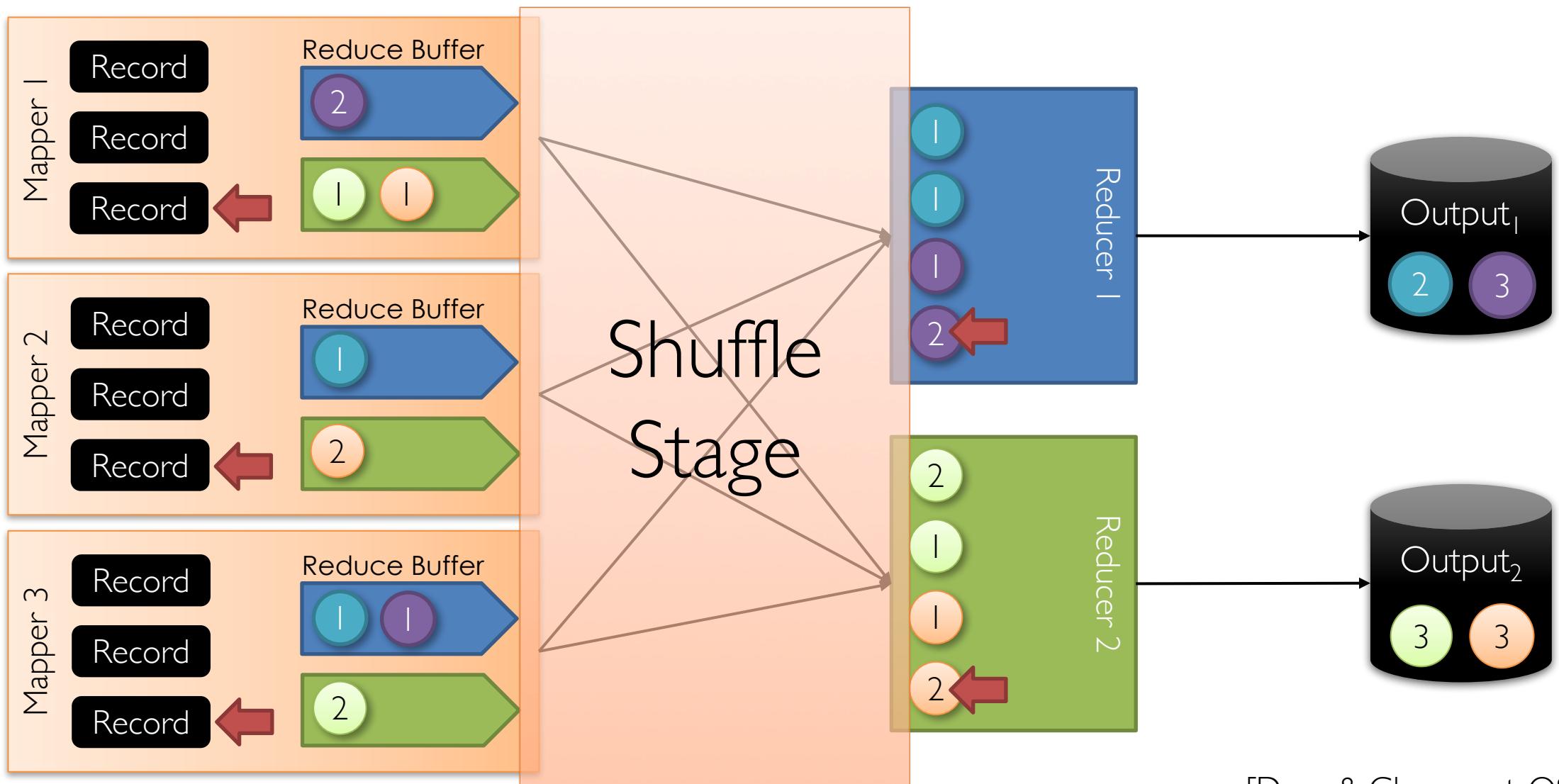
The Map Reduce System



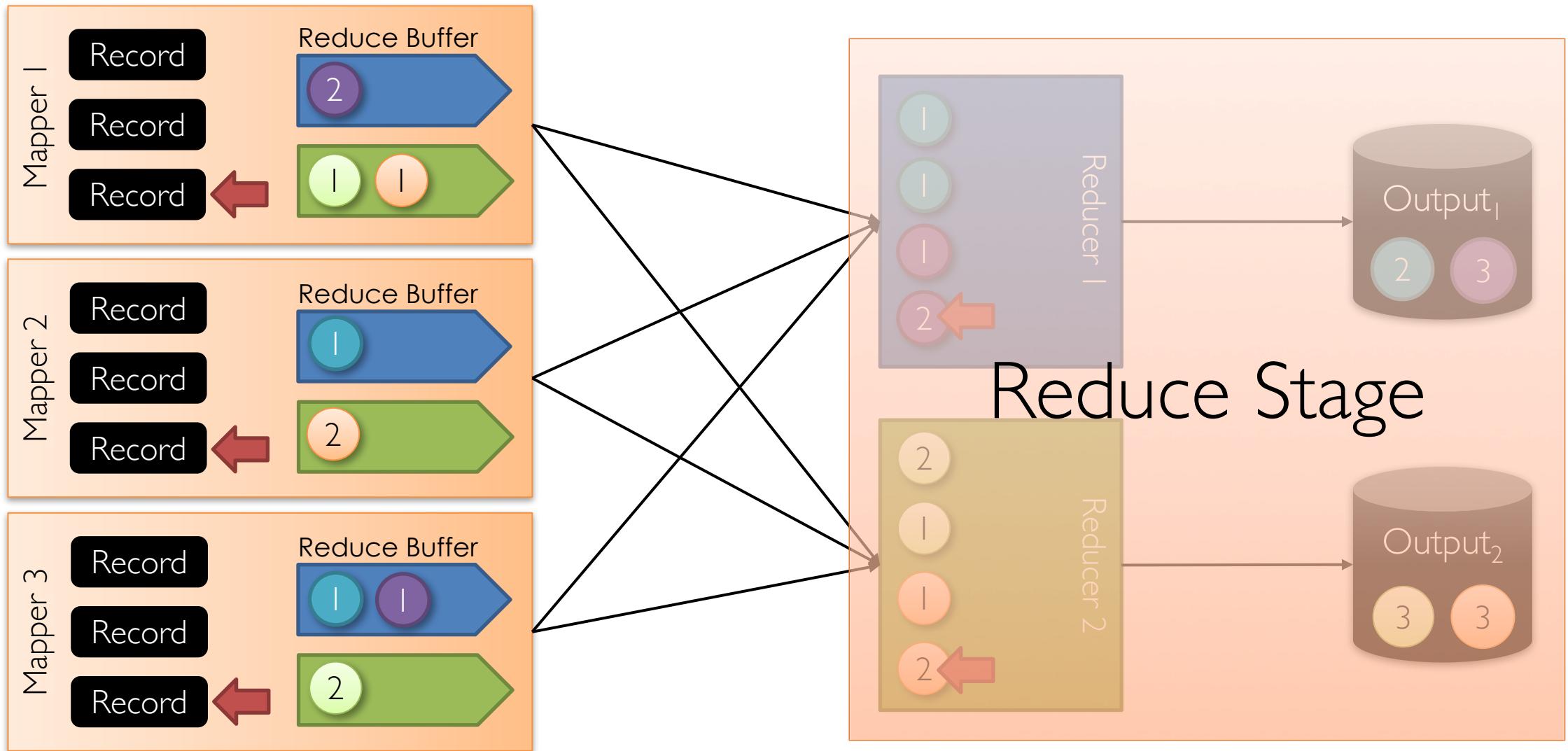
The Map Reduce System



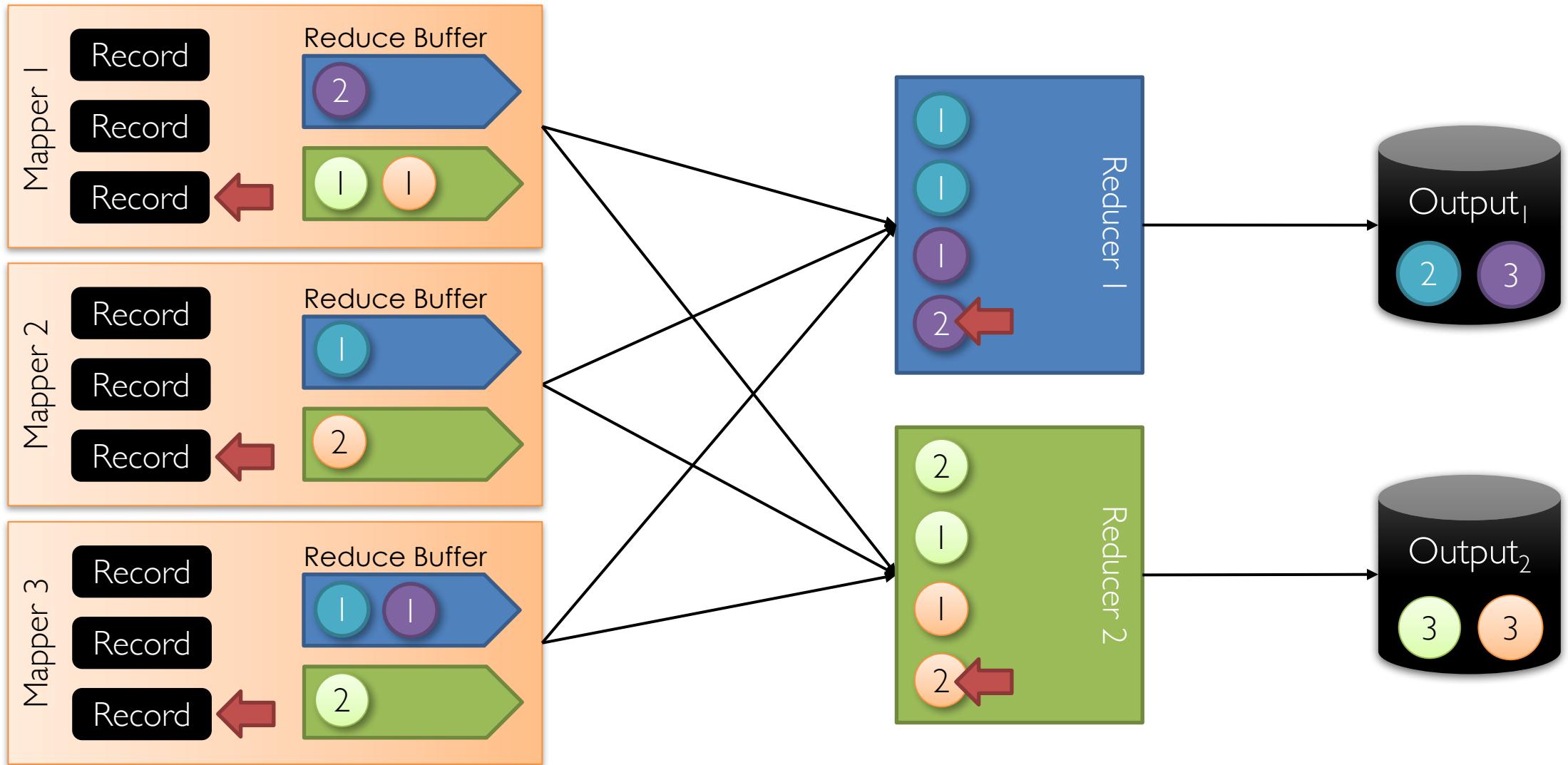
The Map Reduce System



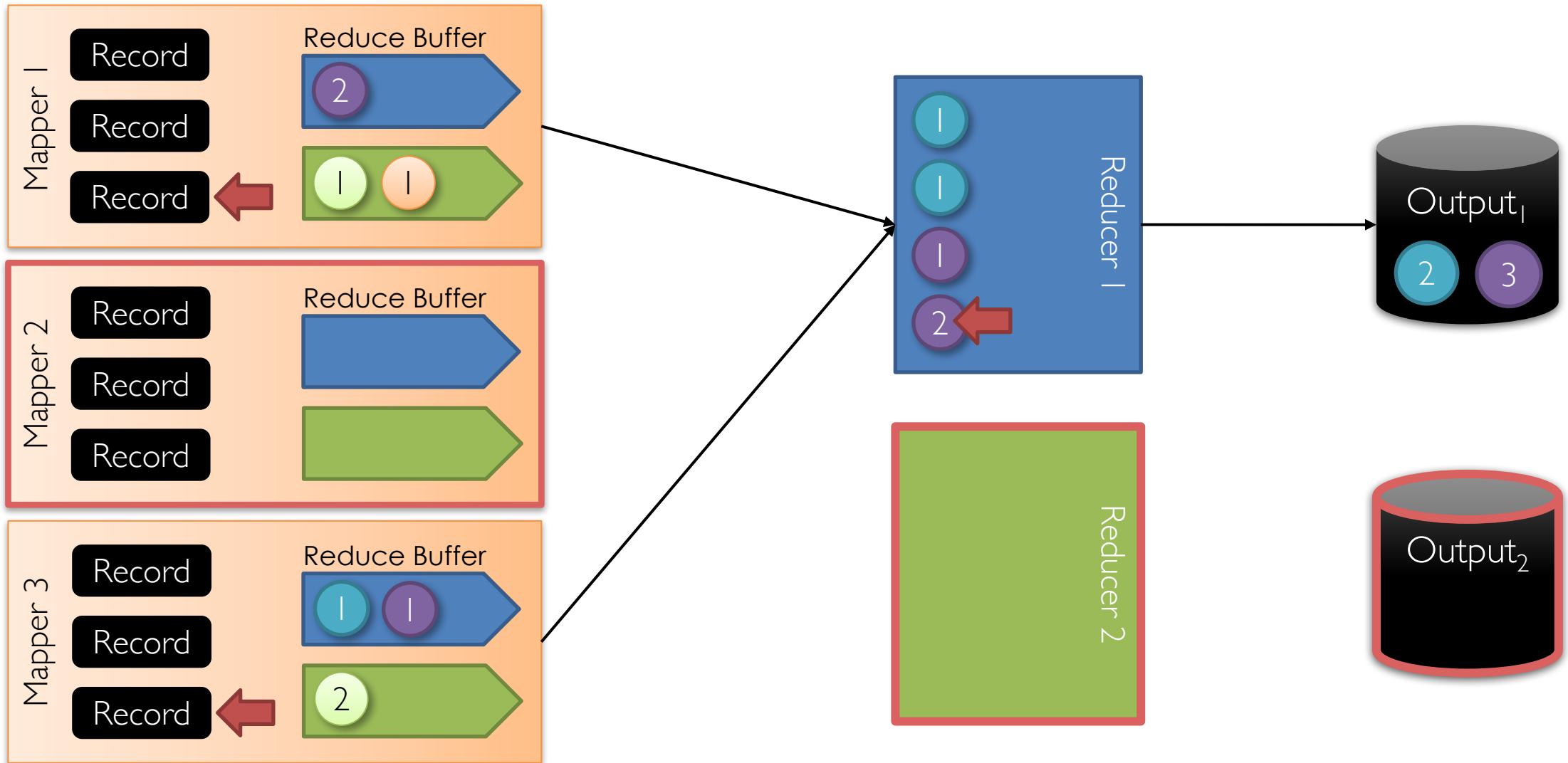
The Map Reduce System



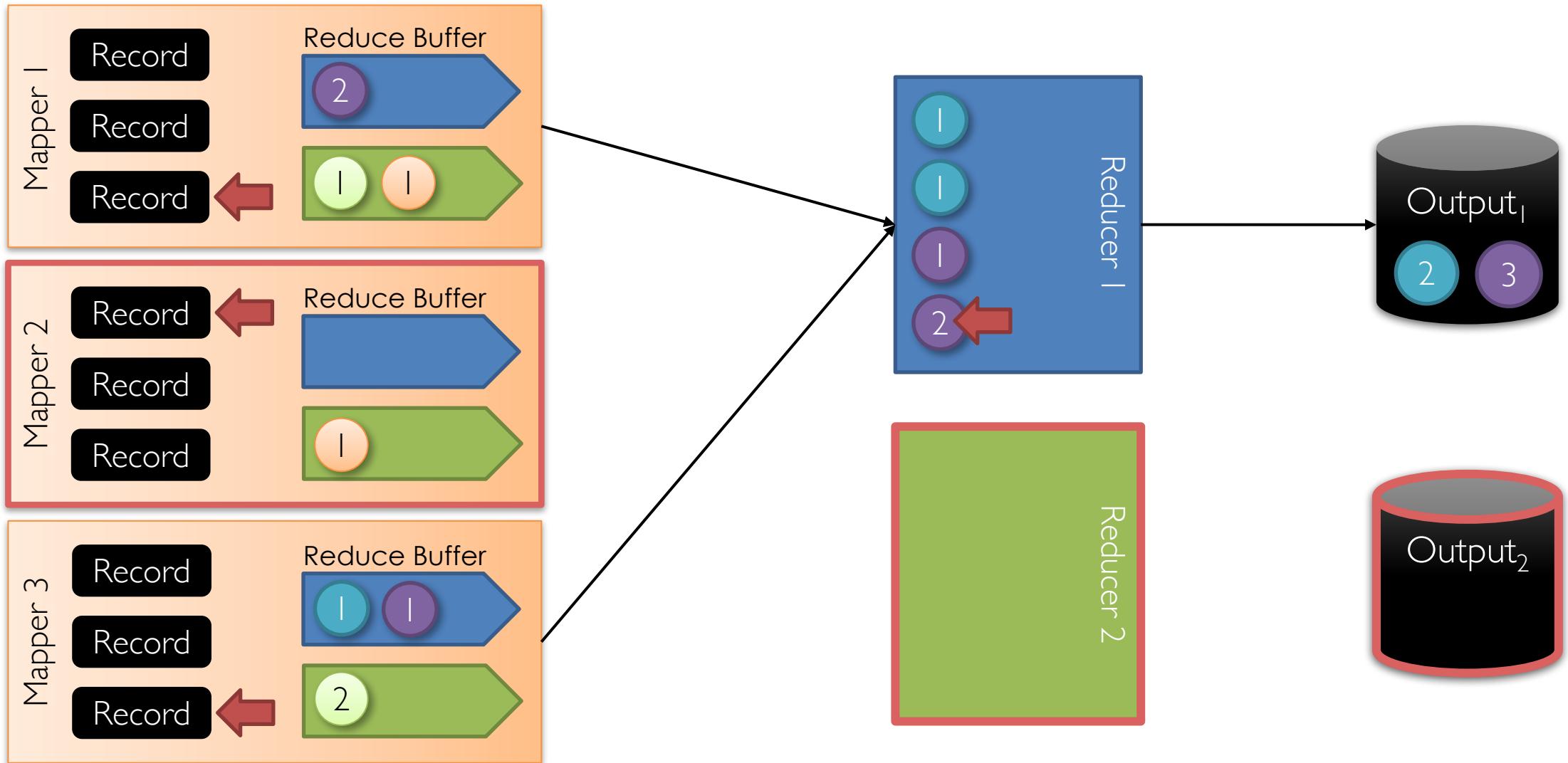
The Map Reduce: Fault Tolerance



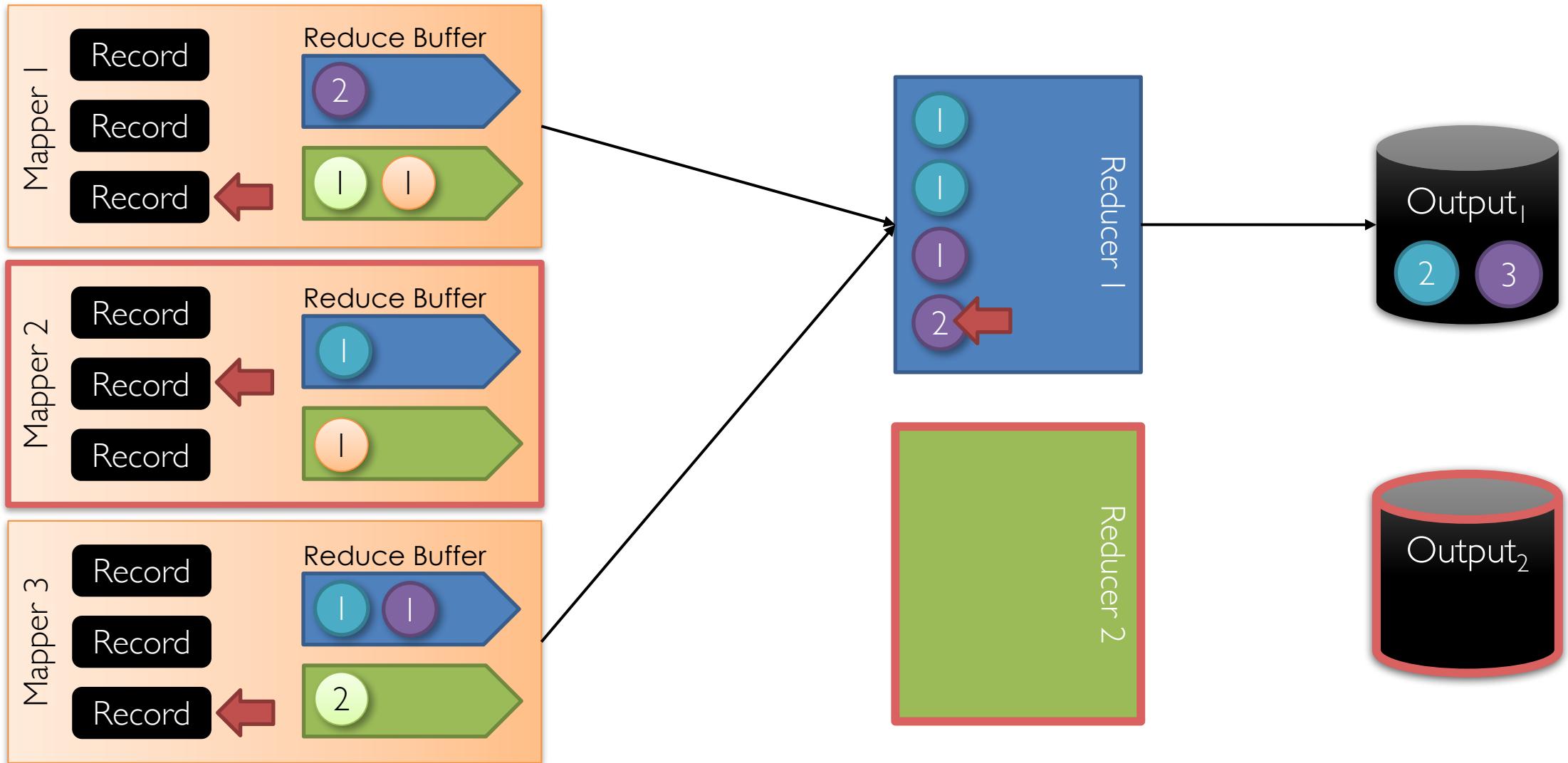
The Map Reduce: Fault Tolerance



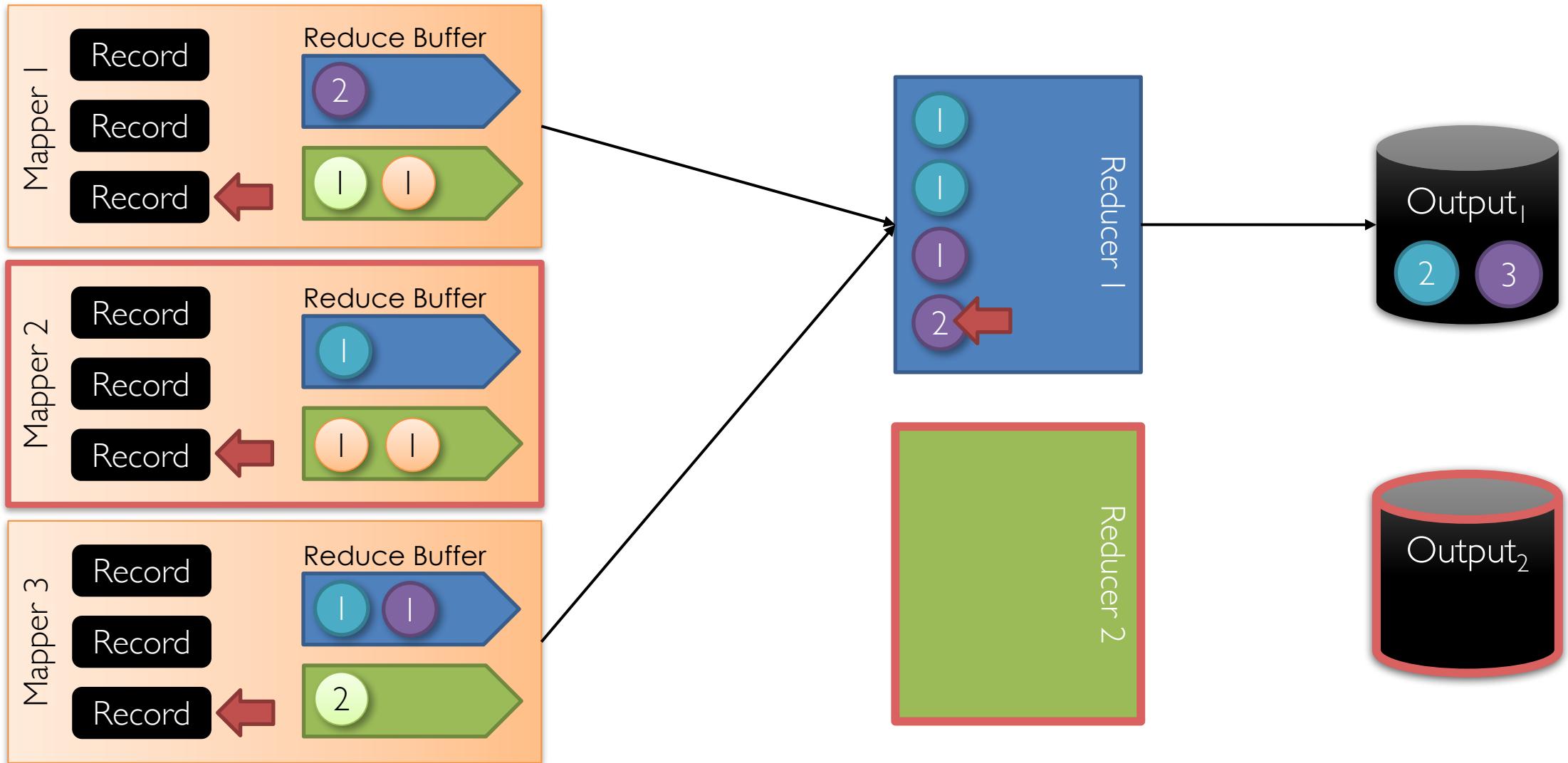
The Map Reduce: Fault Tolerance



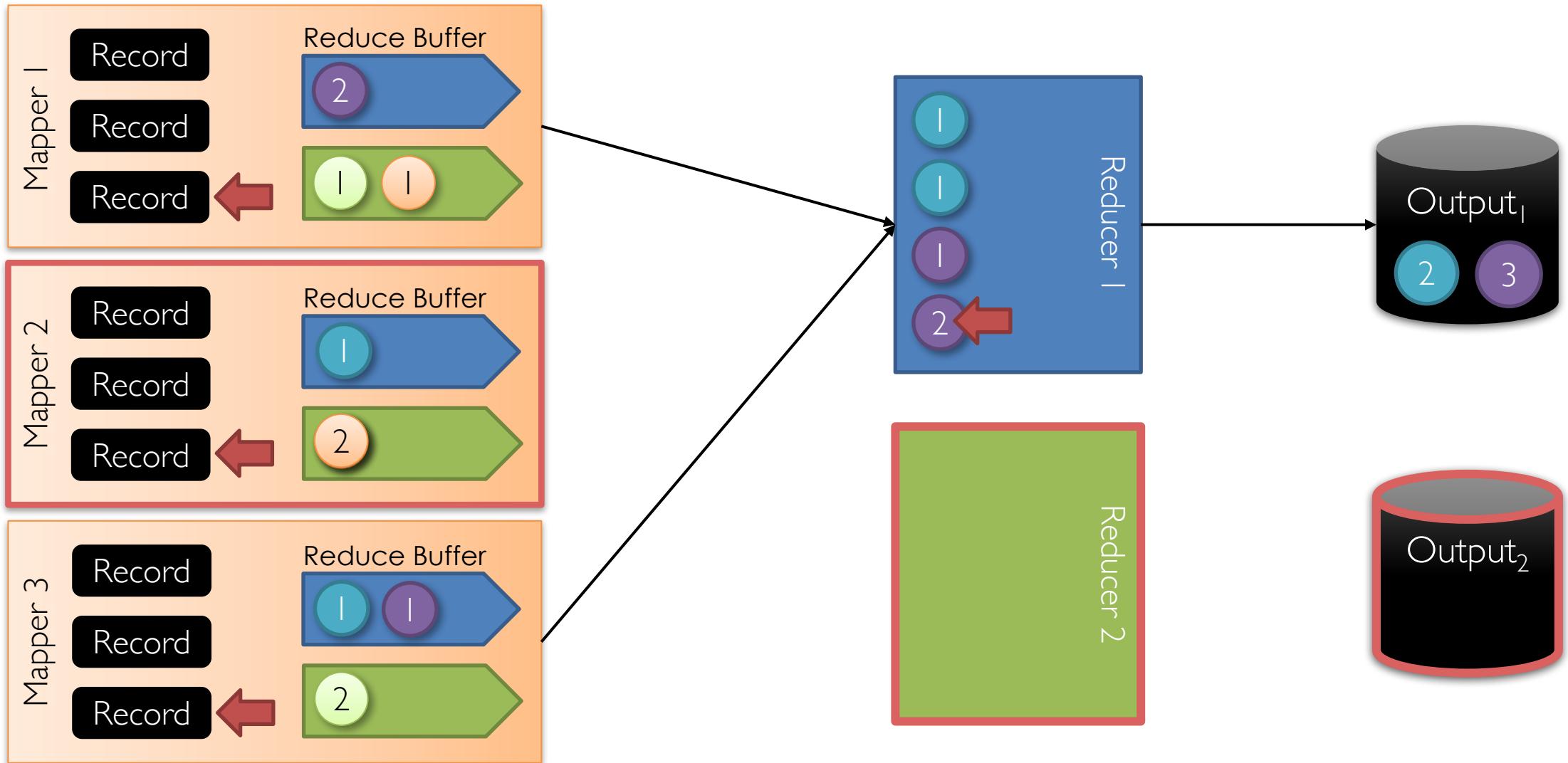
The Map Reduce: Fault Tolerance



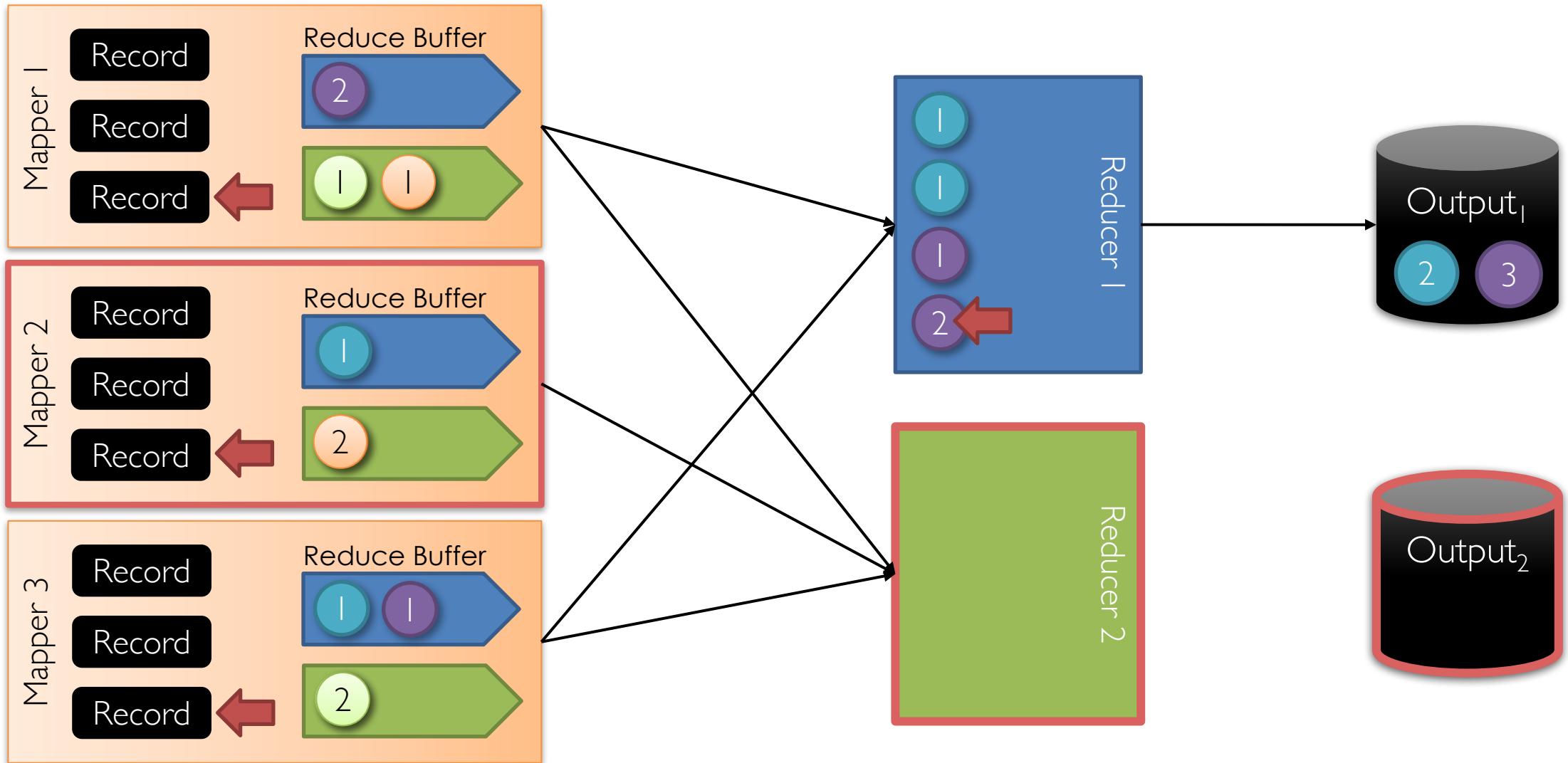
The Map Reduce: Fault Tolerance



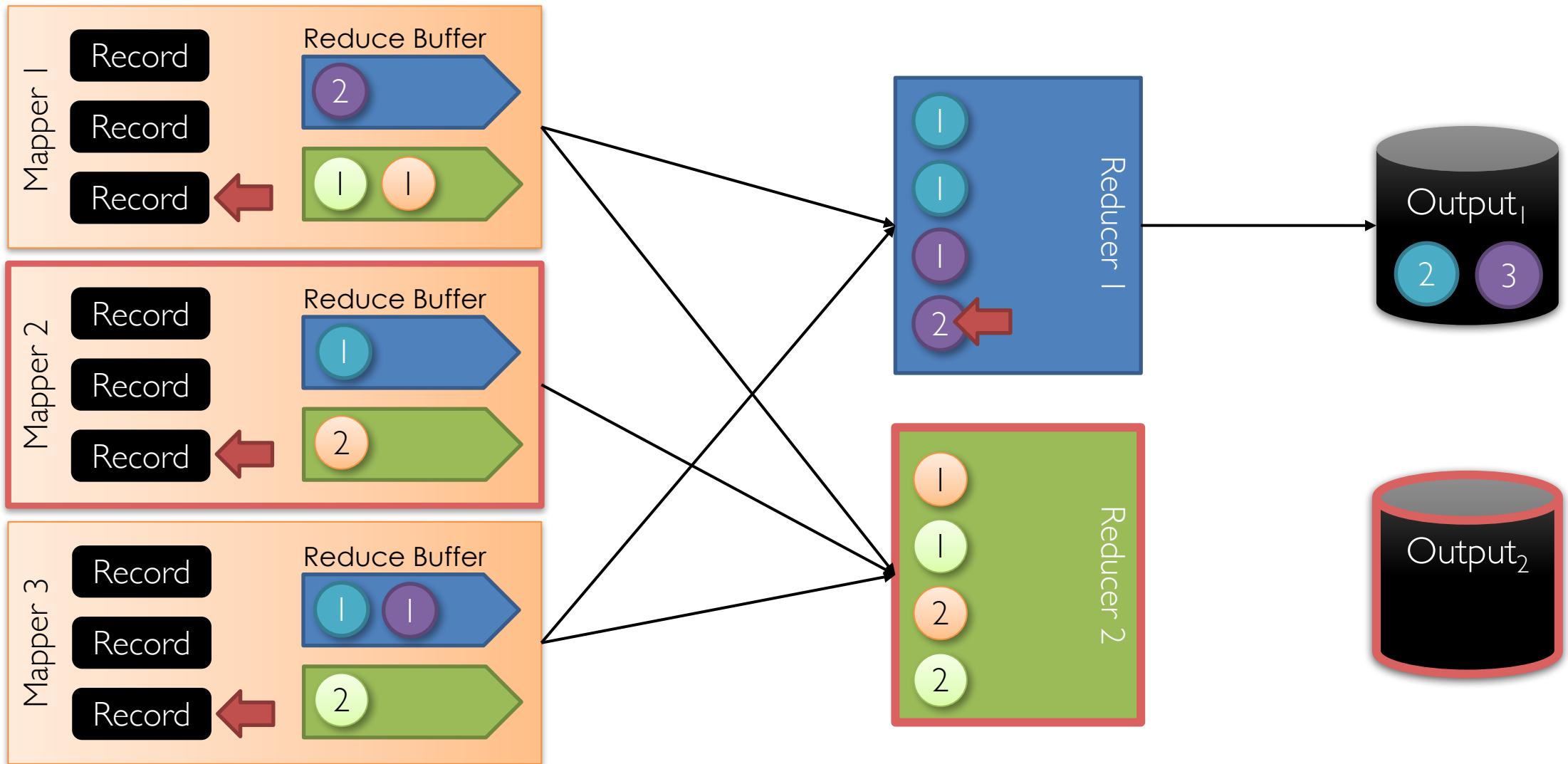
The Map Reduce: Fault Tolerance



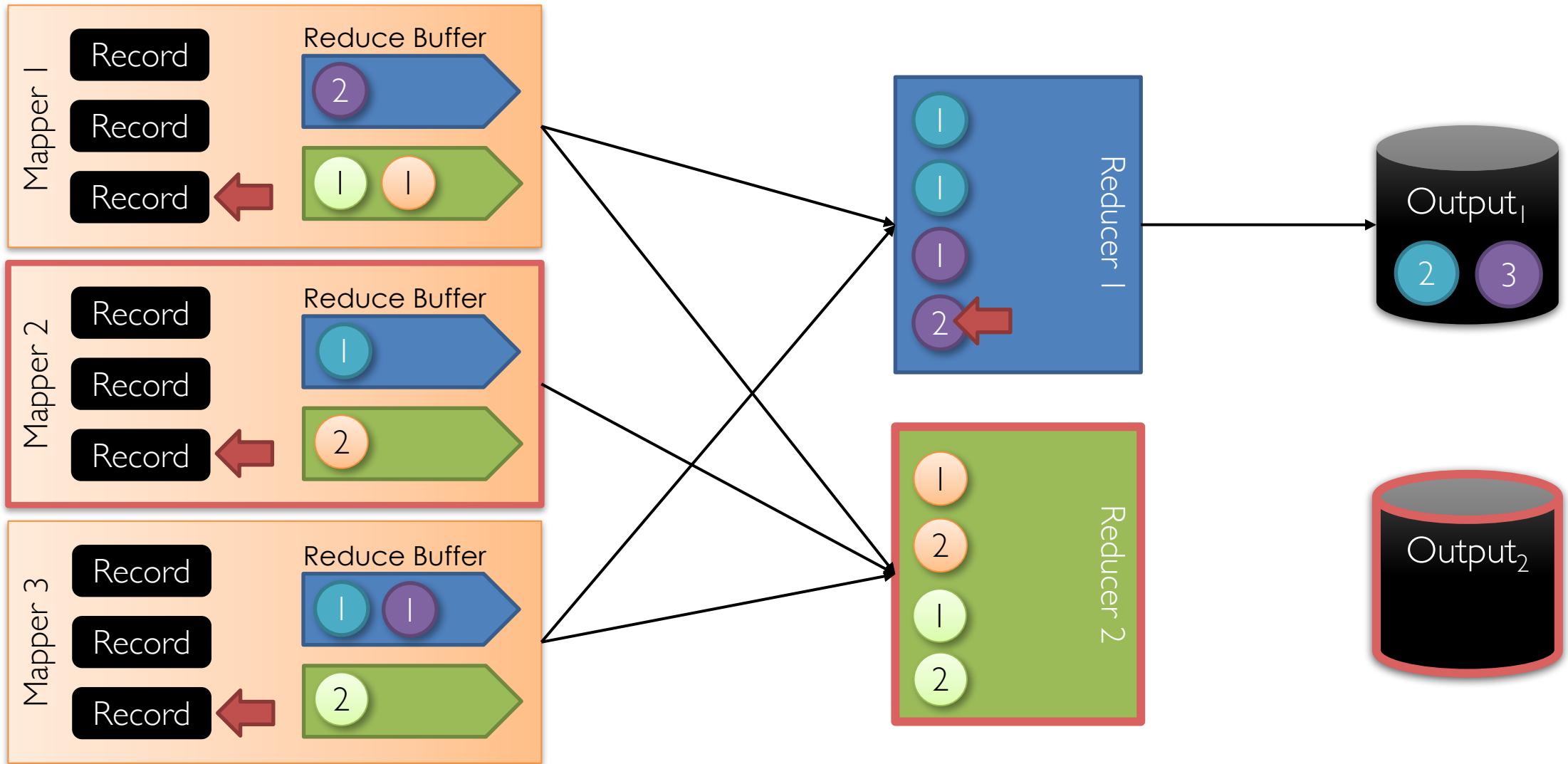
The Map Reduce: Fault Tolerance



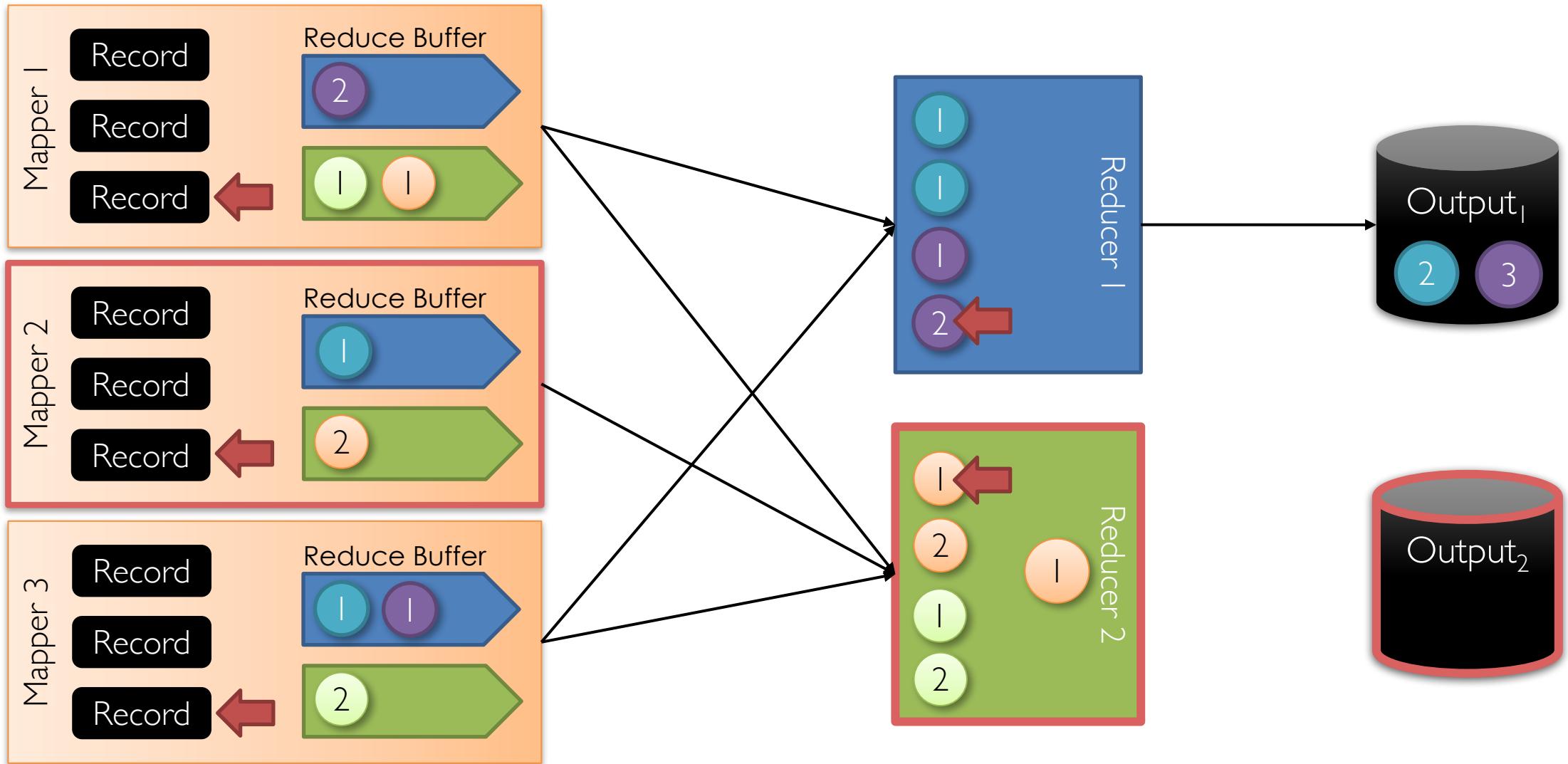
The Map Reduce: Fault Tolerance



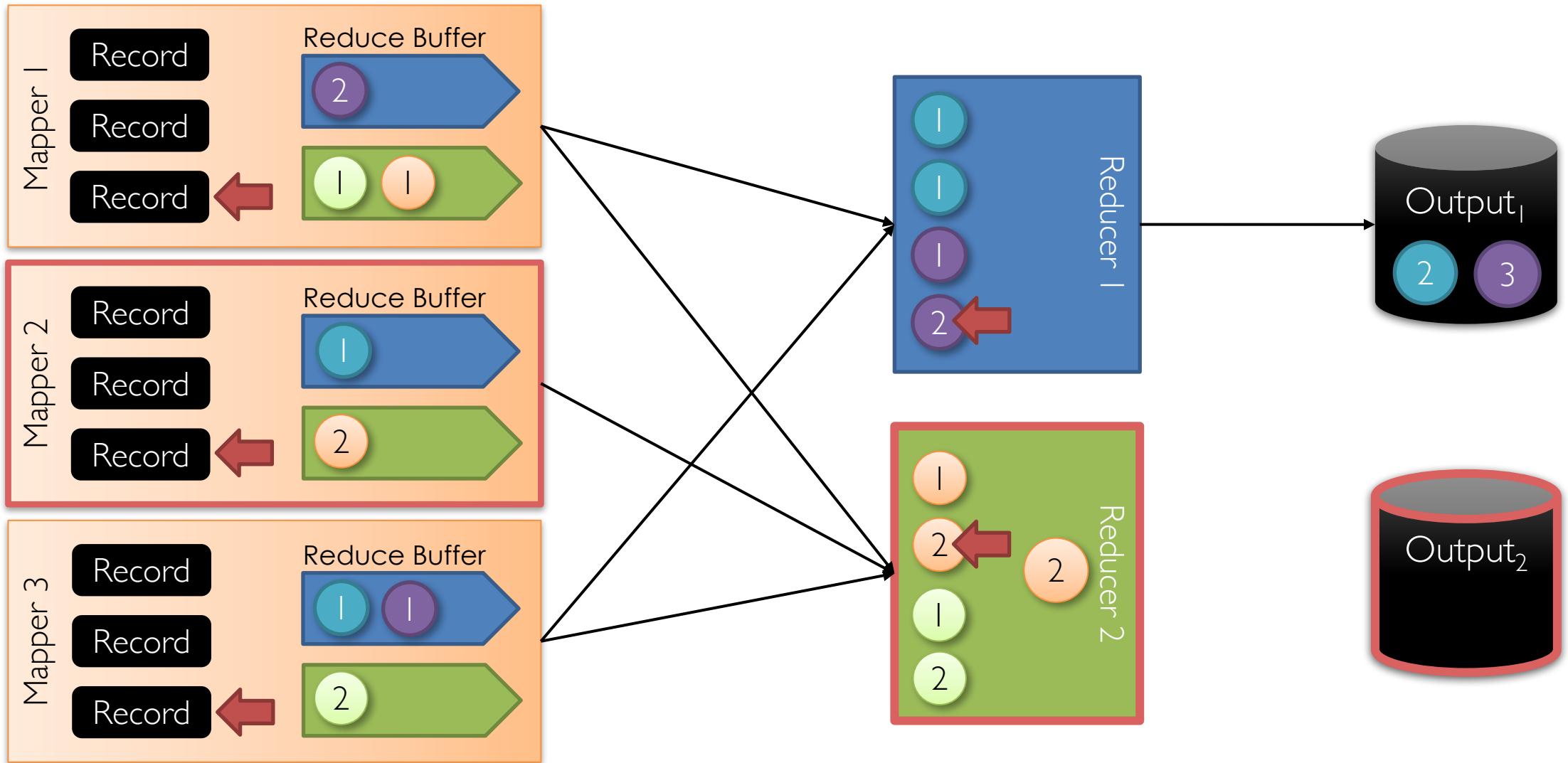
The Map Reduce: Fault Tolerance



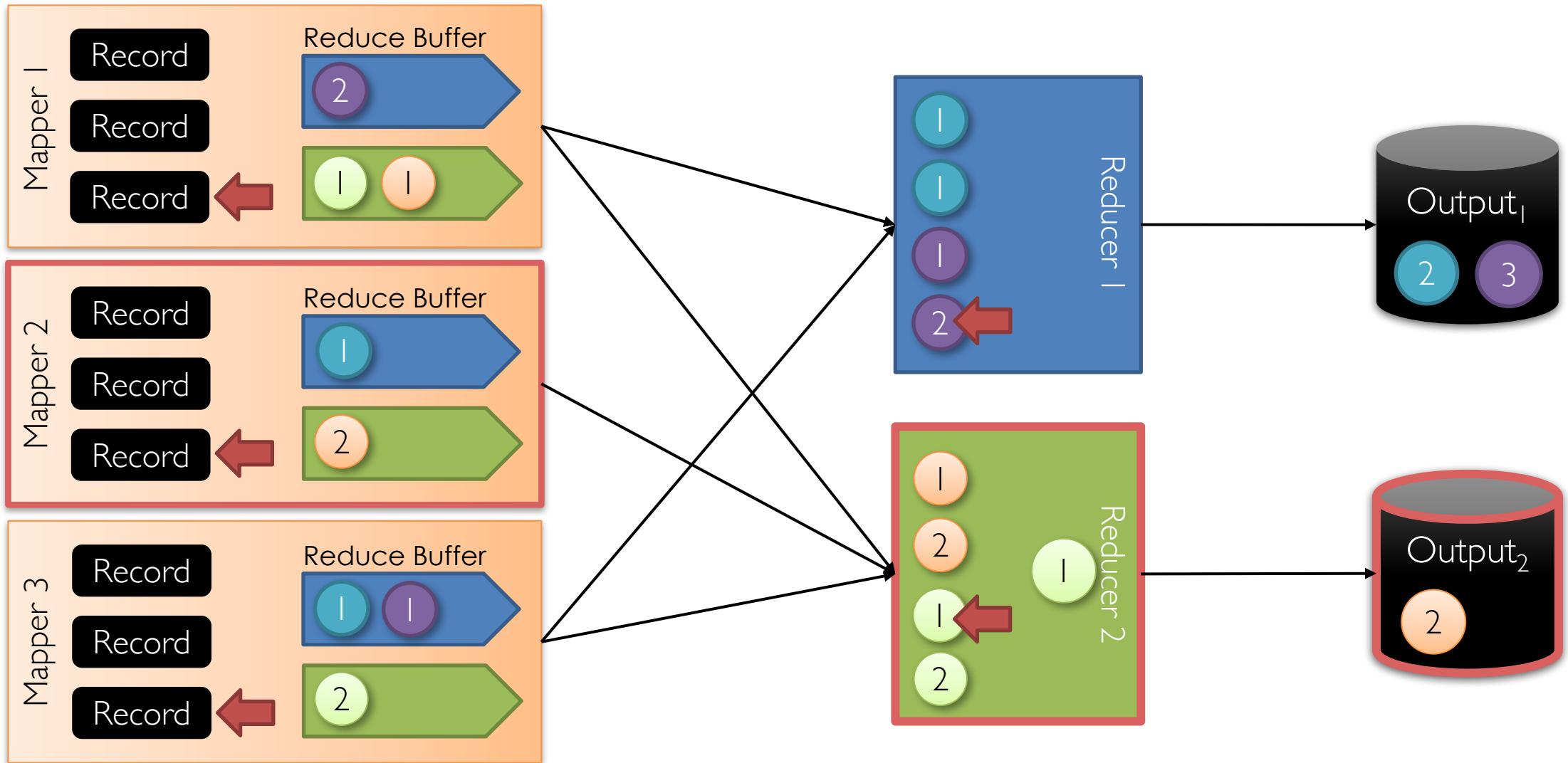
The Map Reduce: Fault Tolerance



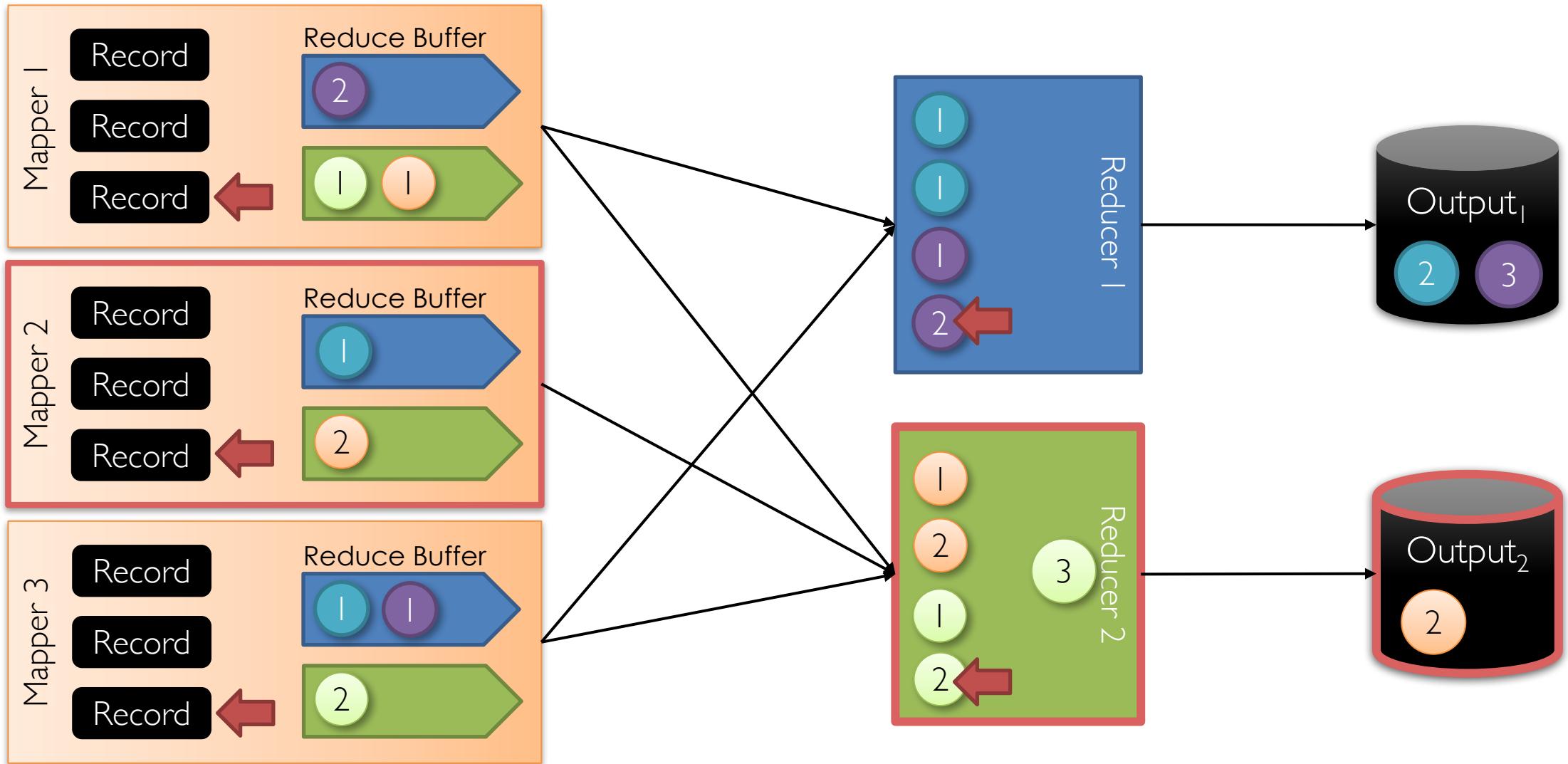
The Map Reduce: Fault Tolerance



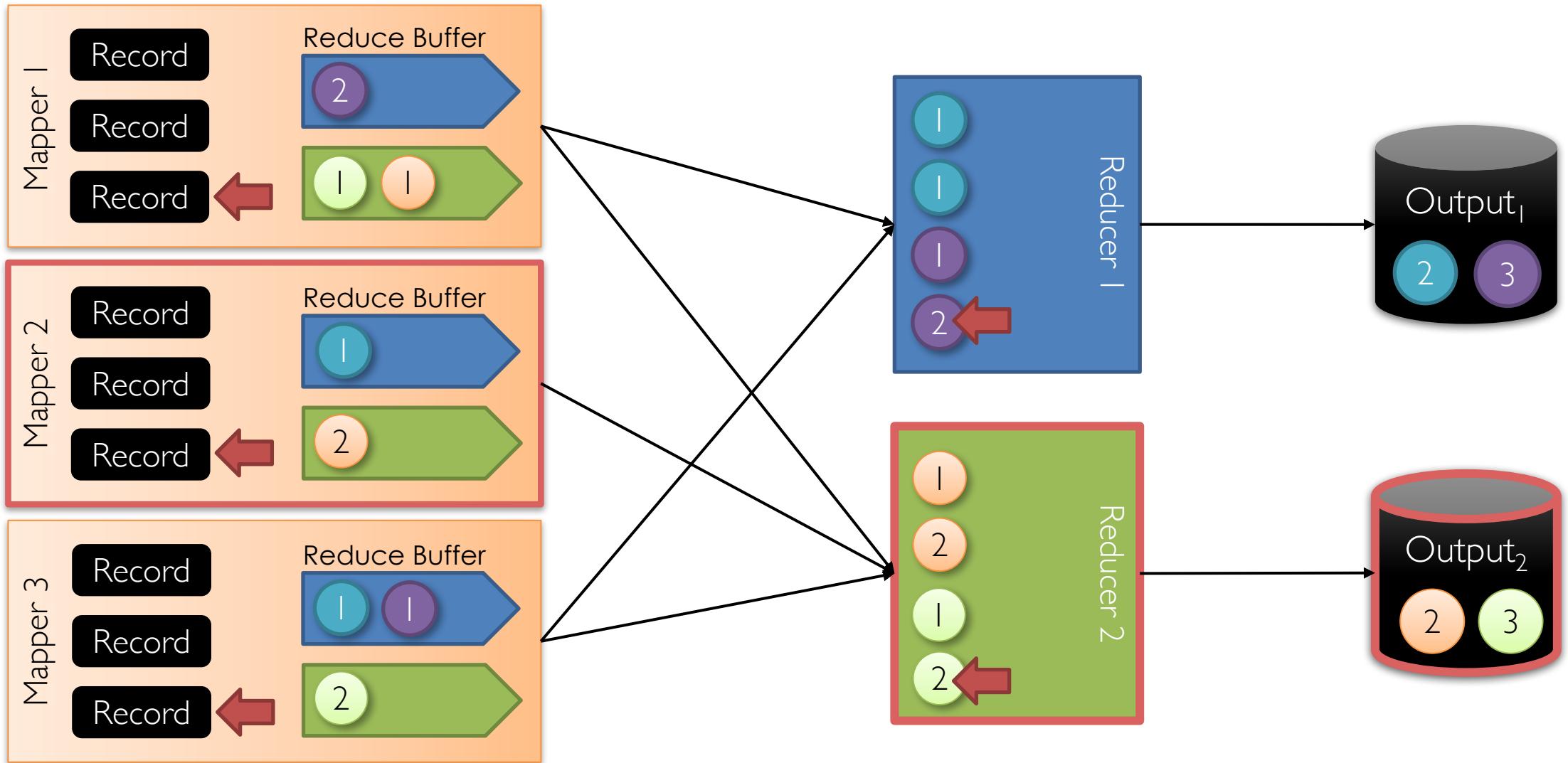
The Map Reduce: Fault Tolerance



The Map Reduce: Fault Tolerance



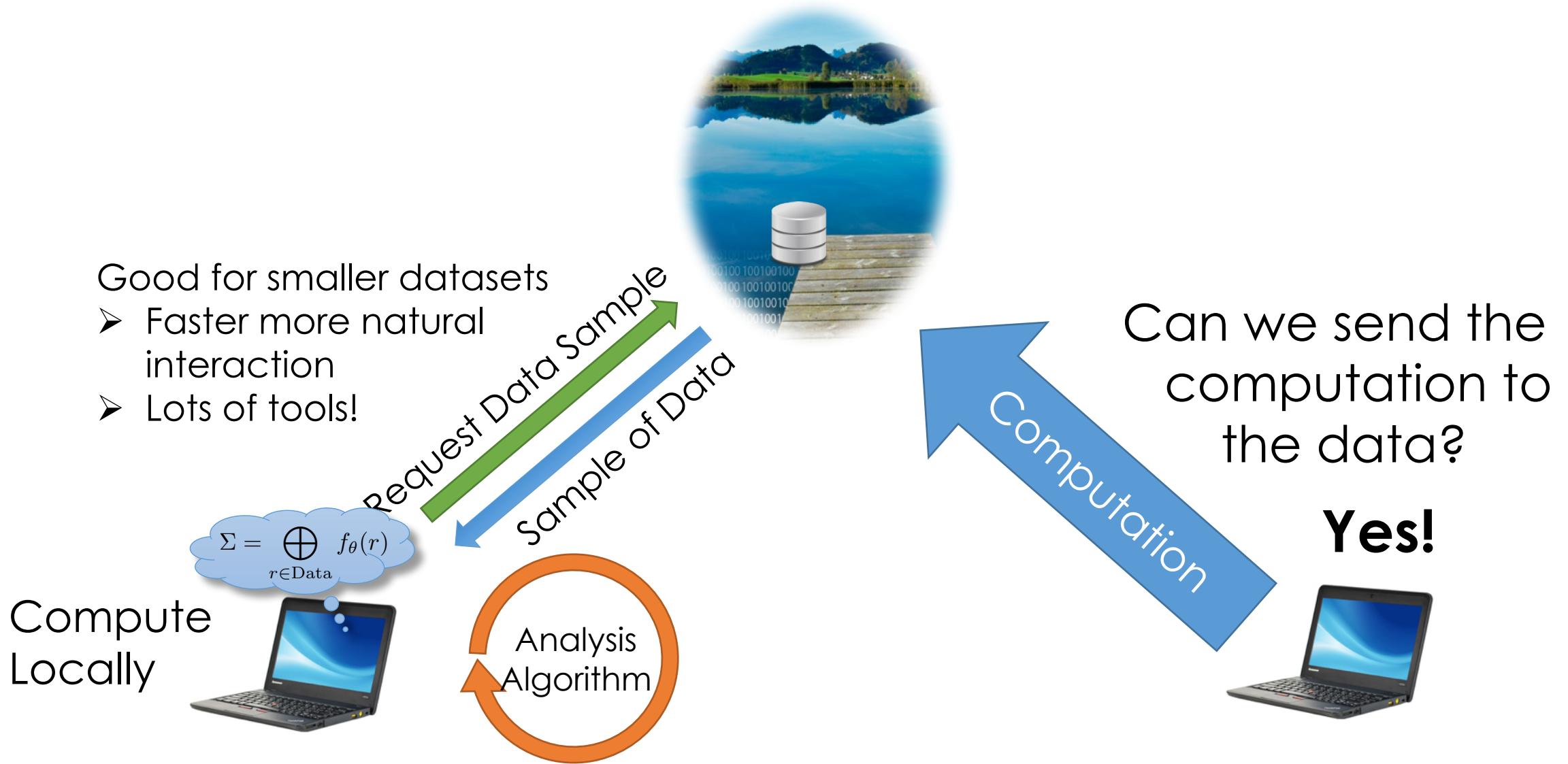
The Map Reduce: Fault Tolerance



Interacting with Data @ Scale

Map-Reduce

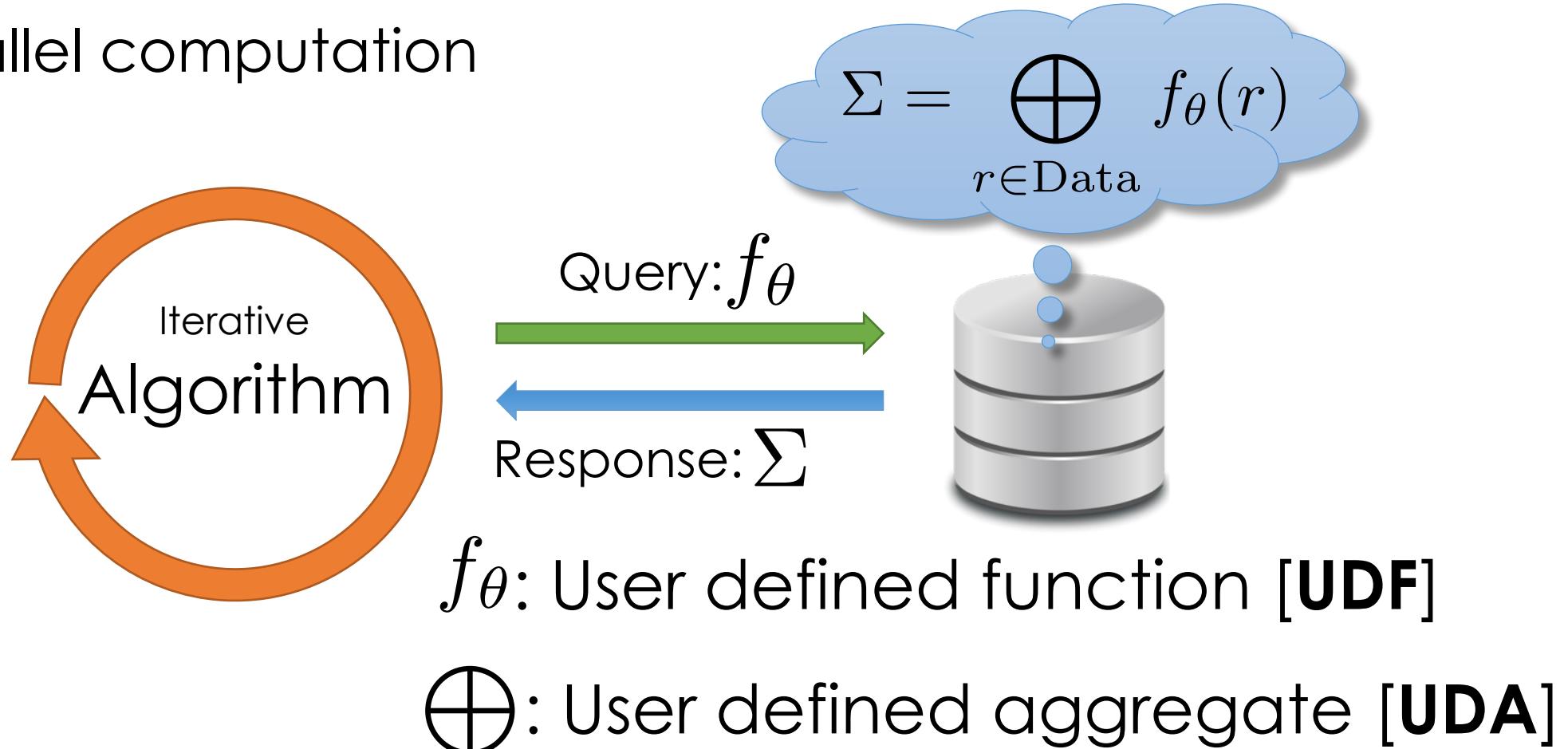
Interacting With the Data

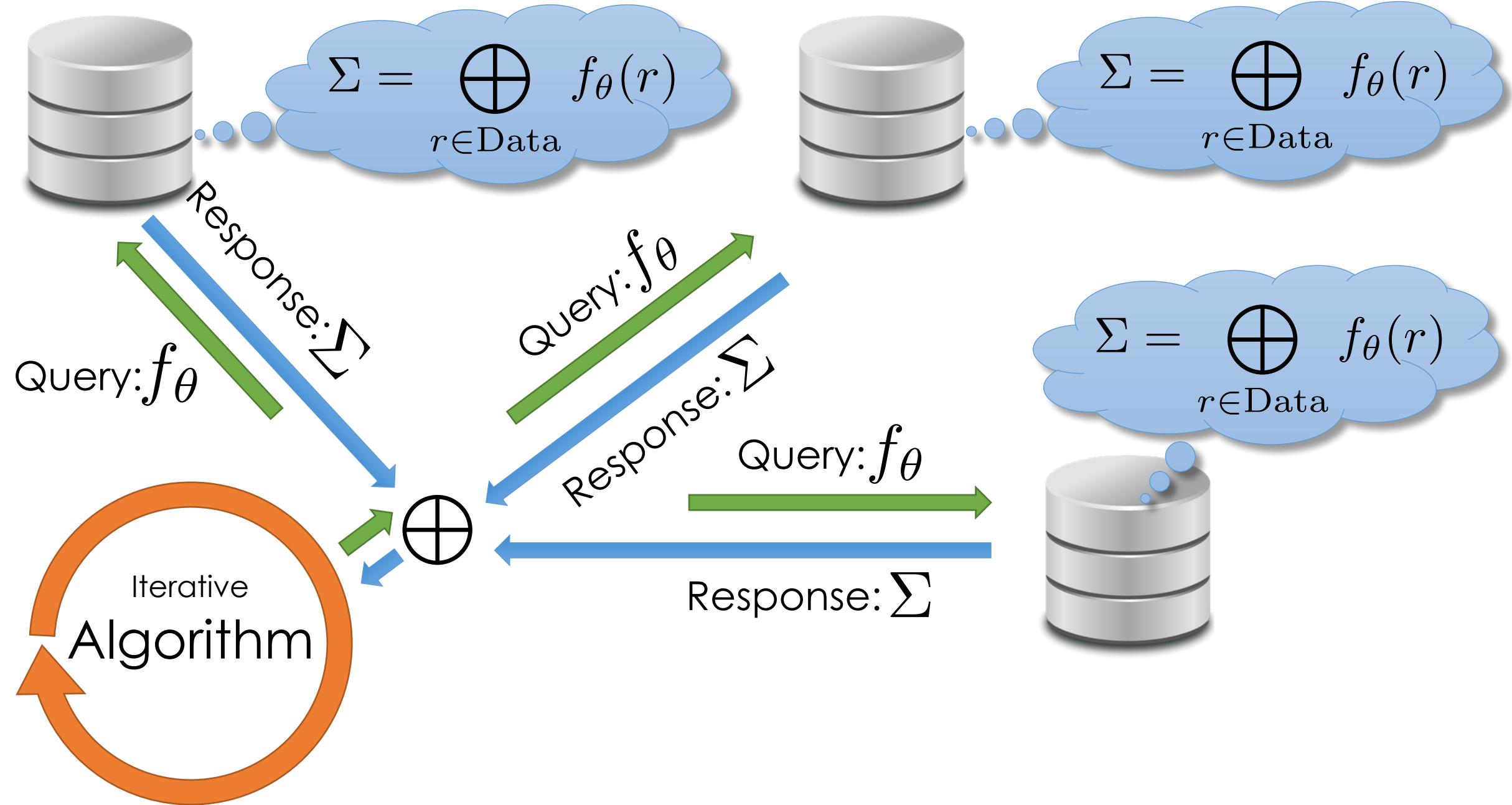


Statistical Query Pattern

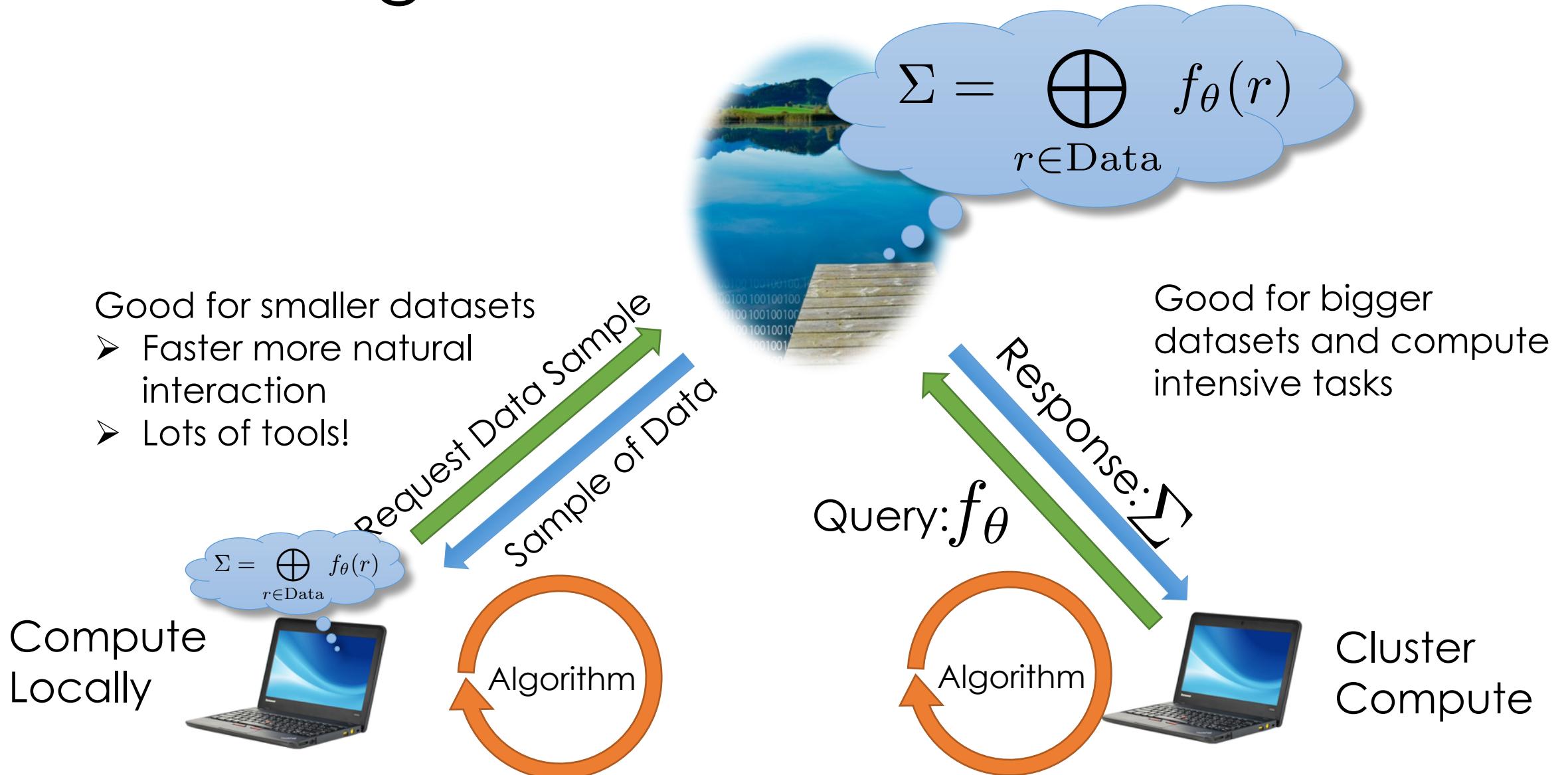
Common Machine Learning Pattern

- Computing aggregates of user defined functions
- Data-Parallel computation





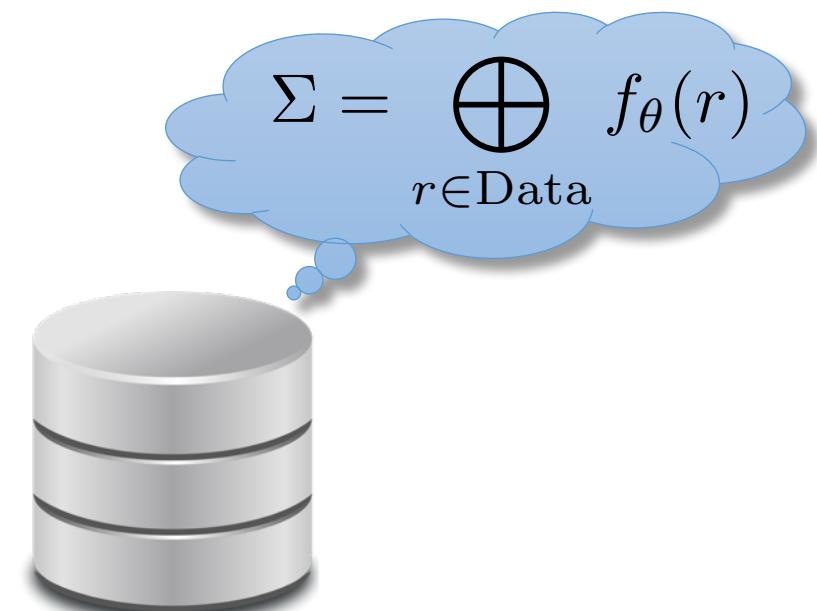
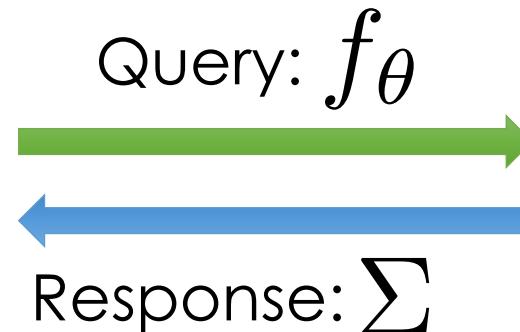
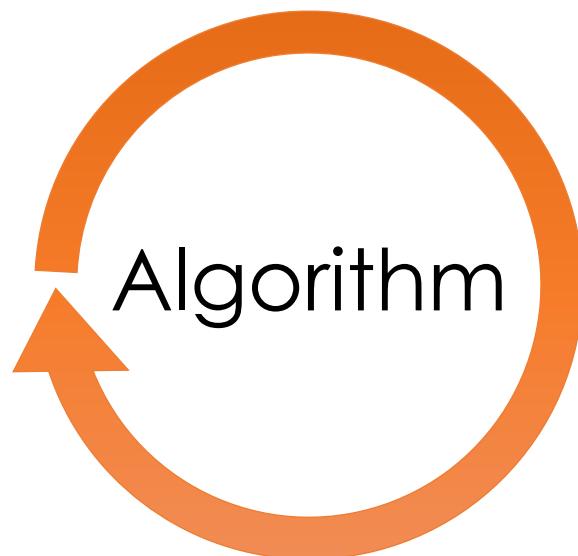
Interacting With the Data



Can we compute

$$\hat{\theta} = (X^T X)^{-1} X^T Y$$

using the statistical query pattern
in map-reduce?



Can we compute

$$\hat{\theta} = \underline{(X^T X)^{-1}} \underline{X^T Y}$$

using the statistical query pattern
in map-reduce?

Query 1

Query 2

Query: f_θ

$$\Sigma = \bigoplus_{r \in \text{Data}} f_\theta(r)$$

Break computation
into two queries



Computing Query 1

$$X^T X = \sum_{k=1}^n X_{k,1} X_{k,1}$$

$X^T X =$

$\begin{matrix} & & & & \\ & \text{---} & \text{---} & \vdots & \text{---} \\ p & x_1 & x_2 & \dots & x_n \\ & \text{---} & \text{---} & & \text{---} \\ & & & & n \end{matrix}$

$=$

$\begin{matrix} & & & & \\ & x_1 & - & & \\ & x_2 & - & & \\ & \dots & & & \\ & x_n & - & & \\ & \text{---} & \text{---} & \text{---} & \text{---} \end{matrix}$

$=$

$\begin{matrix} & & & & \\ & \text{---} & \text{---} & \text{---} & \text{---} \\ p & & & & p \\ & \text{---} & \text{---} & \text{---} & \text{---} \end{matrix}$

Computing Query 1

$$X^T X = \sum_{k=1}^n X_{k,i} X_{k,j}$$

Computing Query 1

$$X^T X = \underbrace{\begin{pmatrix} | & | & | \\ x_1 & x_2 & \vdots & x_n \\ | & | & | \end{pmatrix}}_{n} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \sum_{k=1}^n X_{k,i} X_{k,j}$$

p p (i,j)

```
def map_xtx(x, y):
    p = len(x)
    for i in range(p):
        for j in range(p):
            yield ([i,j], x[i] * x[j])
```

```
def sum(k, values):
    return (k, np.sum(values))
```

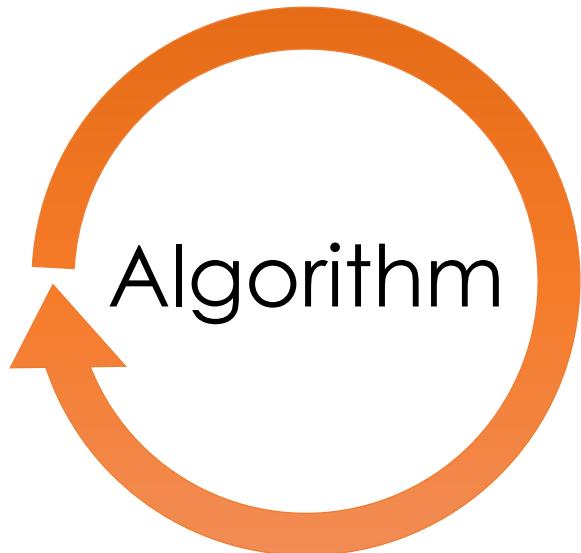
Computing Query 2 ($X^T Y$)

```
def map_xty(x, y):
    for i in range(len(x)):
        yield (i, x[i] * y[i])
```

```
def sum(k, values):
    return (k, np.sum(values))
```

```
def map_xtx(x, y):  
    p = len(x)  
    for i in range(p):  
        for j in range(p):  
            yield ([i,j], x[i] * x[j])
```

$O(nd^2)$
#mappers



$O(d^3)$

```
def map_xty(x, y):  
    for i in range(len(x)):  
        yield (i, x[i] * y[i])
```

```
def sum(k, values):  
    return (k, np.sum(values))
```

Run Locally:

```
XtX_pairs = data.map(map_xtx).reduce(sum)  
XtX = np.zeros(p,p)  
for ((i,j), v) in XtX_pairs: XtX[i,j] = v
```

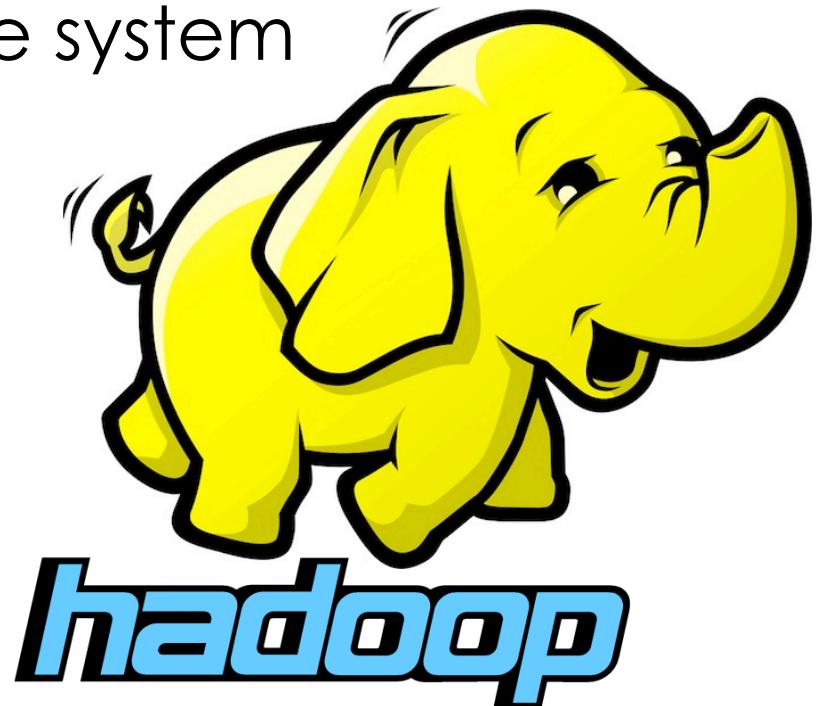
```
XtY_pairs = data.map(map_xty).reduce(sum)  
XtY = np.zeros(p,1)  
for (i,v) in XtY_pairs: XtY[i] = v
```

```
theta = np.linalg.solve(XtX, XtY)
```

Map Reduce Technologies

Hadoop

- First open-source map-reduce software system
 - Managed by Apache foundation
- Based on Google's
 - Map-Reduce
 - Google File System
- Several key technologies
 - **HDFS:** Hadoop File System
 - **Yarn:** Yet another resource negotiator
 - **MapReduce:** map-reduce compute framework





In-Memory Dataflow System

Developed at the UC Berkeley AMP Lab

M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. *Spark: cluster computing with working sets*. HotCloud'10

M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M.J. Franklin, S. Shenker, I. Stoica. *Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing*, NSDI 2012

What Is Spark

- Parallel execution engine for big data processing
- **General**: efficient support for multiple workloads
- **Easy** to use: 2-5x less code than Hadoop MR
 - High level API's in Python, Java, and Scala
- **Fast**: up to 100x faster than Hadoop MR
 - Can exploit in-memory when available
 - Low overhead scheduling, optimized engine

Spark Programming Abstraction

- Write programs *in terms of transformations on distributed datasets*
- Resilient Distributed Datasets (**RDDs**)
 - Distributed **collections of objects** that can be stored **in memory** or **on disk**
 - Built via parallel transformations (map, filter, ...)
 - Automatically rebuilt on failure

General

- Unifies **batch, interactive, streaming** workloads
- Easy to build sophisticated applications
 - Support iterative, graph-parallel algorithms
 - Powerful APIs in Scala, Python, Java, R



SparkSQL

Spark
Streaming

Mlib

GraphX

SparkR

Spark Core

Easy to Write Code

```
1 public class WordCount {
2     public static class TokenizerMapper
3         extends Mapper<Object, Text, Text, IntWritable>
4     {
5         private final static IntWritable one = new IntWritable(1);
6         private Text word = new Text();
7
8         public void map(Object key, Text value, Context context
9             throws IOException, InterruptedException {
10             StringTokenizer itr = new StringTokenizer(value.toString());
11             while (itr.hasMoreTokens()) {
12                 word.set(itr.nextToken());
13                 context.write(word, one);
14             }
15         }
16     }
17
18     public static class IntSumReducer
19         extends Reducer<Text, IntWritable, Text, IntWritable>
20     {
21         private IntWritable result = new IntWritable();
22
23         public void reduce(Text key, Iterable<IntWritable> values,
24             Context context
25             throws IOException, InterruptedException {
26             int sum = 0;
27             for (IntWritable val : values) {
28                 sum += val.get();
29             }
30             result.set(sum);
31             context.write(key, result);
32         }
33
34     }
35
36     public static void main(String[] args) throws Exception {
37         Configuration conf = new Configuration();
38         String[] otherArgs = new GenericOptionsParser(conf, args).getRemainingArgs();
39         if (otherArgs.length < 2) {
40             System.err.println("Usage: wordcount <in> [<out>]");
41             System.exit(2);
42         }
43         Job job = new Job(conf, "word count");
44         job.setJarByClass(WordCount.class);
45         job.setMapperClass(TokenizerMapper.class);
46         job.setCombinerClass(IntSumReducer.class);
47         job.setReducerClass(IntSumReducer.class);
48         job.setOutputKeyClass(Text.class);
49         job.setOutputValueClass(IntWritable.class);
50         for (int i = 0; i < otherArgs.length - 1; ++i) {
51             FileInputFormat.addInputPath(job, new Path(otherArgs[i]));
52         }
53         FileOutputFormat.setOutputPath(job,
54             new Path(otherArgs[otherArgs.length - 1]));
55         System.exit(job.waitForCompletion(true) ? 0 : 1);
56     }
57 }
```

```
1 val f = sc.textFile(inputPath)
2 val w = f.flatMap(l => l.split(" ")).map(word => (word, 1)).cache()
3 w.reduceByKey(_ + _).saveAsText(outputPath)
```

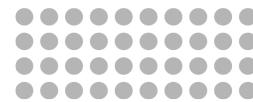
WordCount in 3 lines of Spark

WordCount in 50+ lines of Java MR

Fast: Time to sort 100TB

2013 Record:
Hadoop

2100 machines



72 minutes



2014 Record:
Spark

207 machines



23 minutes

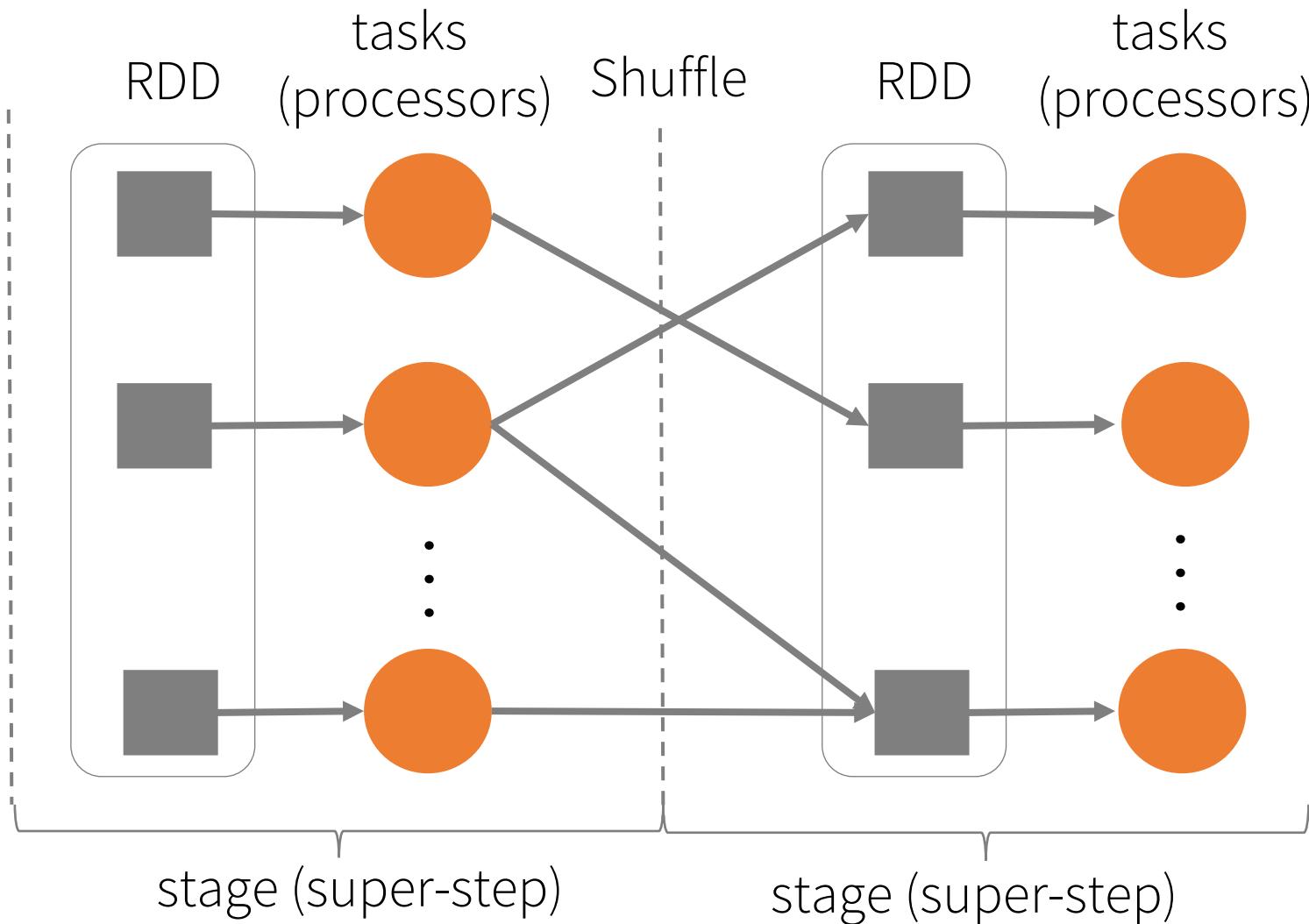


Also sorted 1PB in 4 hours (bottlenecked by network)

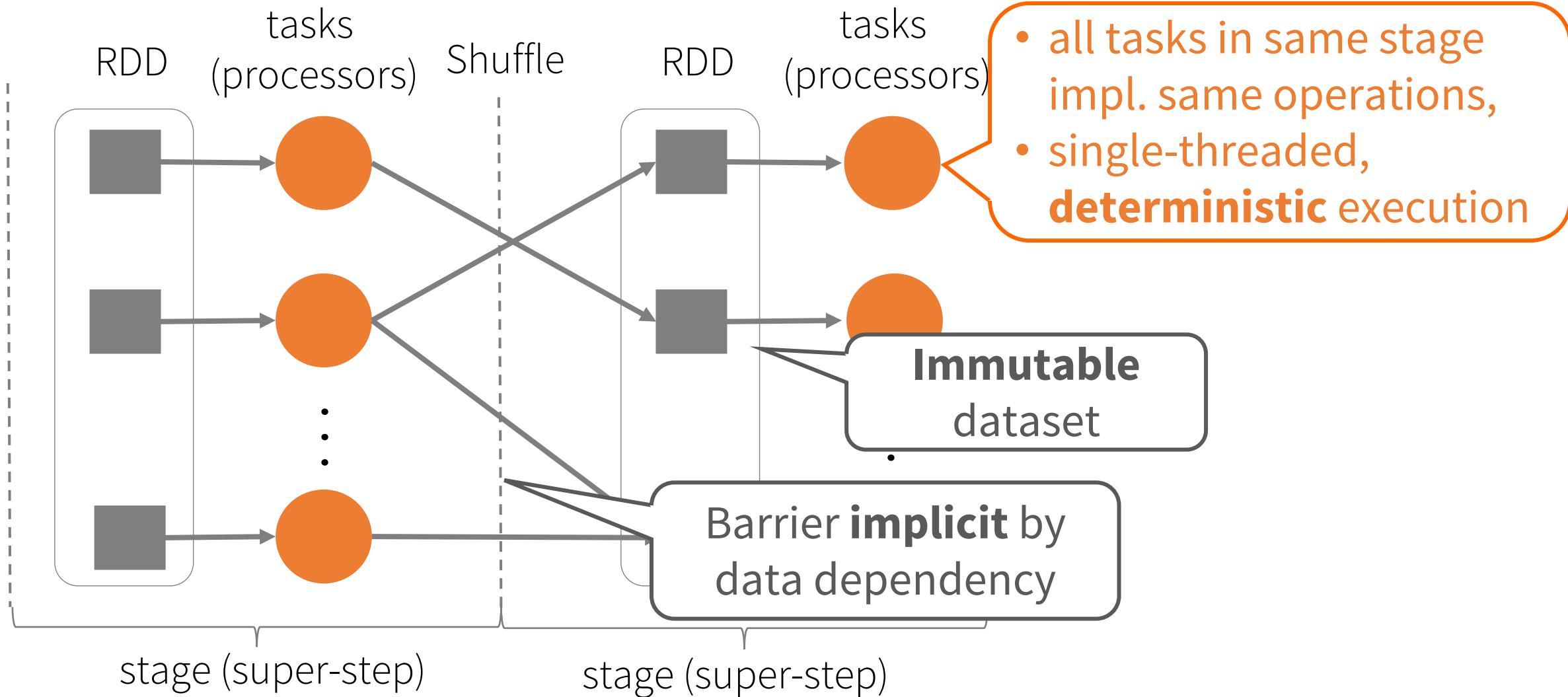
RDD: Resilient Distributed Datasets

- Collections of objects partitioned & distributed across a cluster
 - Stored in RAM or on Disk
 - Resilient to failures
- Operations
 - Transformations
 - Actions

Spark, as a BSP System



Spark, as a BSP System



Operations on RDDs

- Transformations $f(\text{RDD}) \Rightarrow \text{RDD}$
 - Lazy (not computed immediately)
 - E.g., “map”, “filter”, “groupBy”

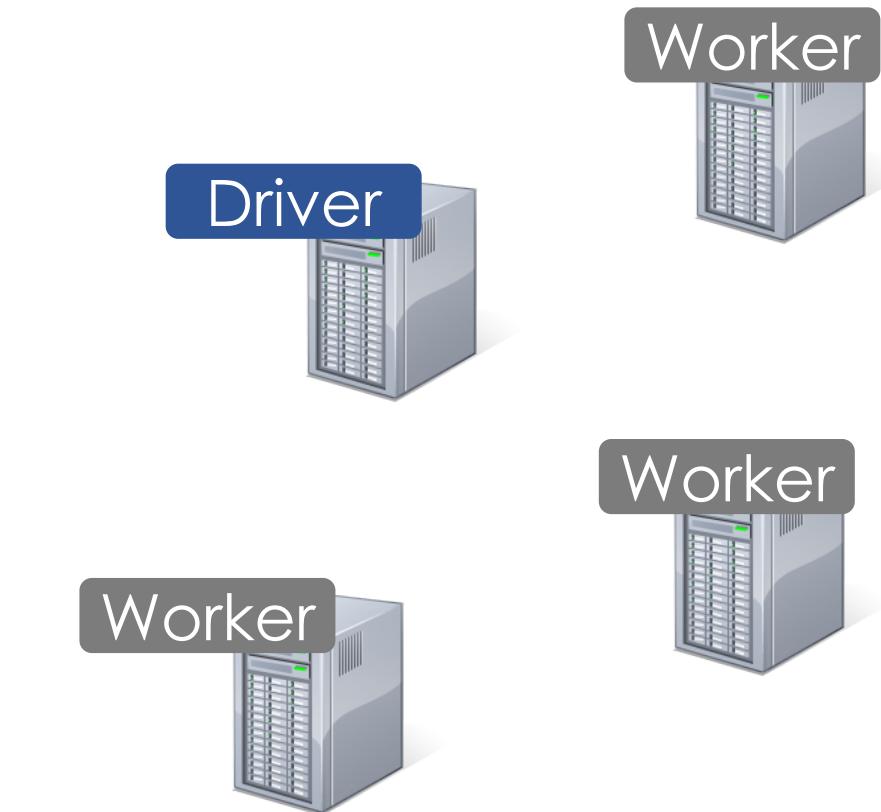
- Actions:
 - Triggers computation
 - E.g. “count”, “collect”, “saveAsTextFile”

Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

Example: Log Mining

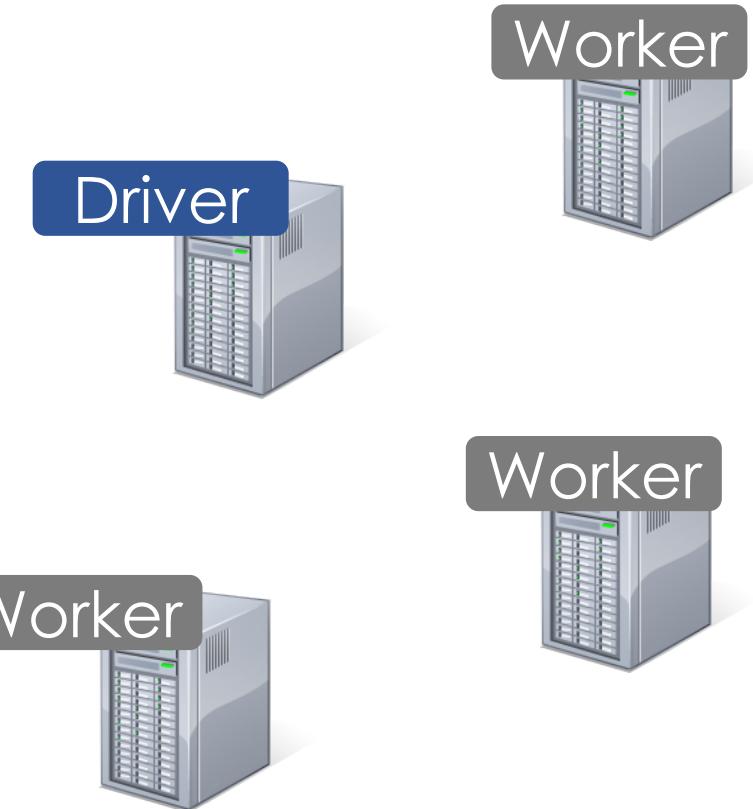
Load error messages from a log into memory,
then interactively search for various patterns



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

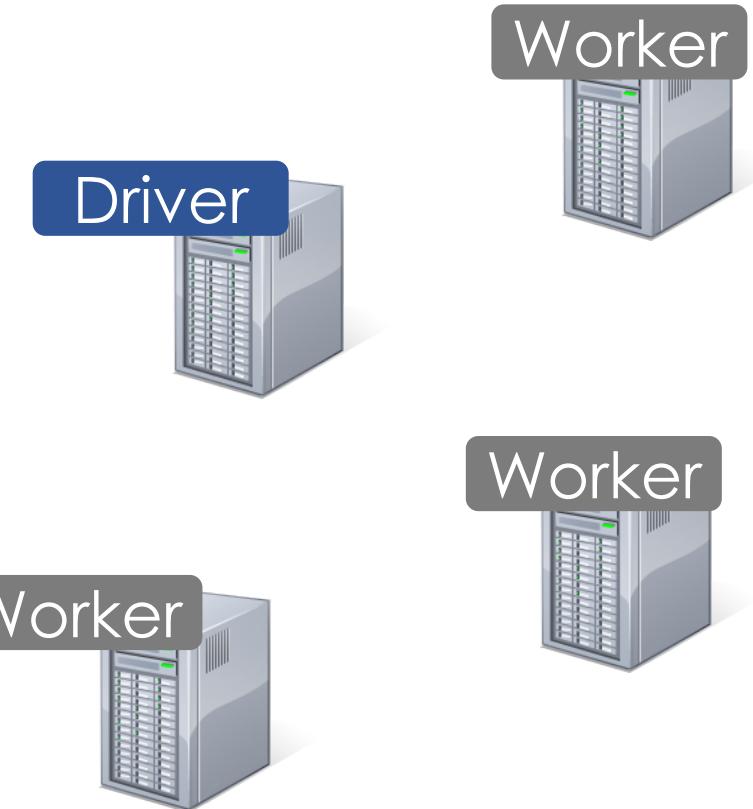
```
lines = spark.textFile("hdfs://...")
```



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

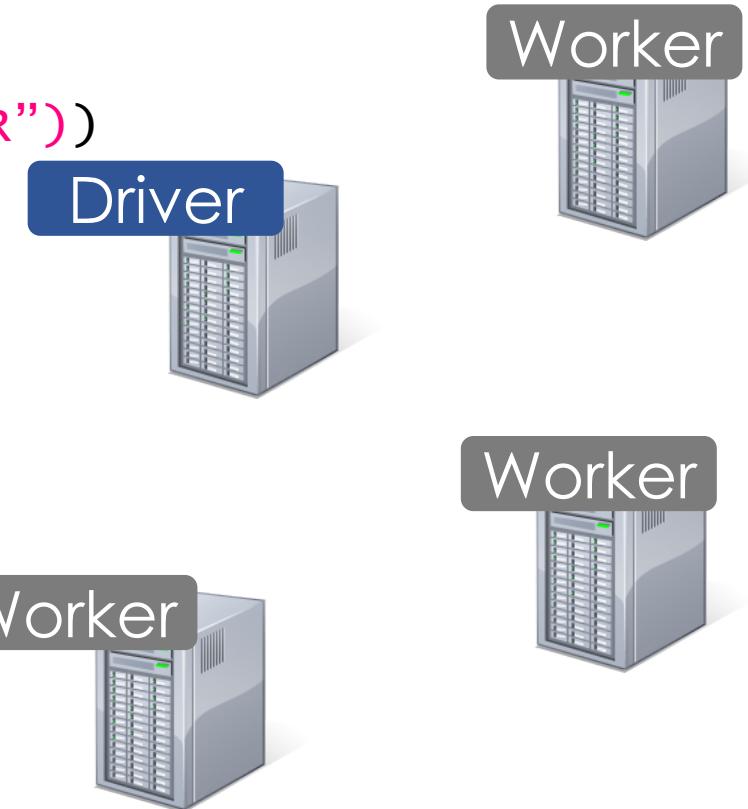
Base RDD
`lines = spark.textFile("hdfs://...")`



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))
```



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

Transformed RDD

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))
```

Driver

Worker

Worker

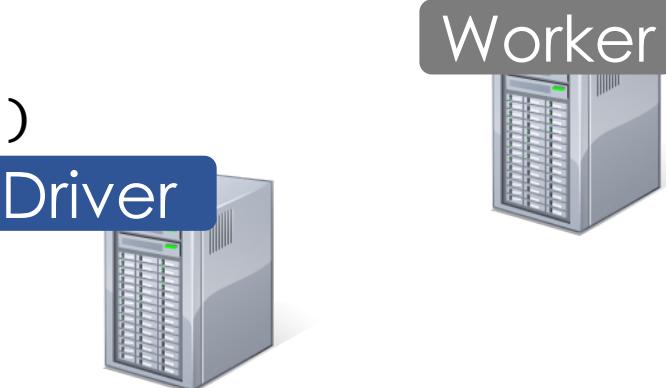
Worker



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

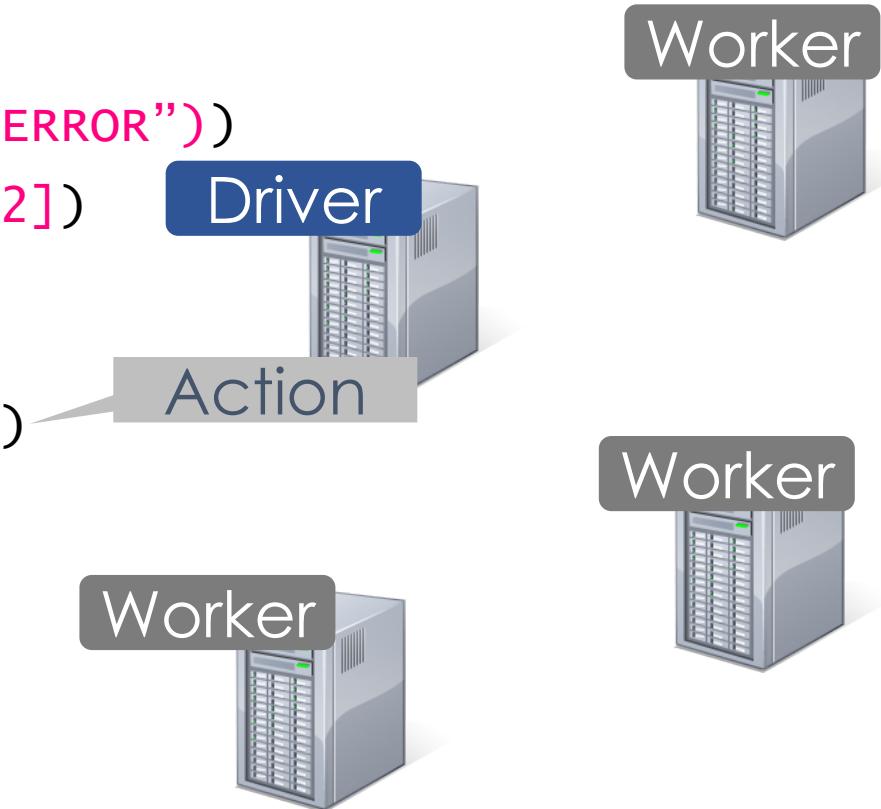
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()
```



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

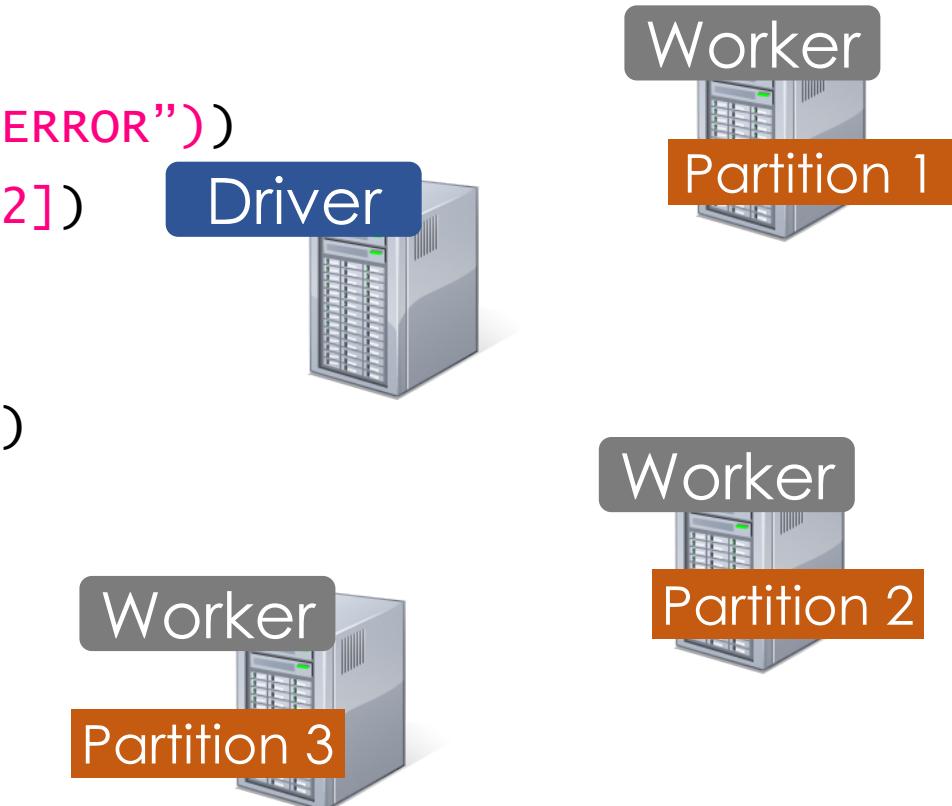
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()
```



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

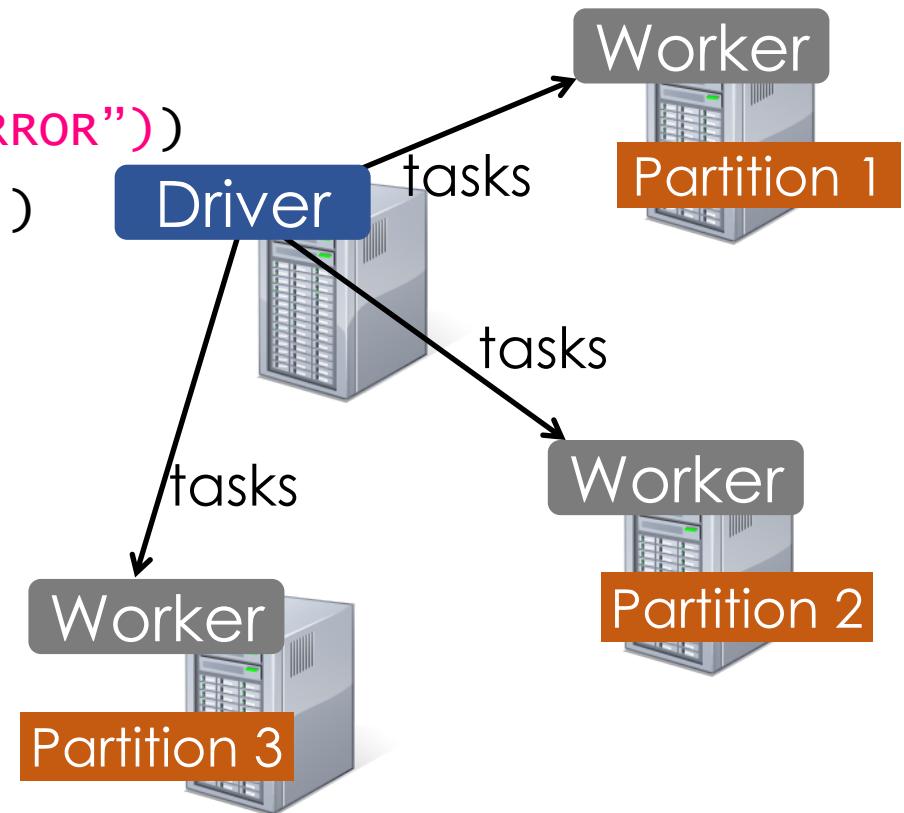
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()
```



Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

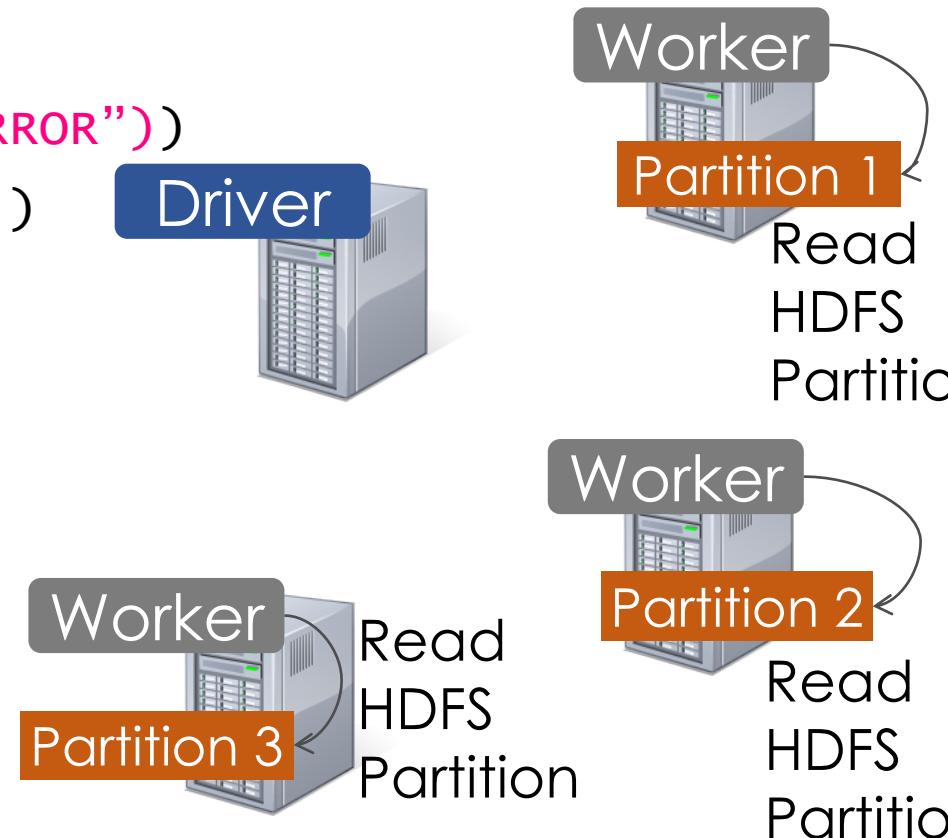
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()
```



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

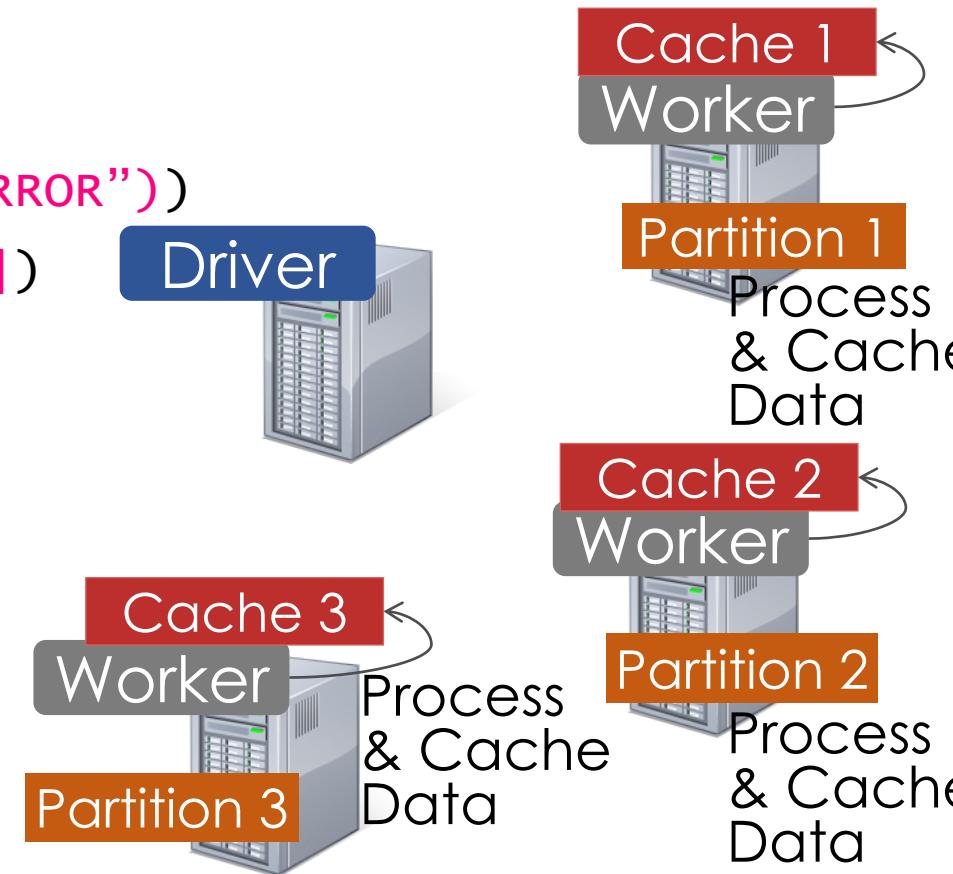
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()
```



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

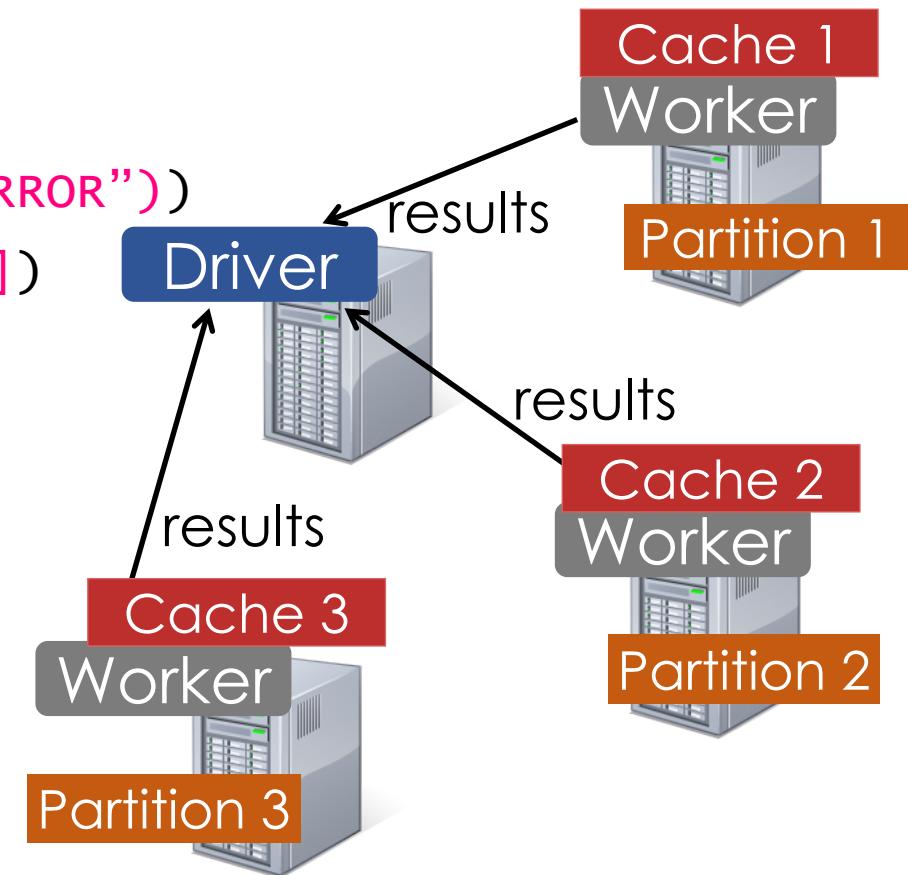
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()
```



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

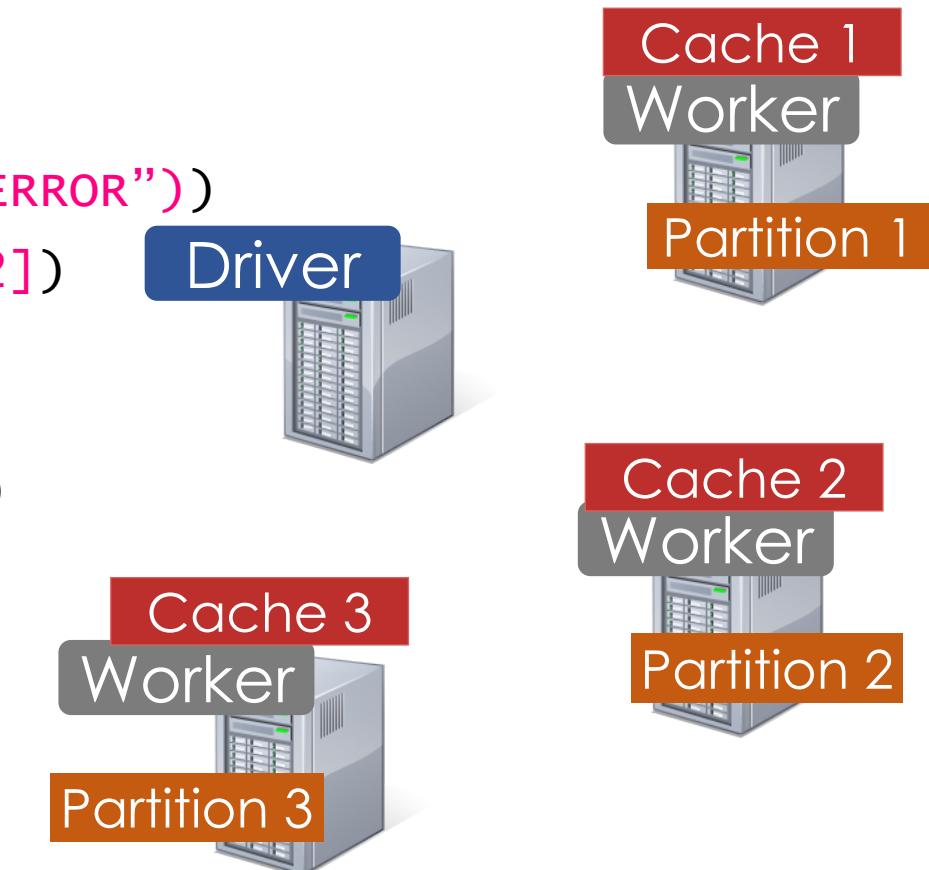
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()
```



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

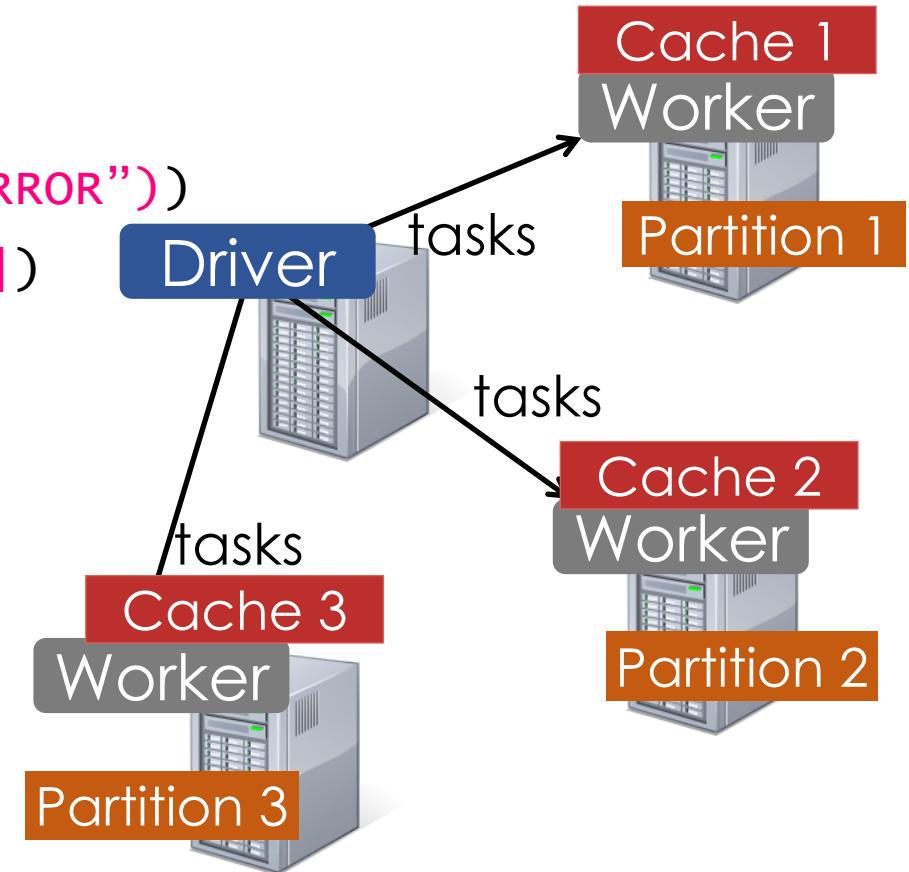
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```



Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

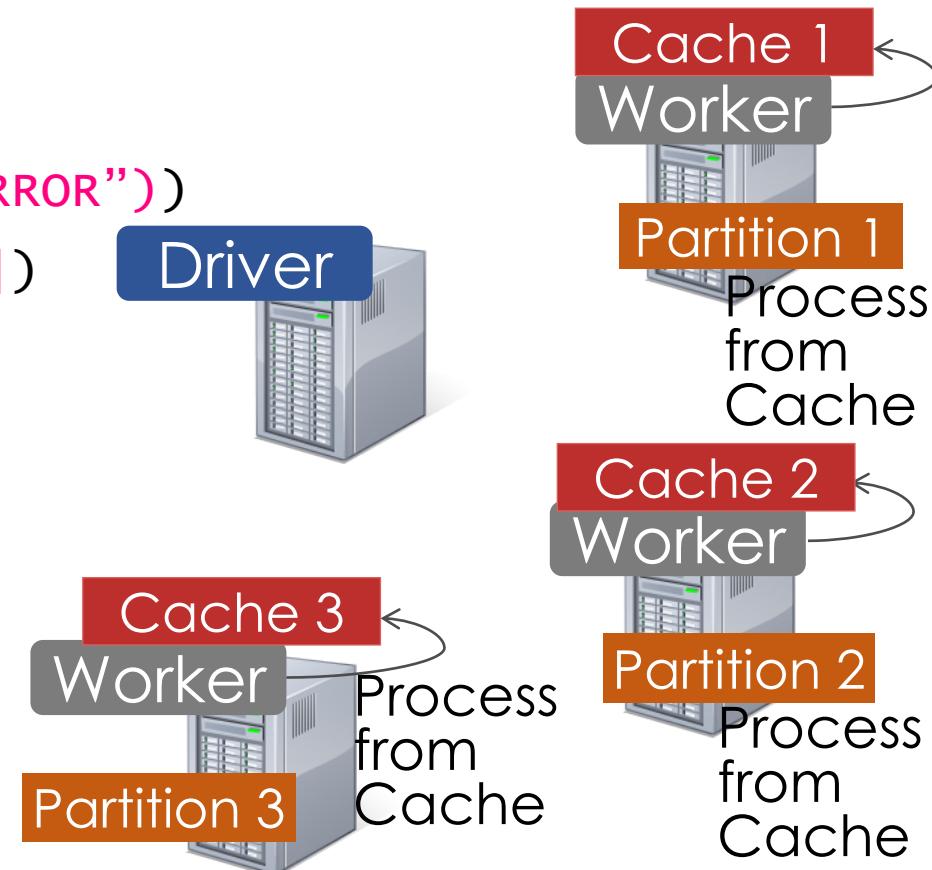
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```



Example: Log Mining

Load error messages from a log into memory,
then interactively search for various patterns

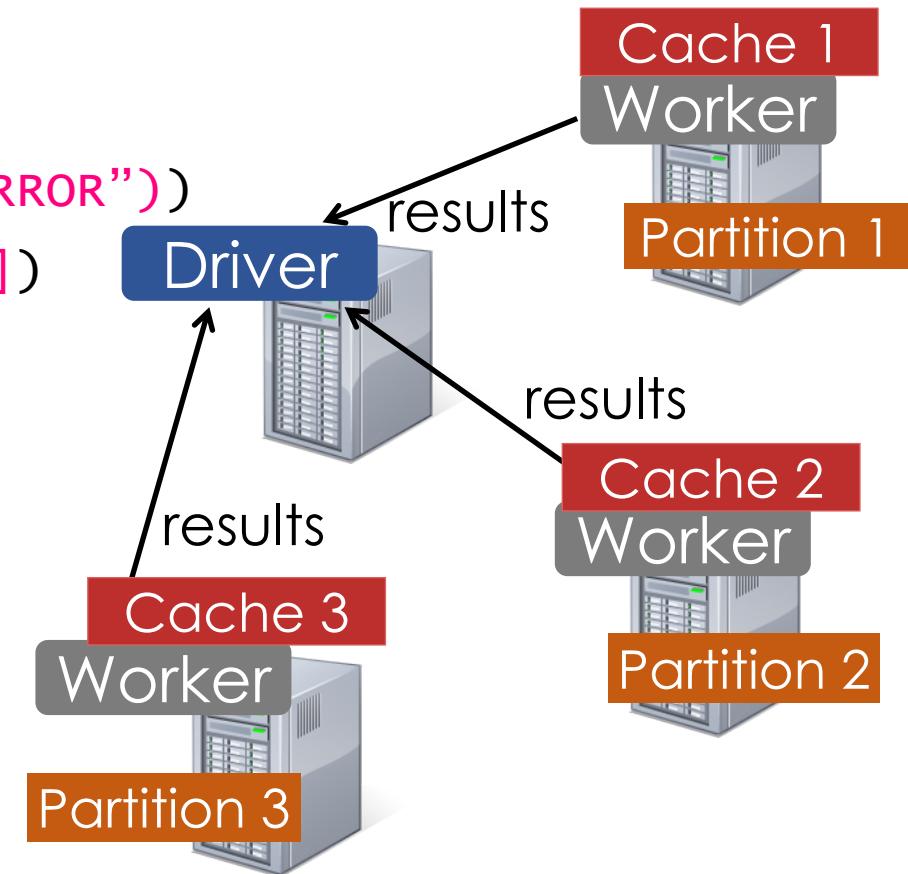
```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```



Example: Log Mining

Load error messages from a log into memory, then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```



Example: Log Mining

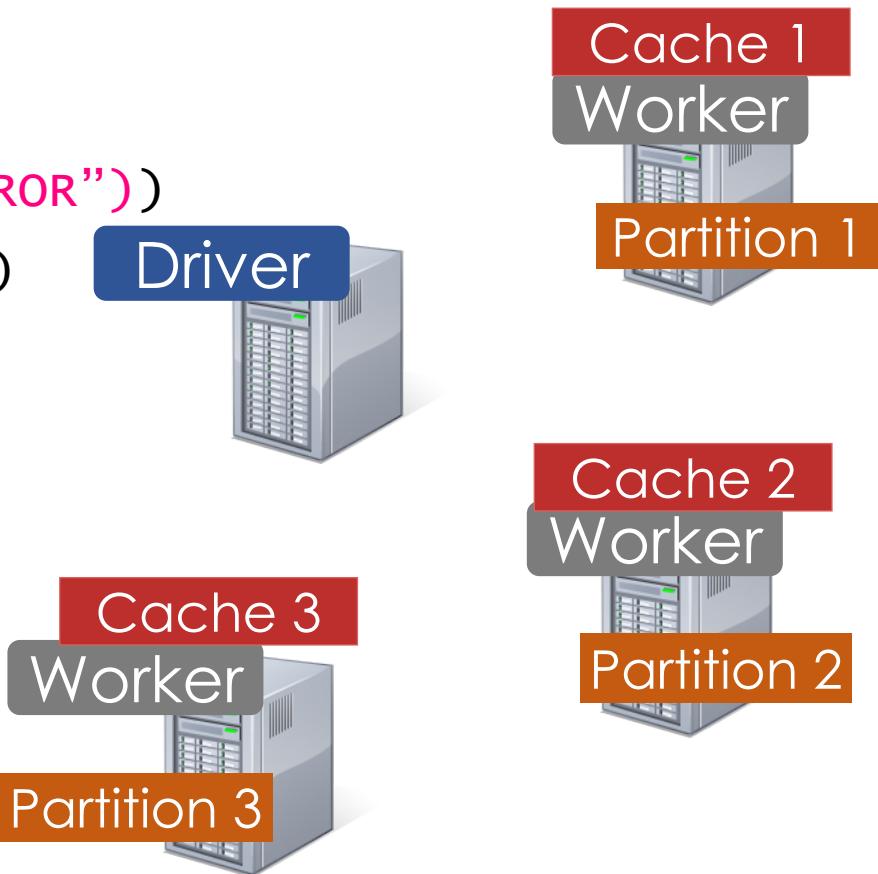
Load error messages from a log into memory,
then interactively search for various patterns

```
lines = spark.textFile("hdfs://...")  
errors = lines.filter(lambda s: s.startswith("ERROR"))  
messages = errors.map(lambda s: s.split("\t")[2])  
messages.cache()  
  
messages.filter(lambda s: "mysql" in s).count()  
messages.filter(lambda s: "php" in s).count()
```

Cache your data → Faster Results

Full-text search of Wikipedia

- 60GB on 20 EC2 machines
- 0.5 sec from mem vs. 20s for on-disk



Abstraction: Dataflow Operators

map

filter

groupBy

sort

union

join

leftOuterJoin

rightOuterJoin

reduce

count

fold

reduceByKey

groupByKey

cogroup

cross

zip

sample

take

first

partitionBy

mapwith

pipe

save

...

Abstraction: Dataflow Operators

map

filter

groupBy

sort

union

join

leftOuterJoin

rightOuterJoin

reduce

count

fold

reduceByKey

groupByKey

cogroup

cross

zip

sample

take

first

partitionBy

mapwith

pipe

save

...

Language Support

Python

```
lines = sc.textFile(...)  
lines.filter(lambda s: "ERROR" in s).count()
```

Scala

```
val lines = sc.textFile(...)  
lines.filter(x => x.contains("ERROR")).count()
```

Java

```
JavaRDD<String> lines = sc.textFile(...);  
lines.filter(new Function<String, Boolean>() {  
    Boolean call(String s) {  
        return s.contains("error");  
    }  
}).count();
```

Standalone Programs

Python, Scala, & Java

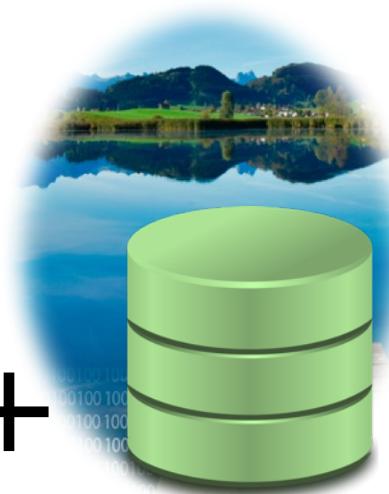
Interactive Shells

Python & Scala

Performance

Java & Scala are faster
due to static typing

K-Means +



How do we run k-means on the
data warehouse / data lake?

Can we express K-Means in the Statistical Query Pattern?

```
centers ← pick k initial Centers
while (centers are changing):
    // Compute the assignments (E-Step)
    asg ← [(x, nearest(centers, x)) for x in data] 
    for i in range(k): // Compute the new centers (M-Step)
        centers[i] = mean([x for (x, c) in asg if c == i])
```

Query returns
all the data ...

Merge with M-Step
→
**Statistical Query
Pattern**

```
centers ← pick k initial Centers
while (centers are changing):
    for i in range(k):
        new_centers[i] =
            mean([x for x in data if nearest(centers, x) == i])
    centers = new_centers
```

K-Means in Map-Reduce (for HW)

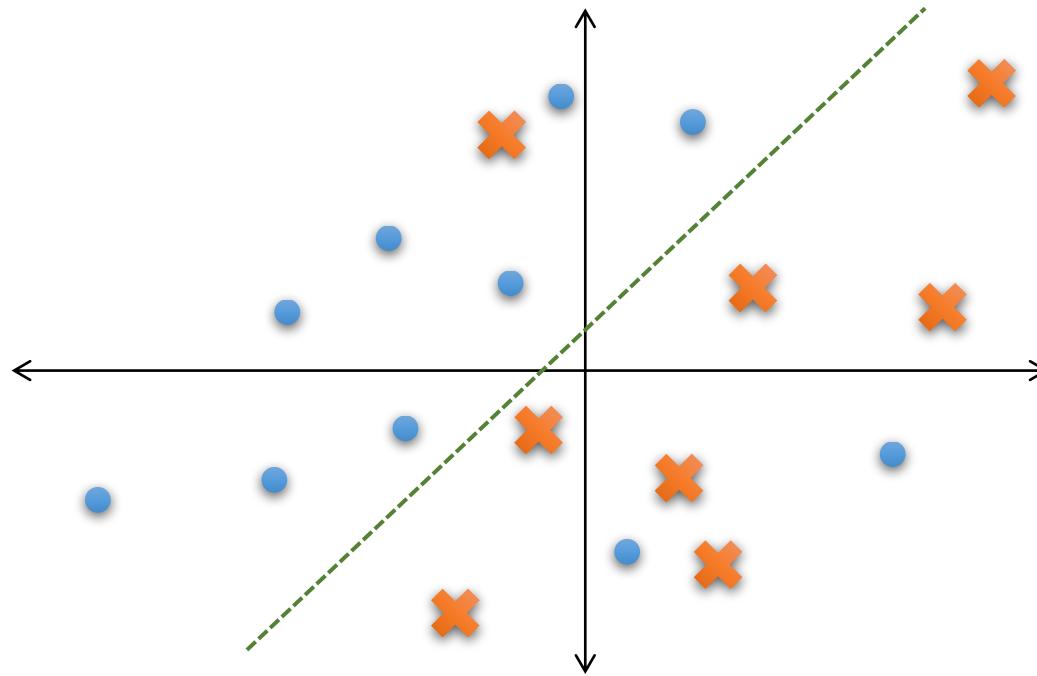
- **MapFunction**(old_centers, x)
 - Compute the index of the nearest old center
 - Return (**key** = nearest_centers, **value** = (x, 1))
- **ReduceFunction** combines values and counts
 - For each cluster center (Group By)
- Data system returns aggregate statistics:

$$s_i = \sum_{x \in \text{Cluster } i} x_i \quad \text{and} \quad n_i = \sum_{x \in \text{Cluster } i} 1$$

- ML algorithm computes new centers: $\mu_i = s_i/n_i$

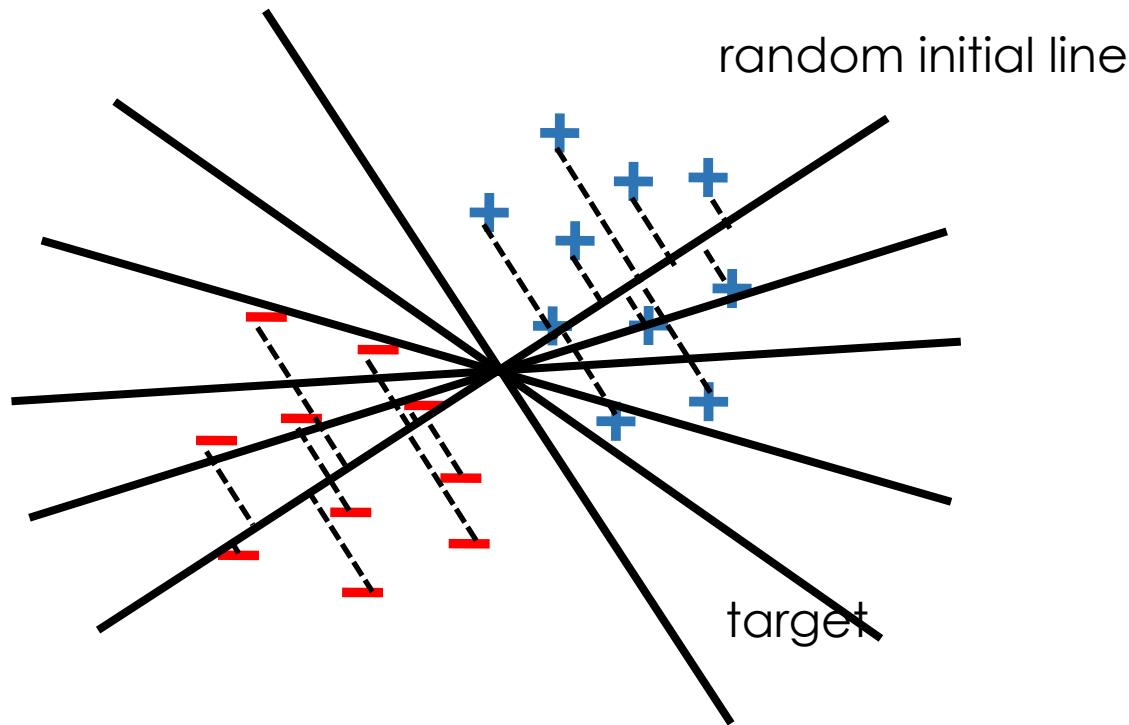
Not Covered In Class

Logistic Regression?



Logistic Regression

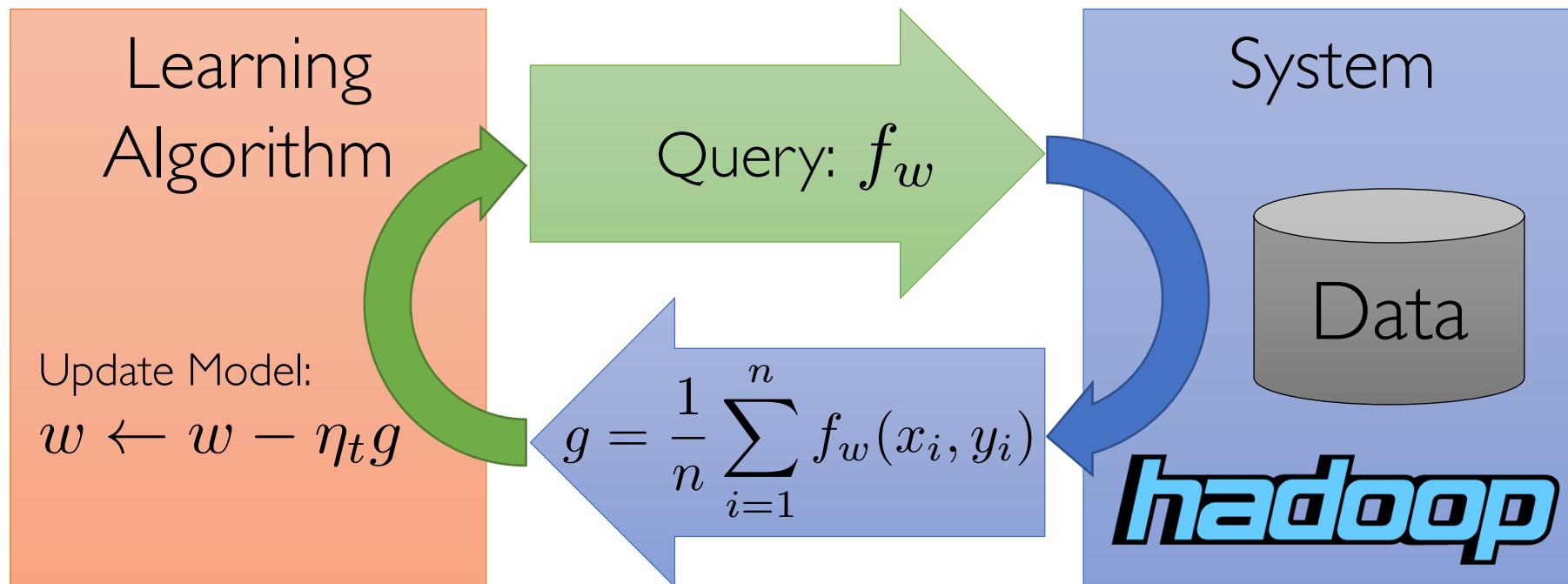
Iterative batch gradient descent.



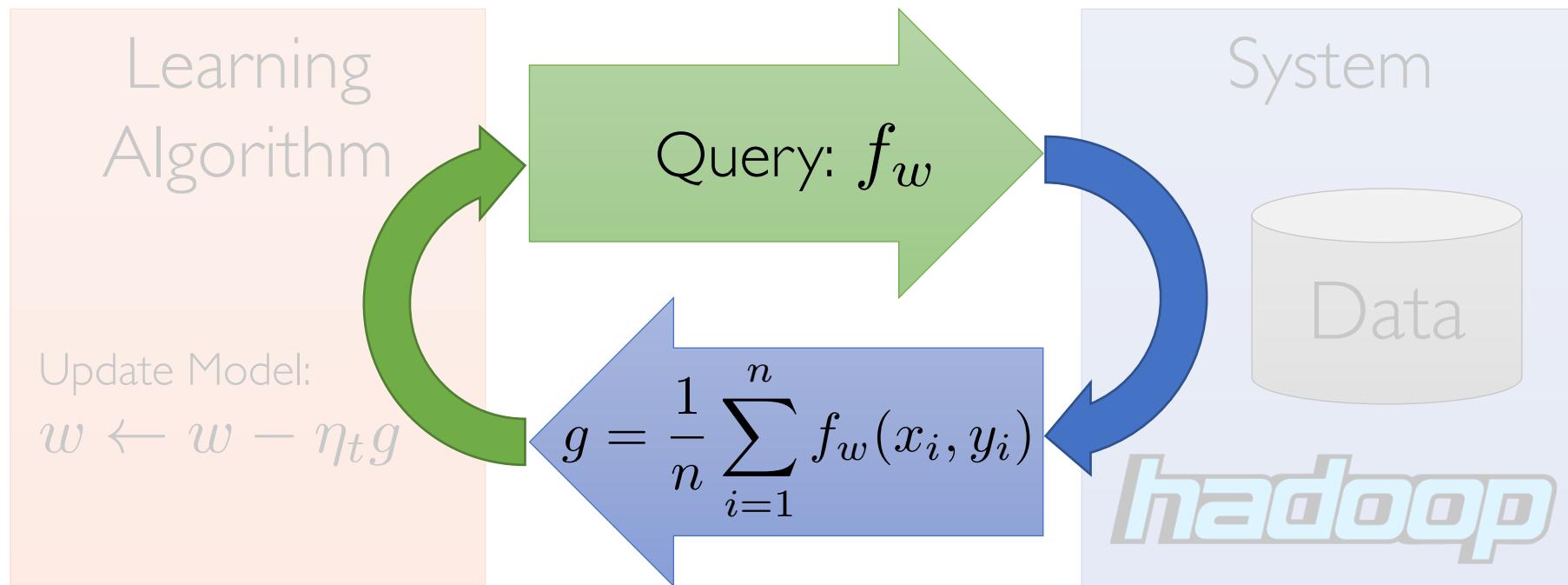
Logistic Regression in Map-Reduce

- Gradient descent:

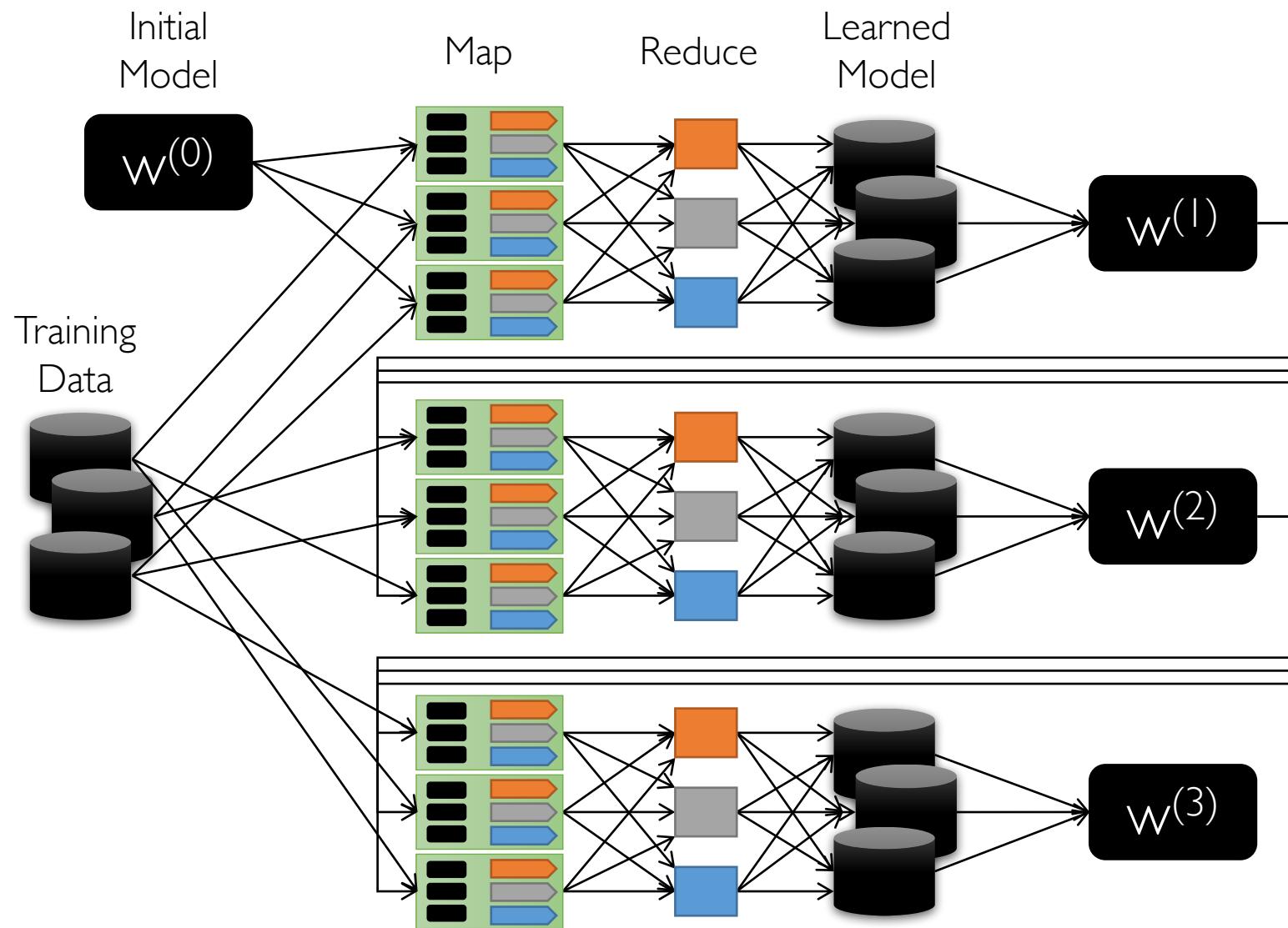
$$f_{\theta}(x, y) = \nabla_{\theta} (-\log \mathcal{L}(\theta; (x, y)))$$



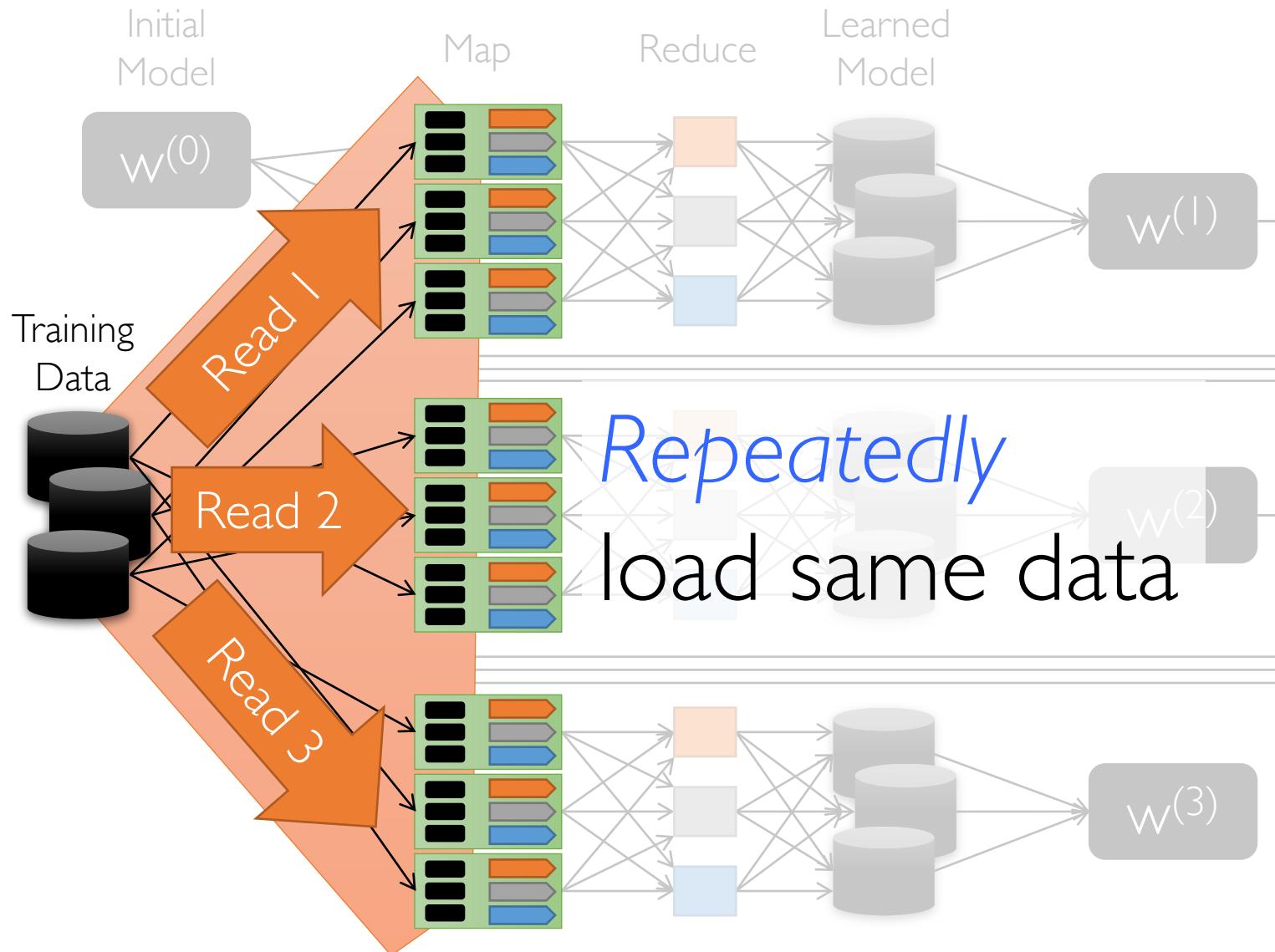
Map-Reduce is not optimized for iteration and multi-stage computation



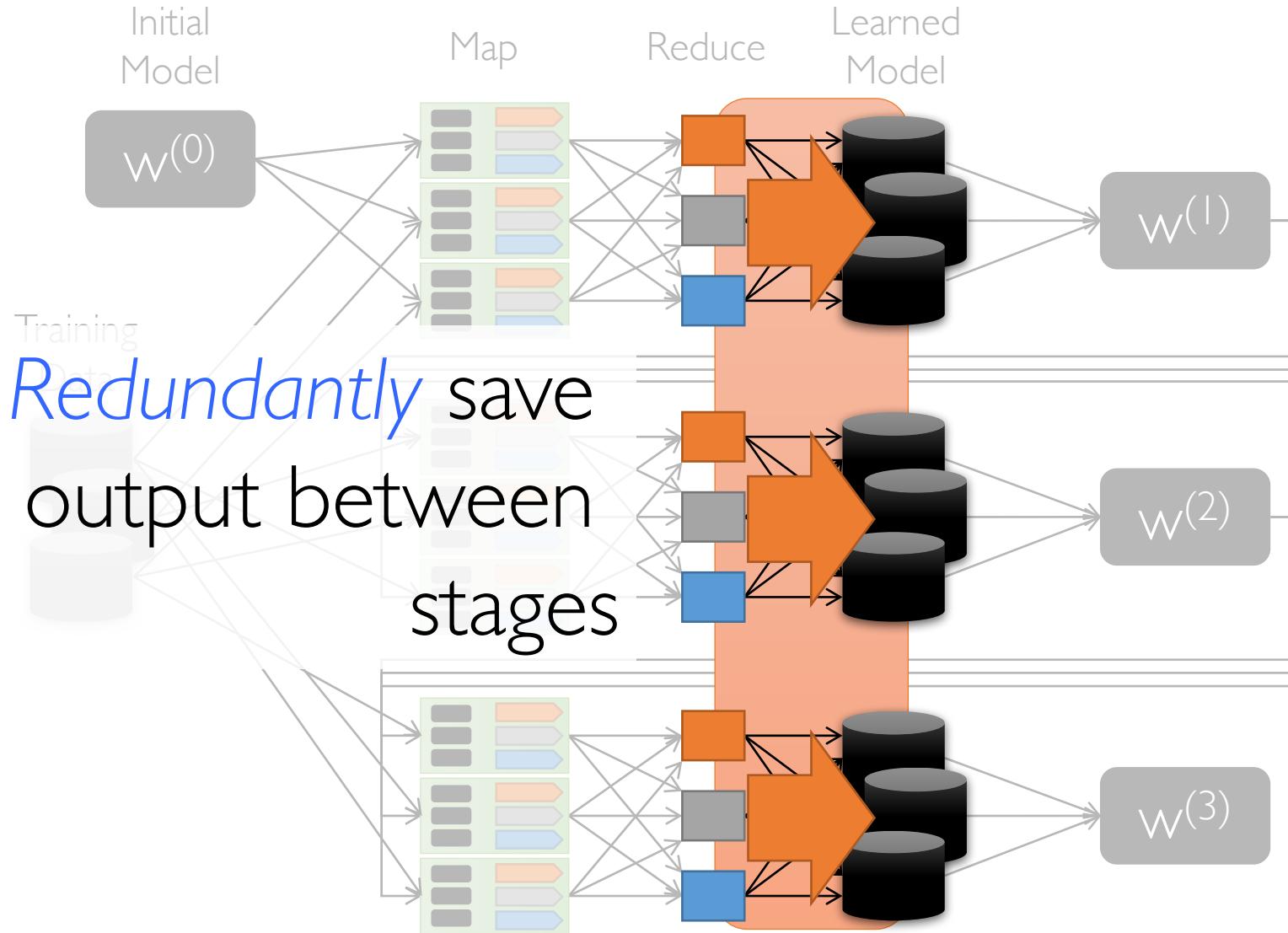
Iteration in Map-Reduce



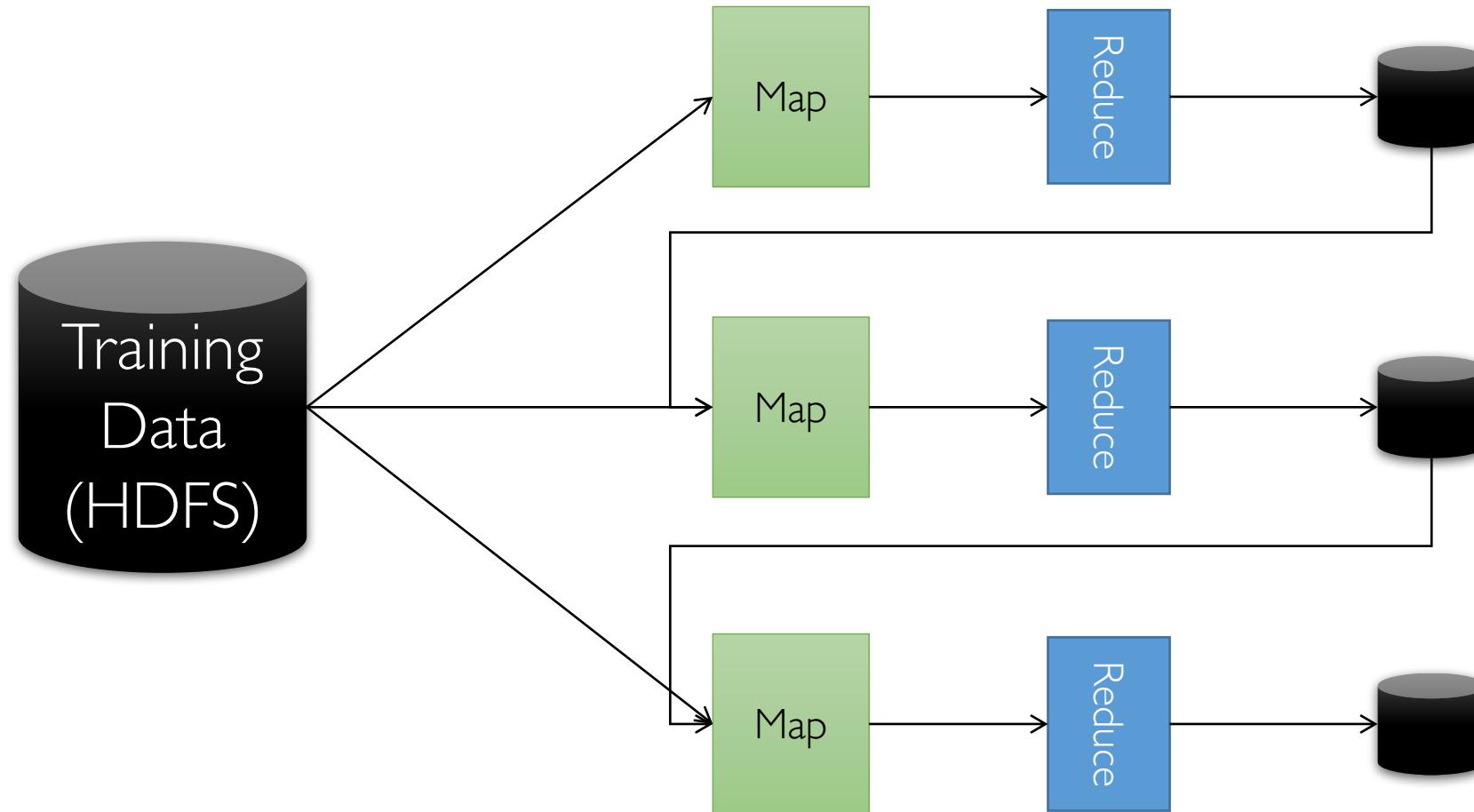
Cost of Iteration in Map-Reduce



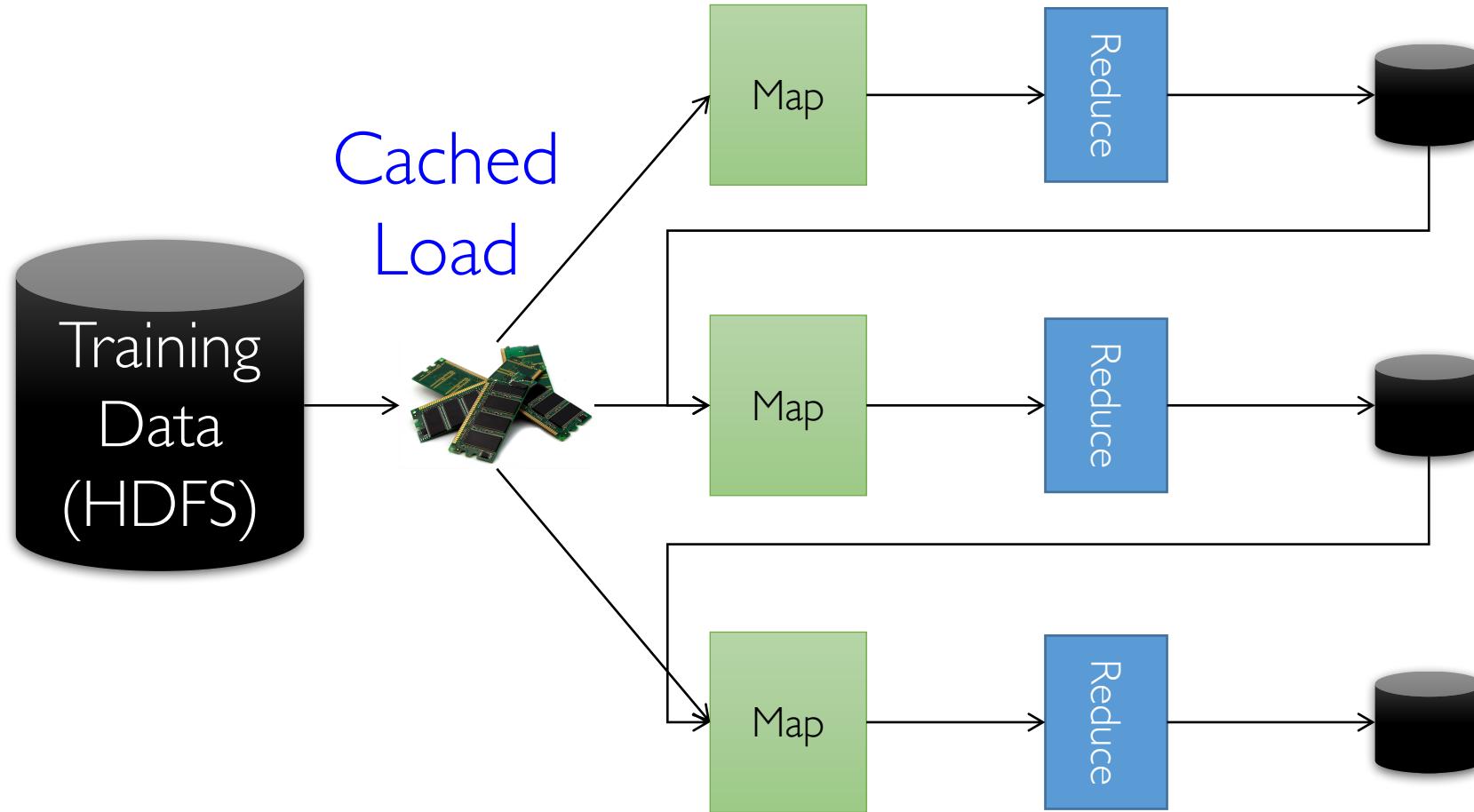
Cost of Iteration in Map-Reduce



Dataflow View



Memory Opt. Dataflow



10-100× faster than network and disk

Memory Opt. Dataflow View

