Integrating MapReduce with HBase

Overview

Understand the Hadoop MapReduce framework

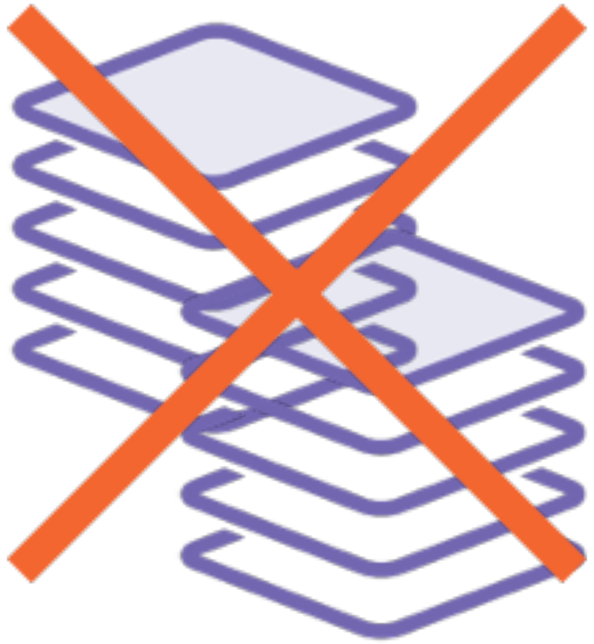
Use MapReduce with HBase for complex SQL-like operations

Set up and run MapReduce on an HBase table

Complex and Multi-table Operations in HBase

Grouping and Sorting by column

Aggregation of column values



Joins across multiple tables

Complex and Multi-table Operations in HBase

There is a way around this though...

HBase allows **programmatic** access to data using **MapReduce**

MapReduce

Parallel programming model usually implemented in Java

Full control over how we process data

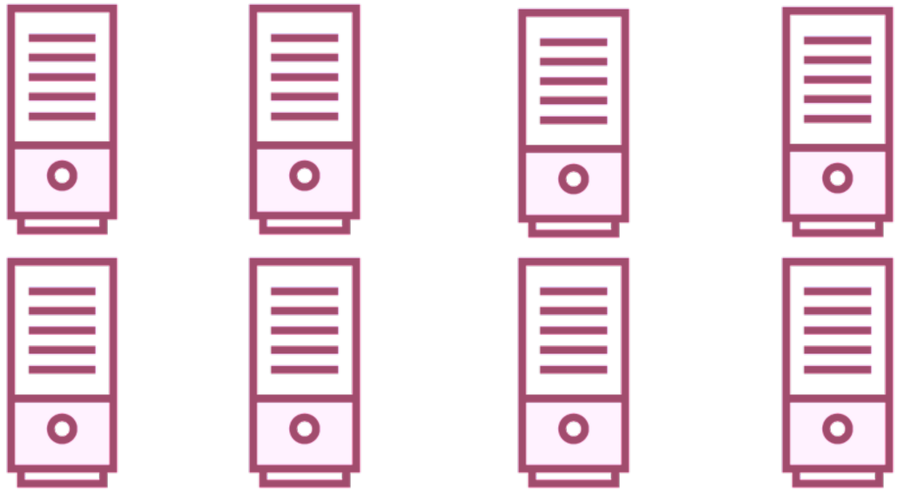
MapReduce

Processing huge amounts of data

MapReduce

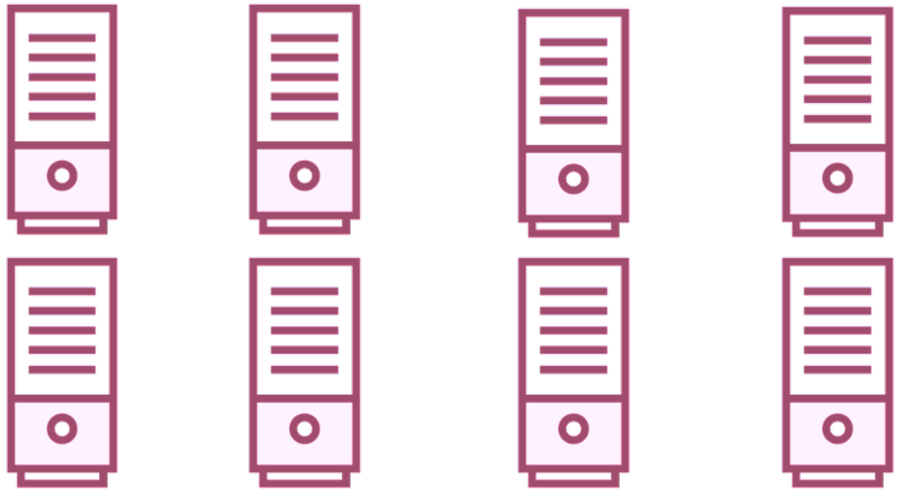
A distributed system

MapReduce



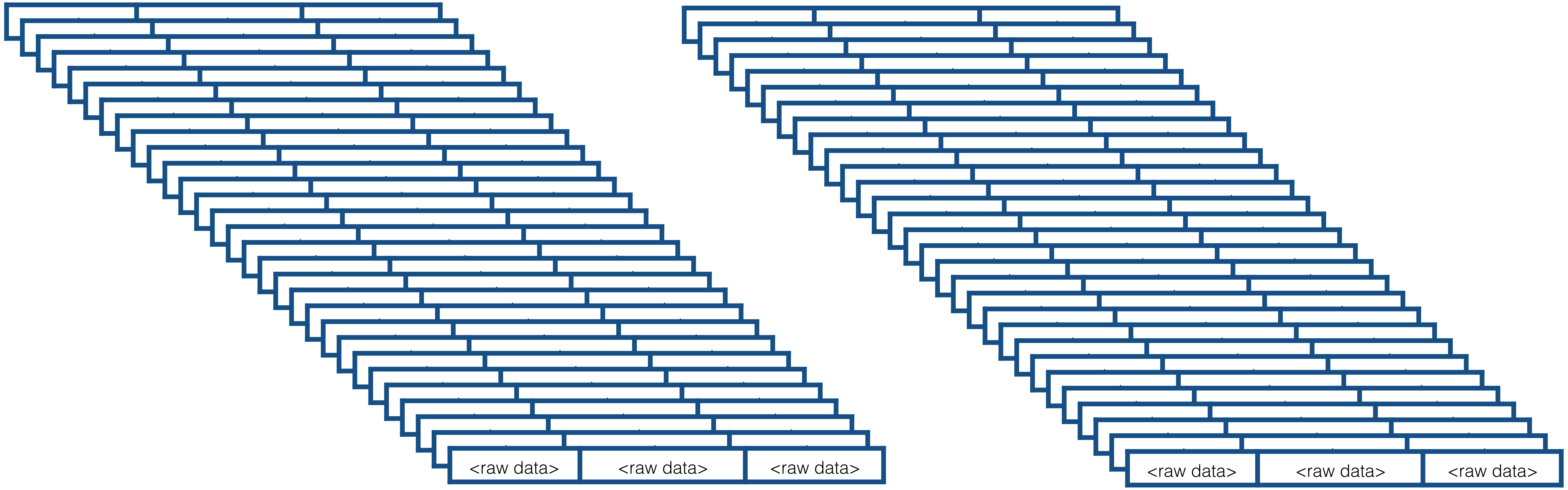
MapReduce is a programming paradigm

MapReduce



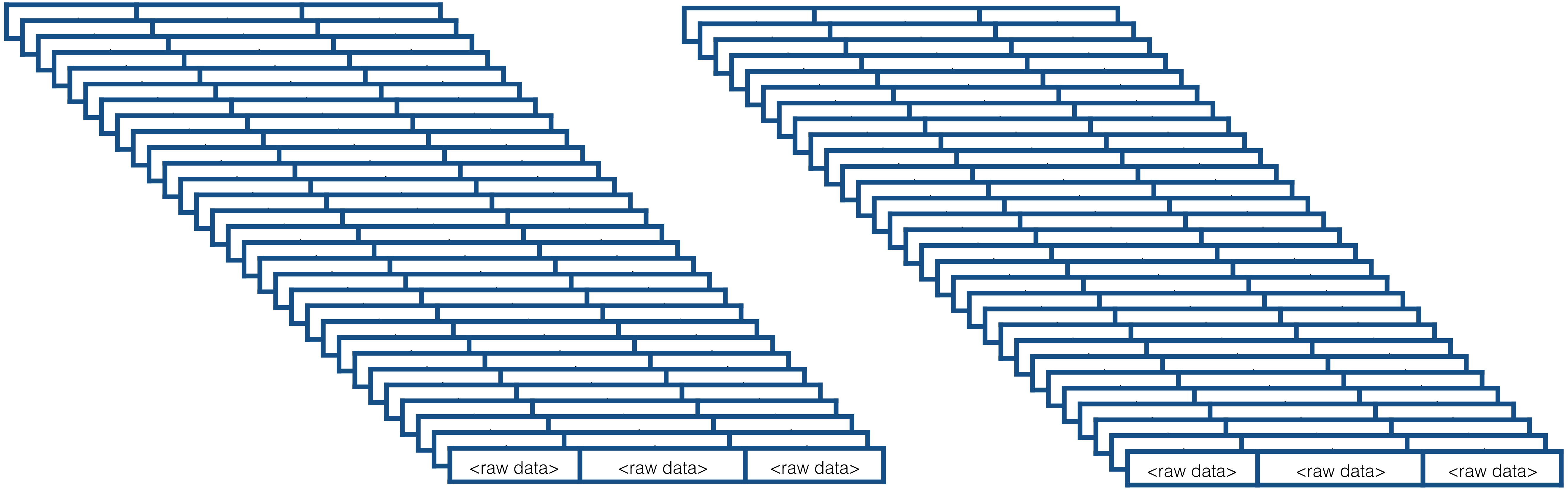
Takes advantage of the inherent parallelism in data processing

MapReduce



Modern systems generate millions of records of raw data

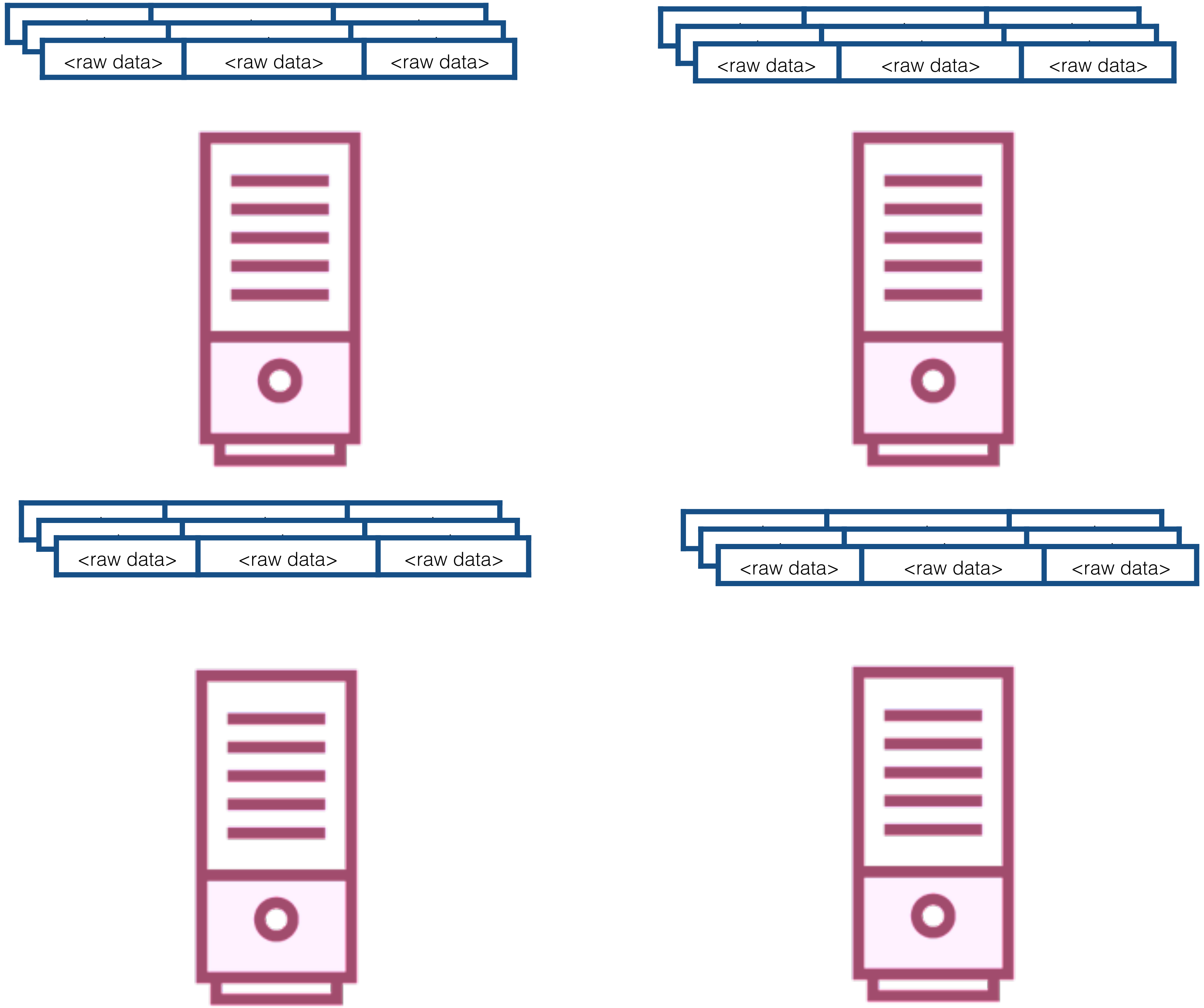
MapReduce



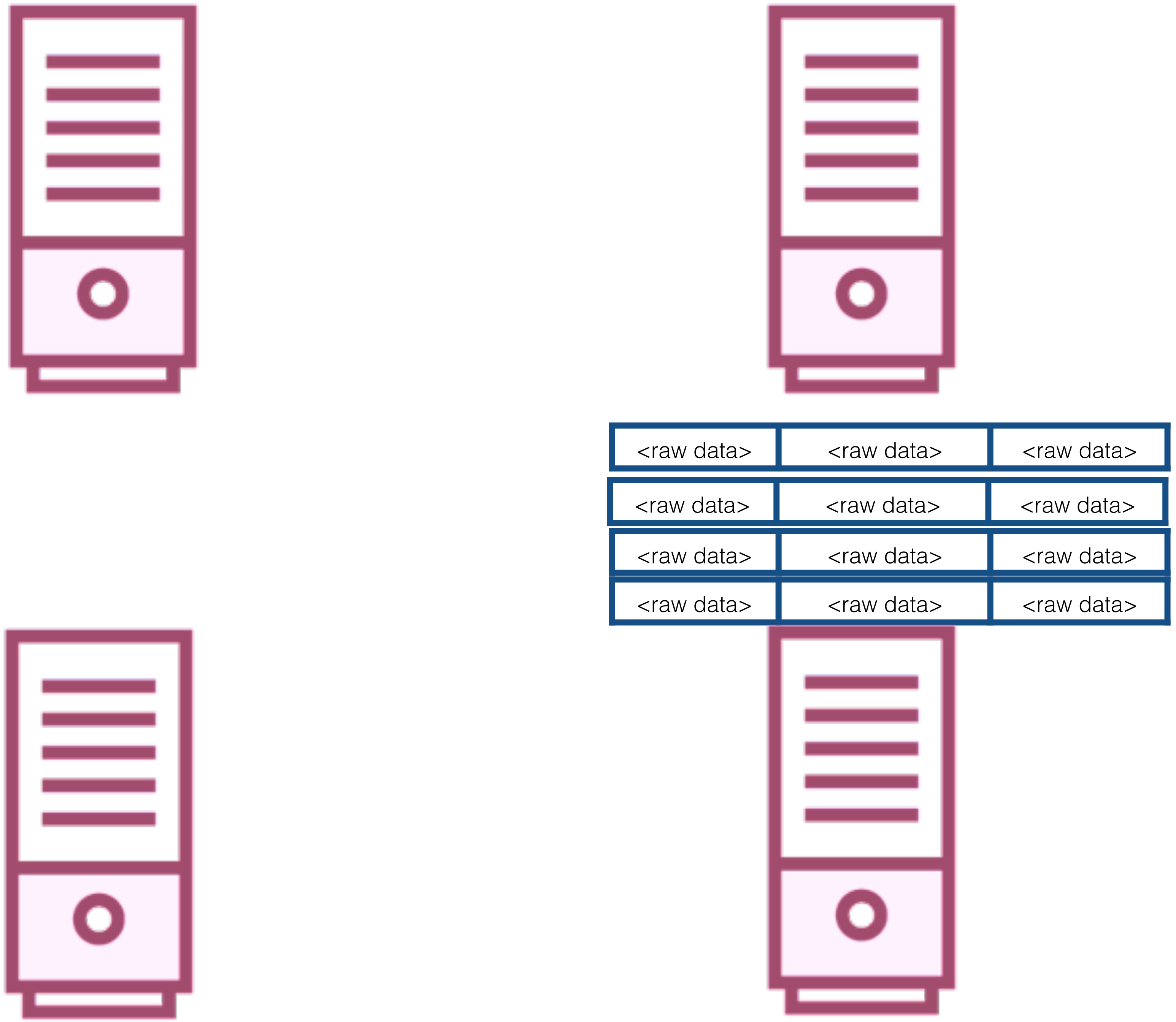
A task of this scale is processed in two stages

map reduce

map



reduce



<raw data>	<raw data>	<raw data>
<raw data>	<raw data>	<raw data>
<raw data>	<raw data>	<raw data>
<raw data>	<raw data>	<raw data>

MapReduce

`map` `reduce`

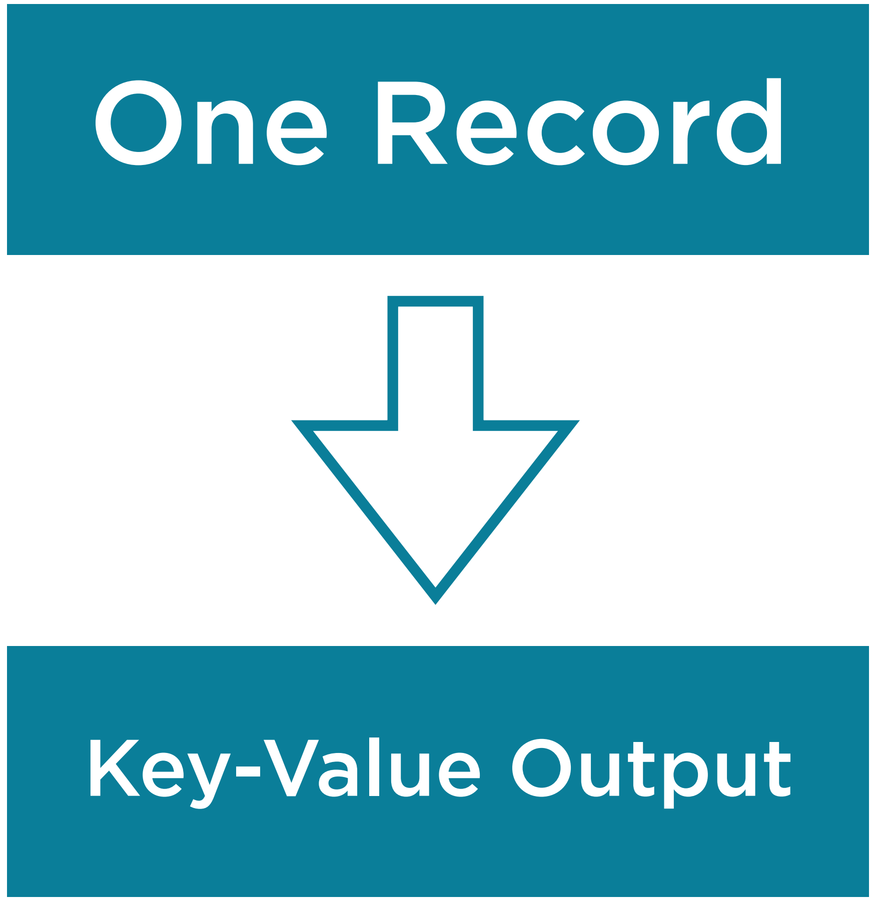
The programmer defines these 2 functions

Hadoop does the rest - behind the scenes

map

An operation performed in parallel, on small portions of the dataset

map



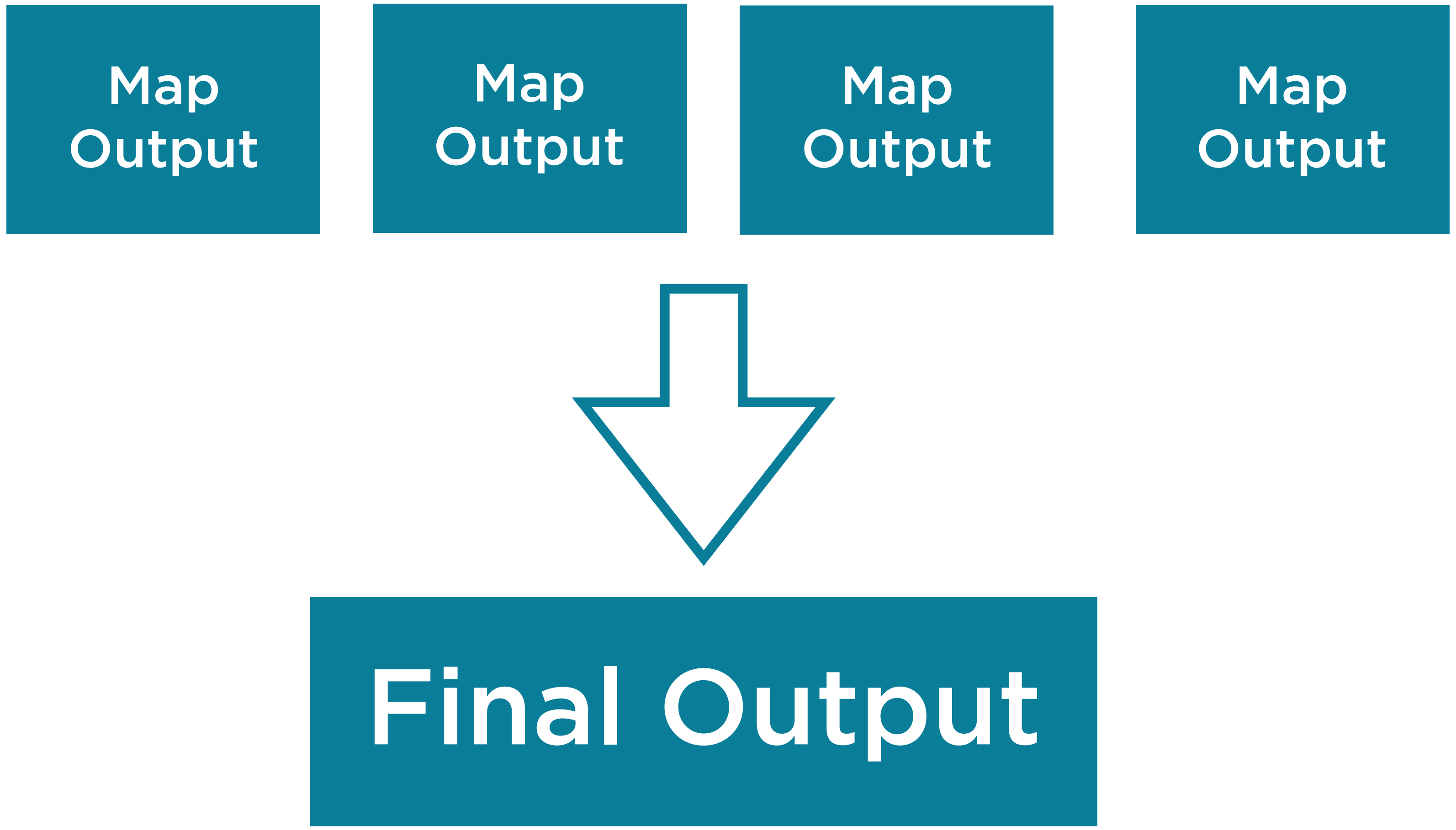
One Record

Key-Value Output

reduce

An operation to combine the results of the map step

reduce



Map Output

Map Output

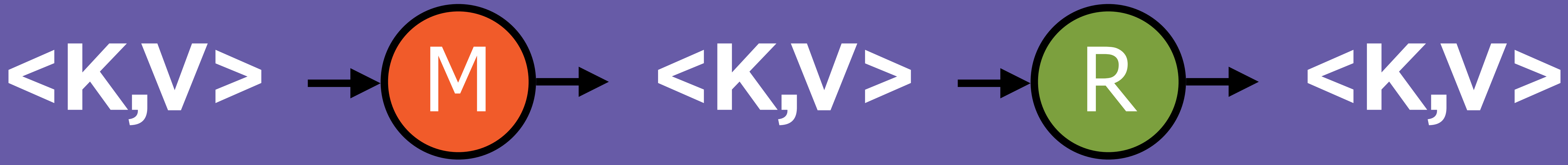
Map Output

Map Output

Final Output

map	A step that can be performed in parallel
reduce	A step to combine the intermediate results

Key Insight Behind MapReduce



Many data processing tasks can be expressed in this form

Key-Value Representation of an HBase Table

Key

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	mike
4	Type	sale
4	Content	Clothes sale

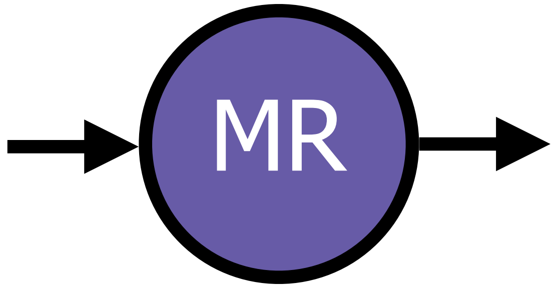
Key-Value Representation of an HBase Table

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	mike
4	Type	sale
4	Content	Clothes sale

Value

Message Count per User

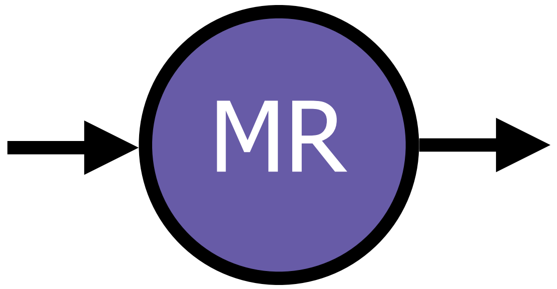
Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	mike
4	Type	sale
4	Content	Clothes sale



To	Count
mike	2
john	1
jill	1

Message Count per User

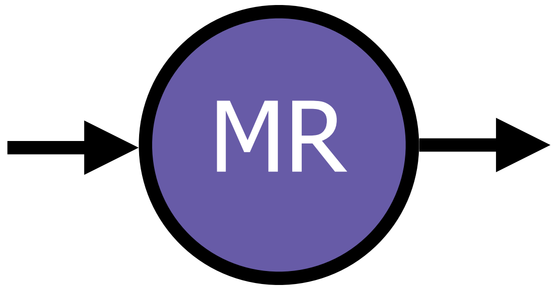
Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	mike
4	Type	sale
4	Content	Clothes sale



To	Count
mike	2
john	1
jill	1

Message Count per User

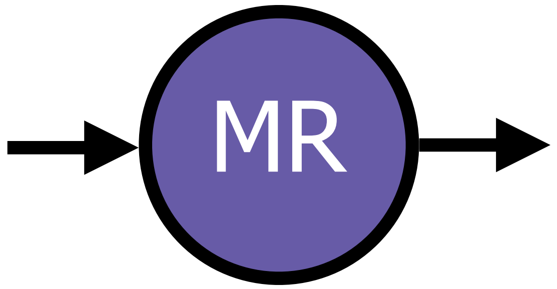
Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	mike
4	Type	sale
4	Content	Clothes sale



To	Count
mike	2
john	1
jill	1

Notification Type Count

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	mike
4	Type	sale
4	Content	Clothes sale



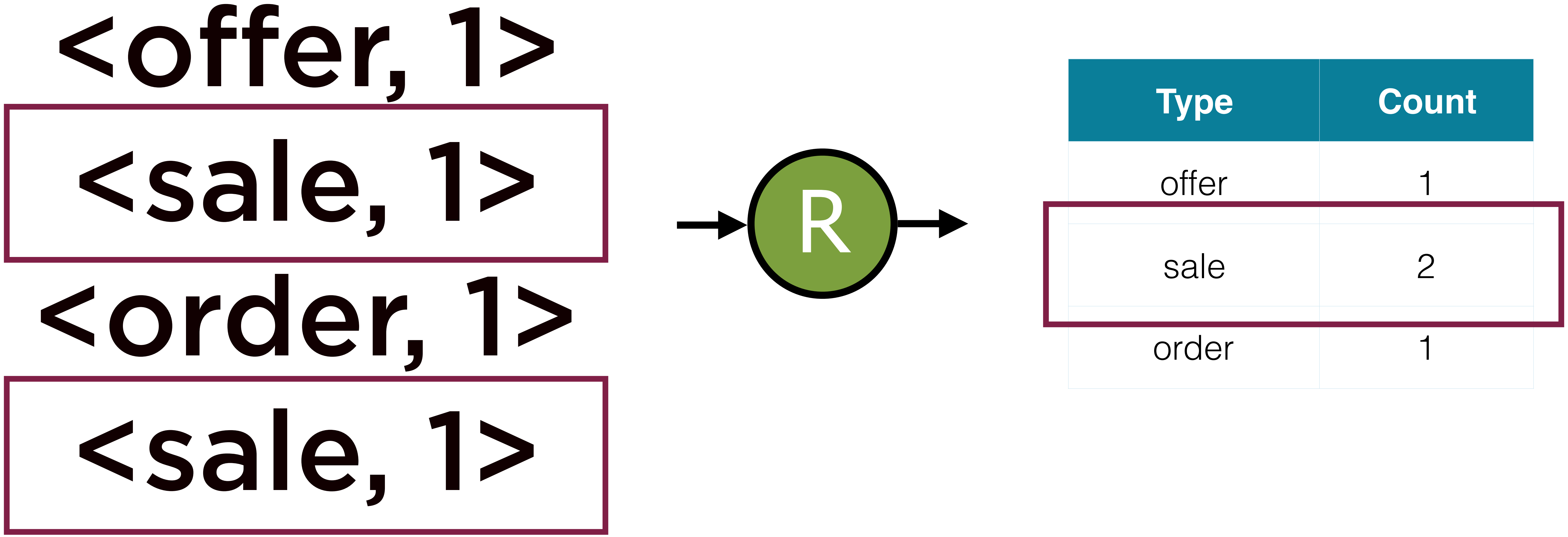
Type	Count
offer	1
sale	2
order	1

Notification Type Count

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	mike
4	Type	sale
4	Content	Clothes sale

<offer, 1>
<sale, 1>
<order, 1>
<sale, 1>

Notification Type Count



<offer, 1>

<sale, 1>

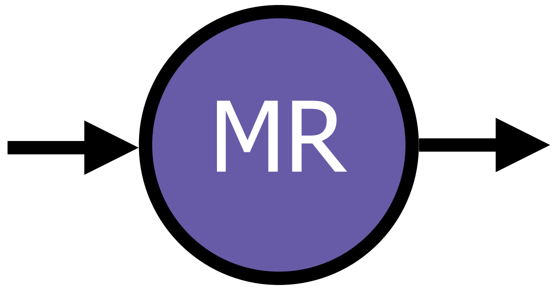
<order, 1>

<sale, 1>

Type	Count
offer	1
sale	2
order	1

Notification Type Count

Id	Column	Value
1	To	mike
1	Type	offer
1	Content	Offer on mobiles
2	To	john
2	Type	sale
2	Content	Redmi sale
3	To	jill
3	Type	order
3	Content	Order delivered
4	To	mike
4	Type	sale
4	Content	Clothes sale



Type	Count
offer	1
sale	2
order	1

Implementing a MapReduce Job

Map

A class where the map logic is implemented

Reduce

A class where the reduce logic is implemented

Main

A driver program that sets up the job

Implementing a MapReduce Job

Map

A class where the map logic is implemented

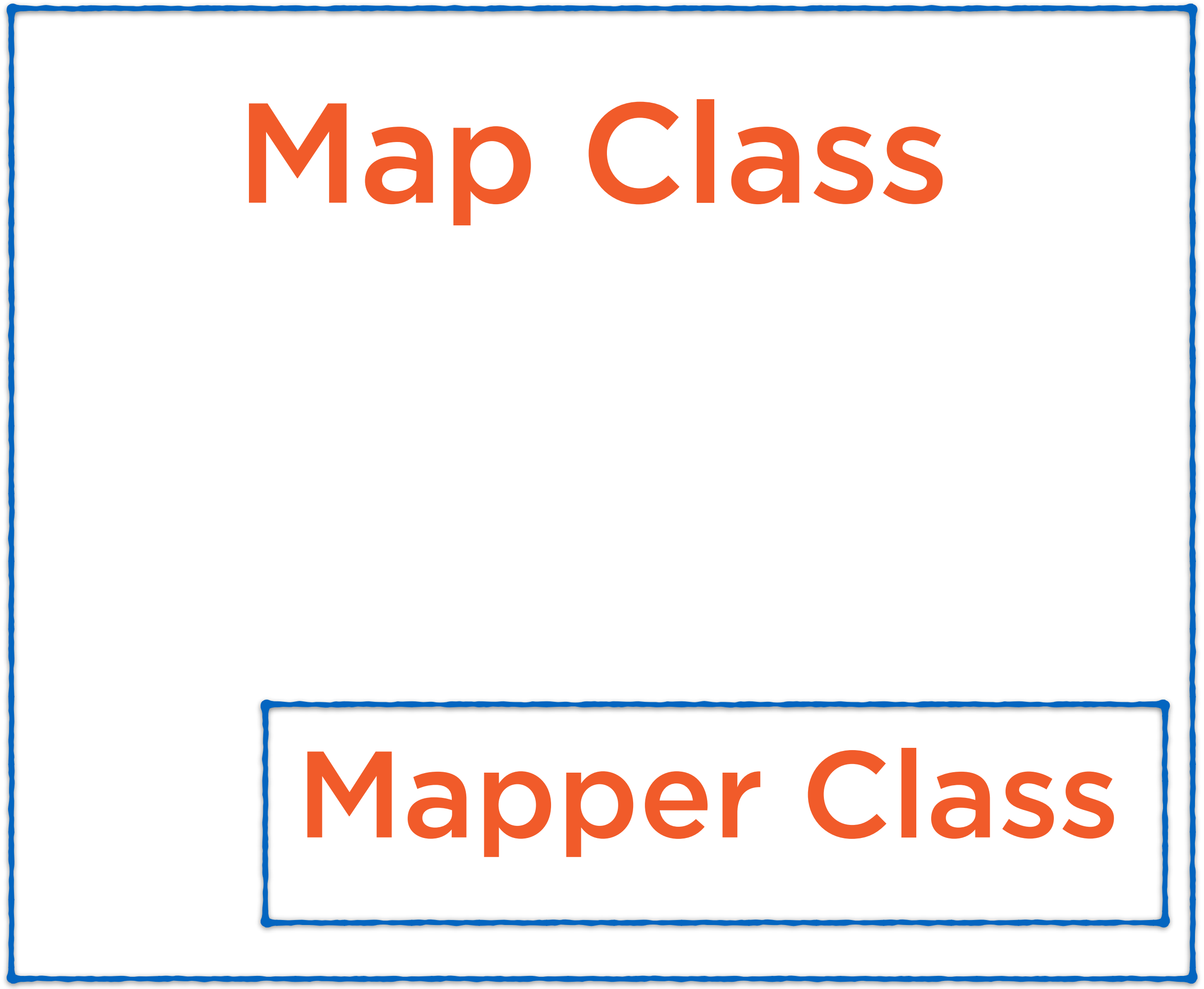
Reduce

A class where the reduce logic is implemented

Main

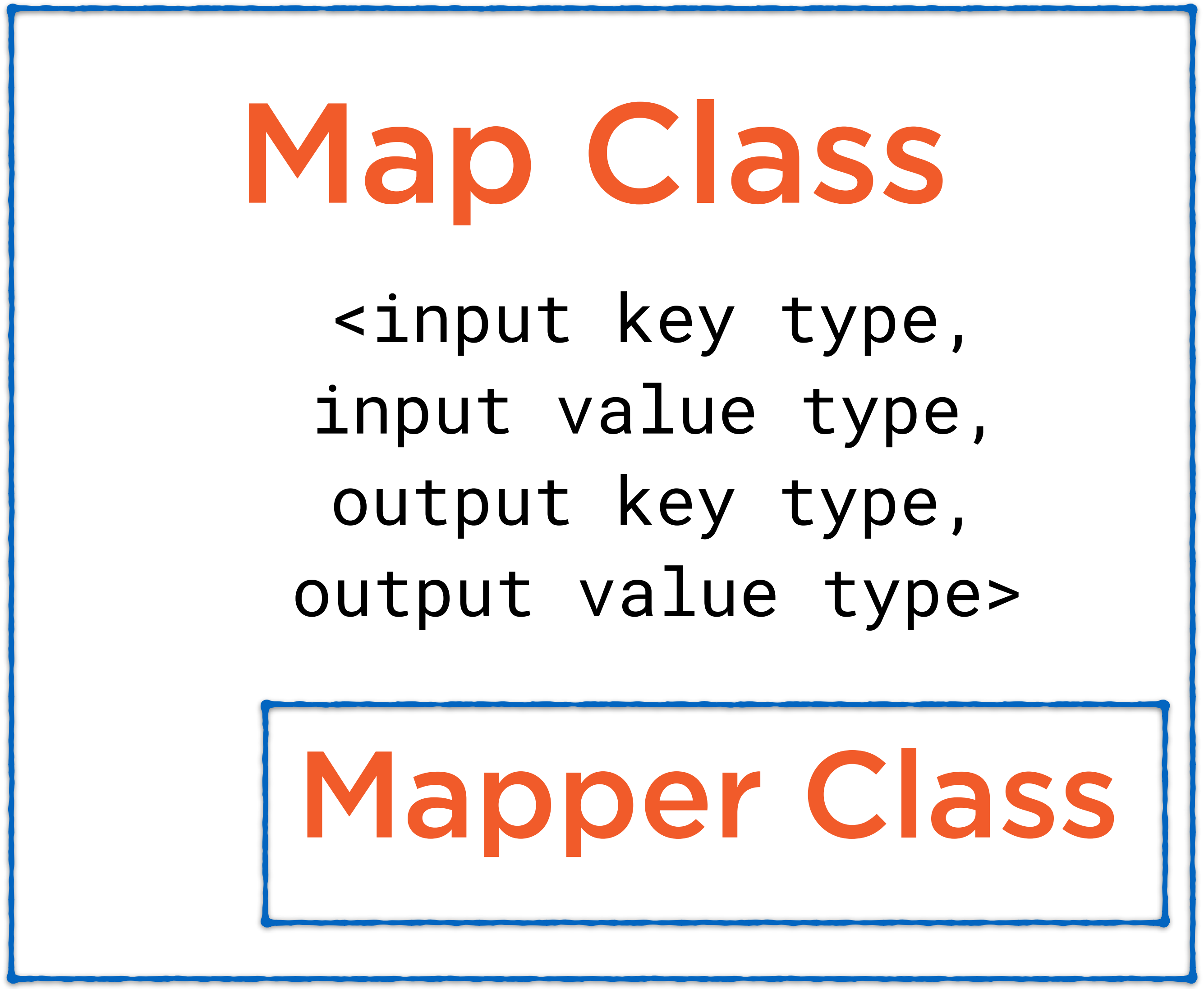
A driver program that sets up the job

The Map Step



The map logic is implemented in a class that extends the Mapper Class

The Map Step



Map Class

```
<input key type,
input value type,
output key type,
output value type>
```

Mapper Class

This is a generic class, with 4 type parameters

Implementing a MapReduce Job

Map

A class where the map logic is implemented

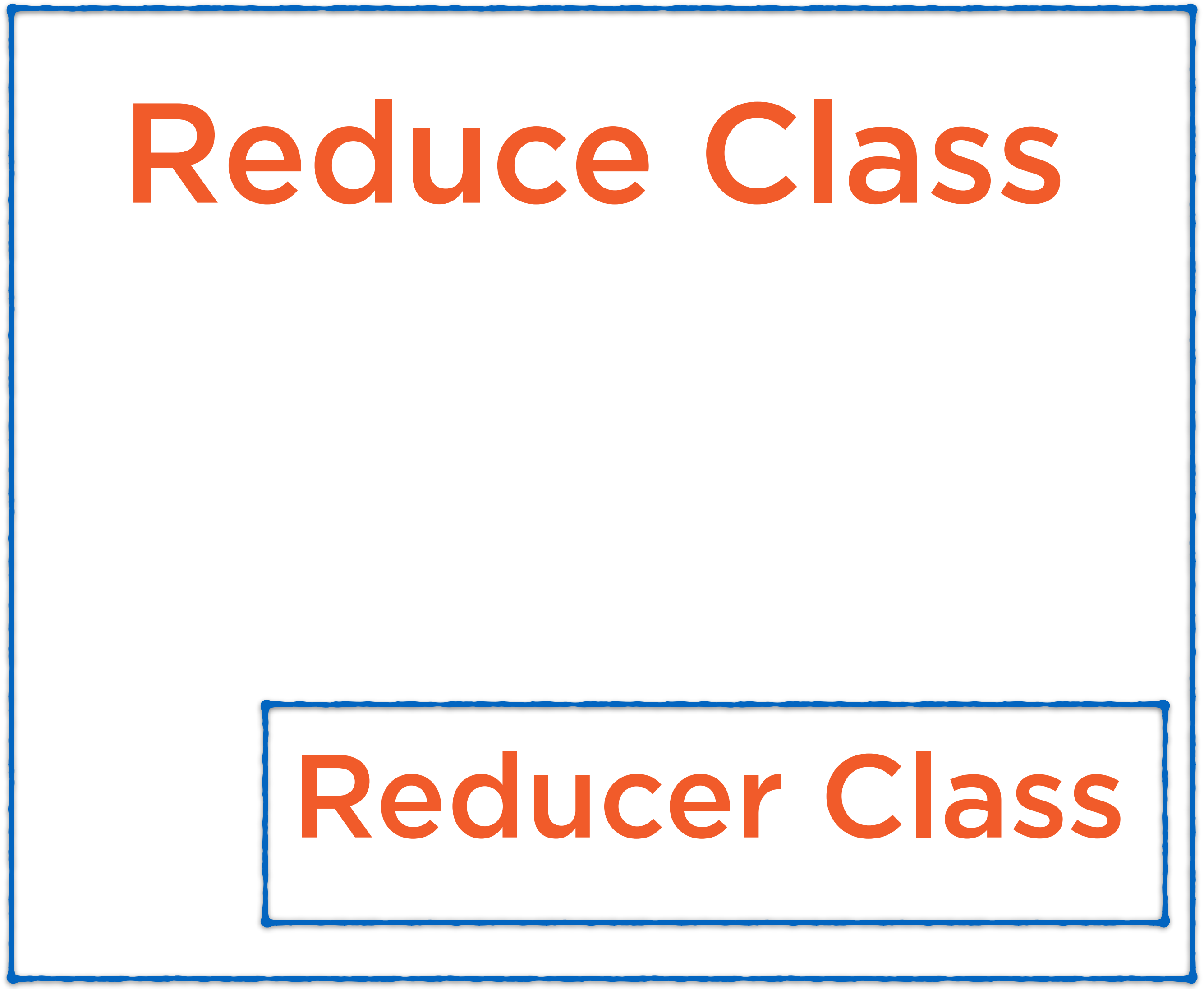
Reduce

A class where the reduce logic is implemented

Main

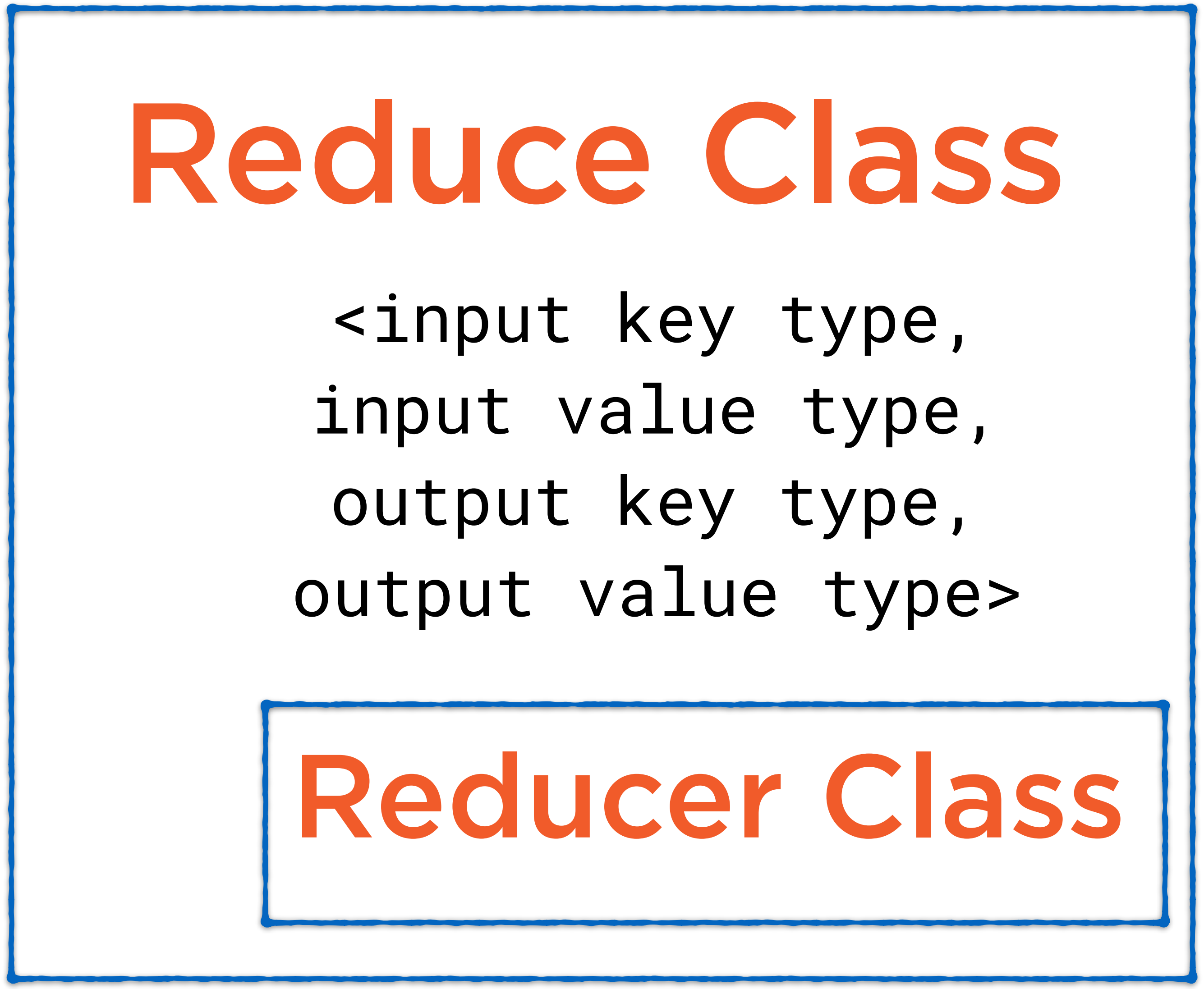
A driver program that sets up the job

The Reduce Step



The reduce logic is implemented in a class that extends the Reducer Class

The Reduce Step



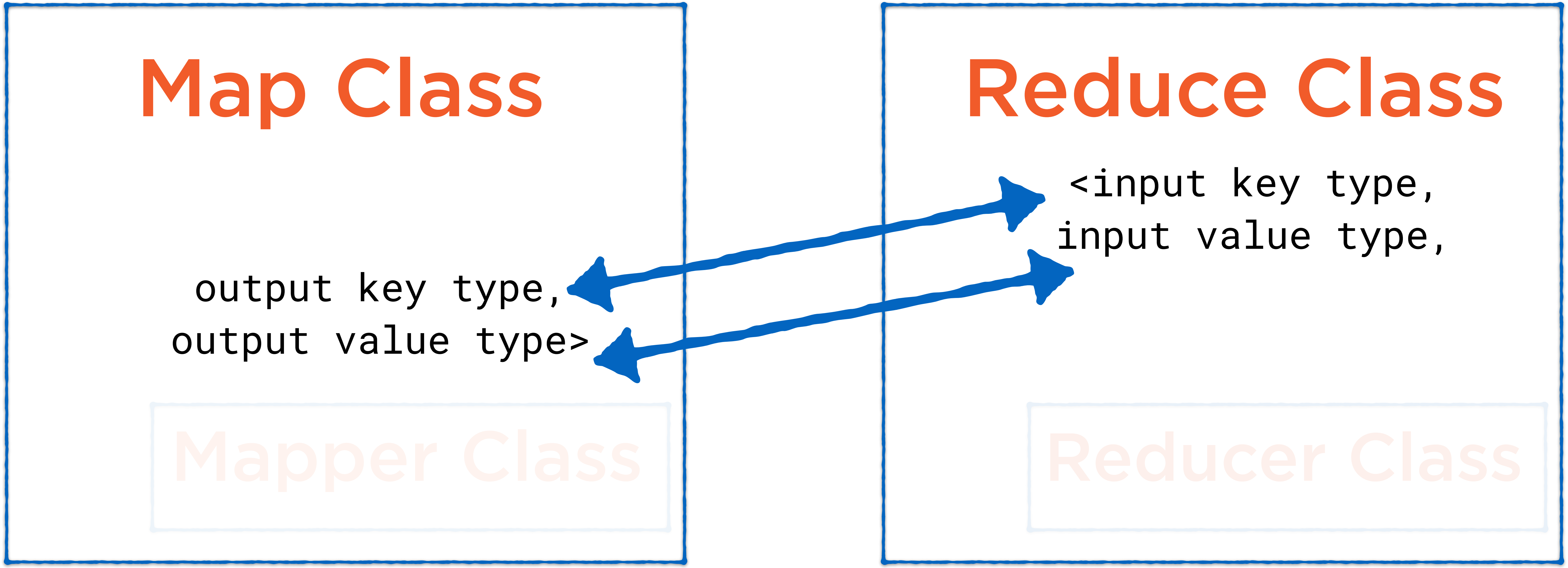
Reduce Class

```
<input key type,
input value type,
output key type,
output value type>
```

Reducer Class

This is also a generic class, with 4 type parameters

Matching Data Types



Map Class

```
output key type,
output value type>
```

Mapper Class

Reduce Class

```
<input key type,
input value type,
```

Reducer Class

Implementing a MapReduce Job

Map

A class where the map logic is implemented

Reduce

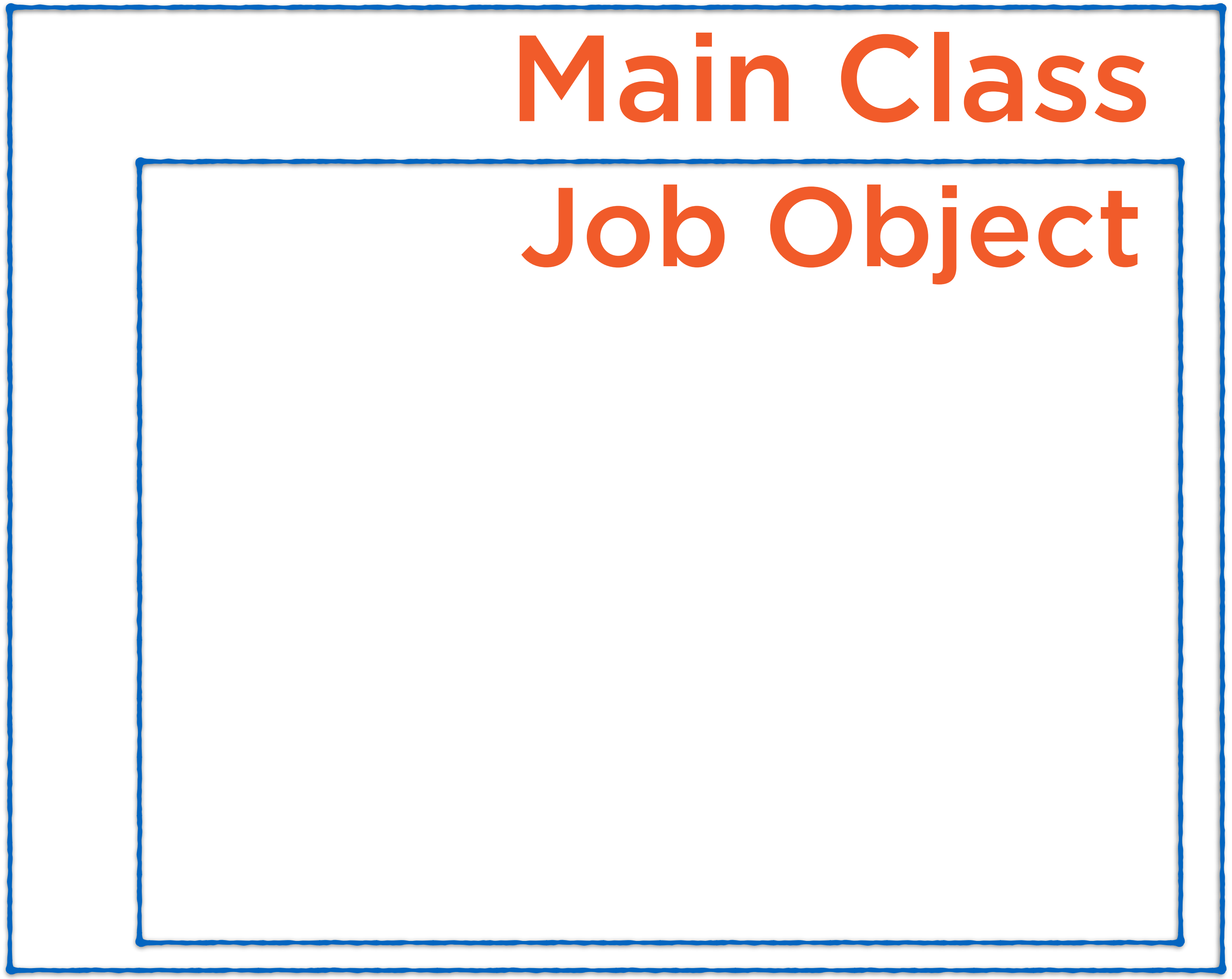
A class where the reduce logic is implemented

Main

A driver program that sets up the job

Setting up the Job

The Mapper and Reducer classes are used by a Job that is configured in the Main Class

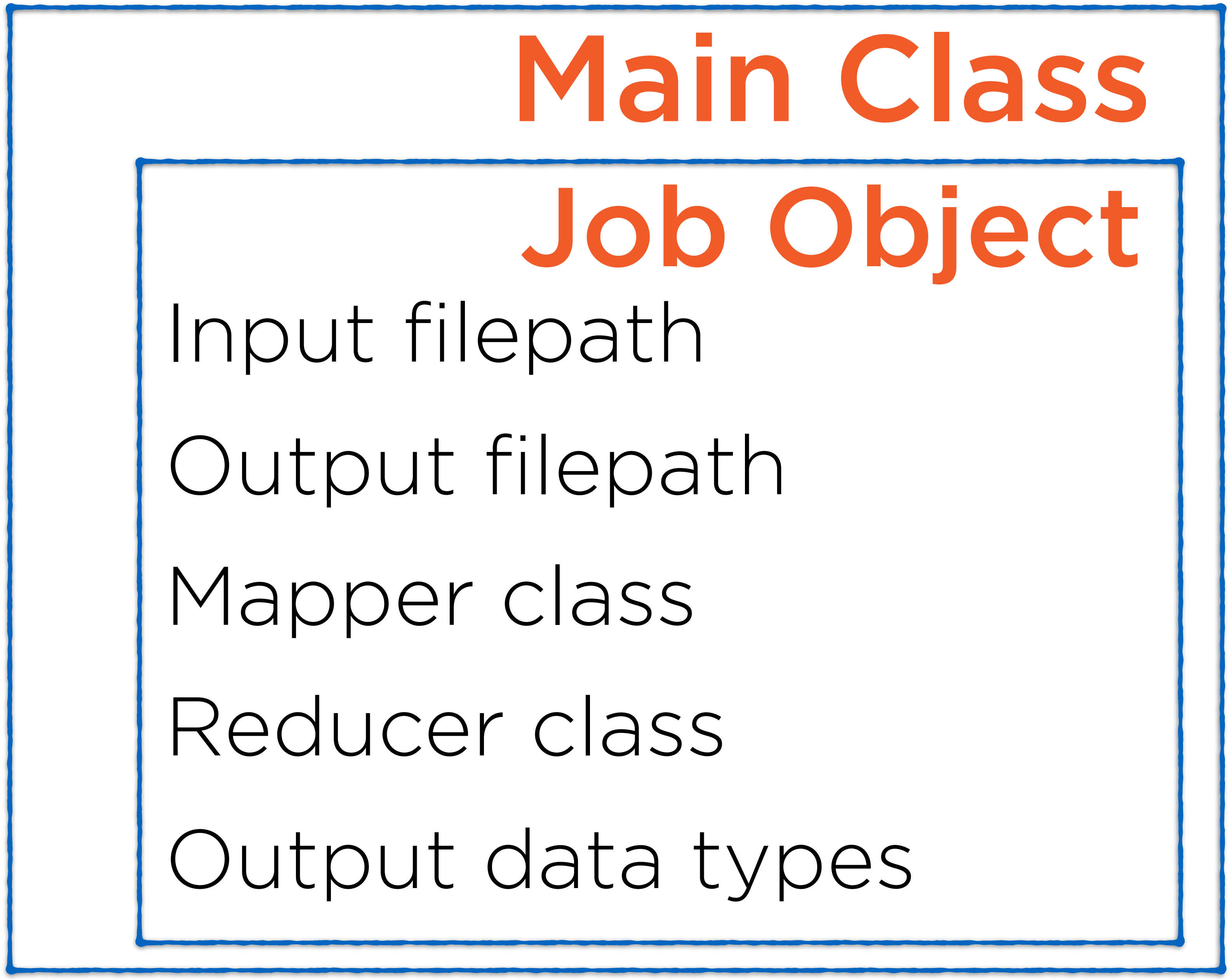


Main Class

Job Object

Setting up the Job

The Job has a bunch of properties that need to be configured



Main Class

Job Object

Input filepath

Output filepath

Mapper class

Reducer class

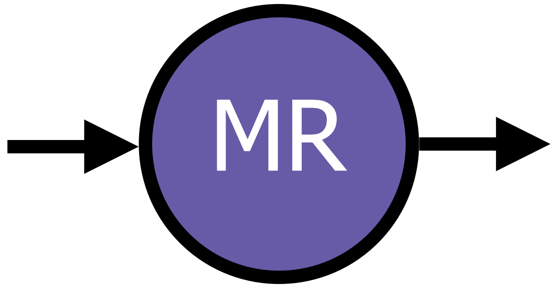
Output data types

Demo

Set up a MapReduce job which runs on the census HBase table

Marital Status Count

Id	Column	Value
1	name	mike
1	marital_status	married
1	employed	yes
2	name	john
2	marital_status	divorced
2	field	real estate
3	name	jill
3	marital_status	married
3	employed	yes
4	name	ben
4	marital_status	separated
4	education_level	undergrad



Marital Status	Count
married	2
divorced	1
separated	1

Marital Status Count

Id	Column	Value
1	name	mike
1	marital_status	married
1	employed	yes
2	name	john
2	marital_status	divorced
2	field	real estate
3	name	jill
3	marital_status	married
3	employed	yes
4	name	ben
4	marital_status	separated
4	education_level	undergrad

<married, 1>
<divorced, 1>
<married, 1>
<separated, 1>

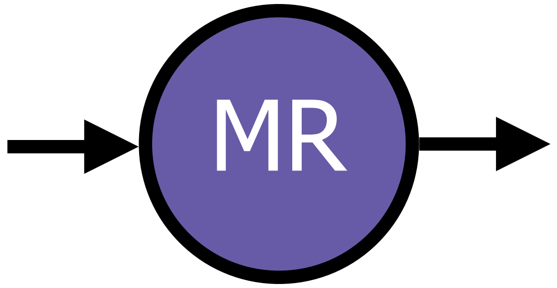
Marital Status Count

<married, 1>
<divorced, 1>
<married, 1>
<separated, 1>

Marital Status	Count
married	2
divorced	1
separated	1

Marital Status Count

Id	Column	Value
1	name	mike
1	marital_status	married
1	employed	yes
2	name	john
2	marital_status	divorced
2	field	real estate
3	name	jill
3	marital_status	married
3	employed	yes
4	name	ben
4	marital_status	separated
4	education_level	undergrad



Marital Status	Count
married	2
divorced	1
separated	1

Summary

A brief overview of the Hadoop MapReduce framework

Ran MapReduce using HBase tables as a source and a sink of data