

Overview

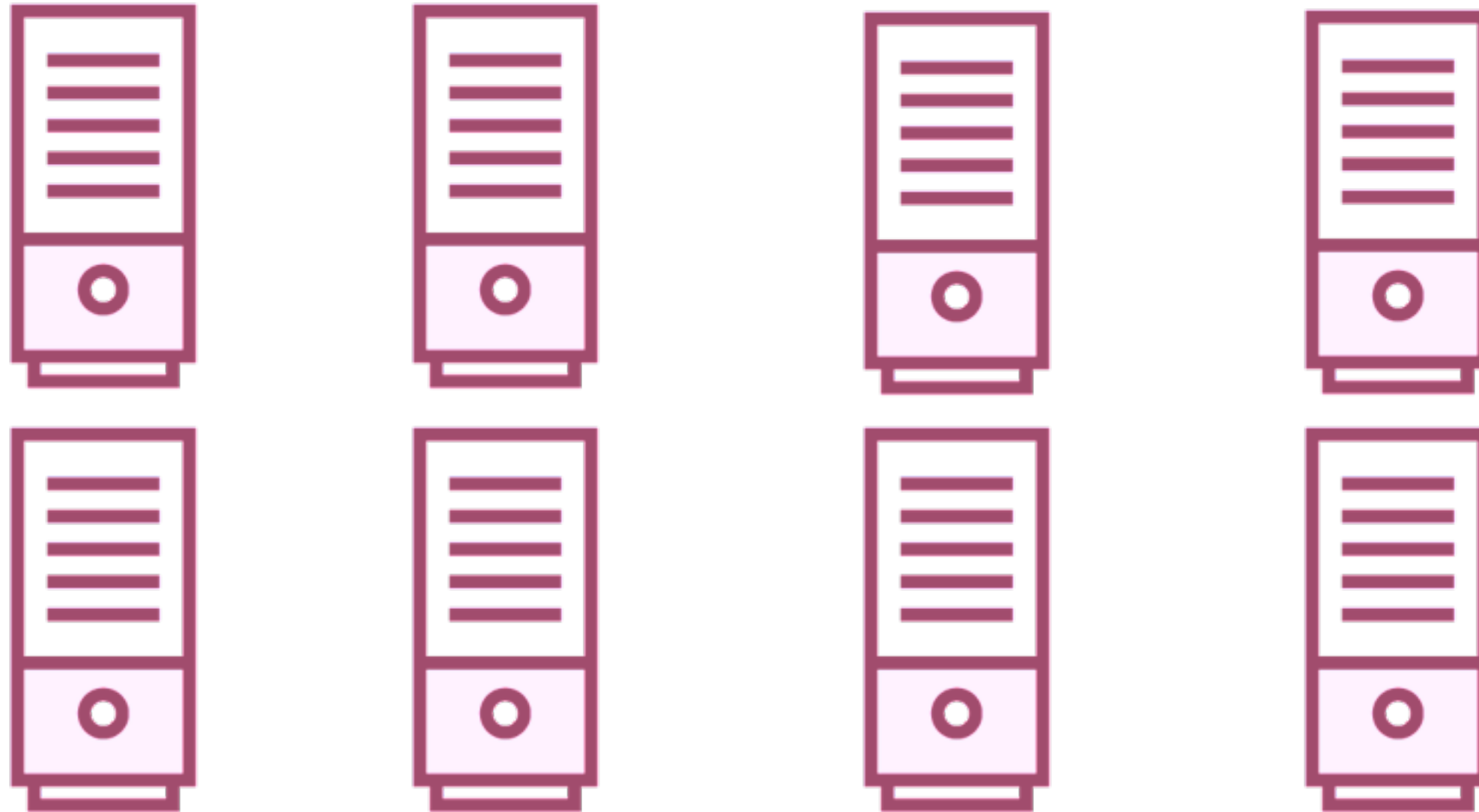
Setting up a MapReduce job for a simple counting task

Submitting a MapReduce job to Hadoop and monitoring it

MapReduce

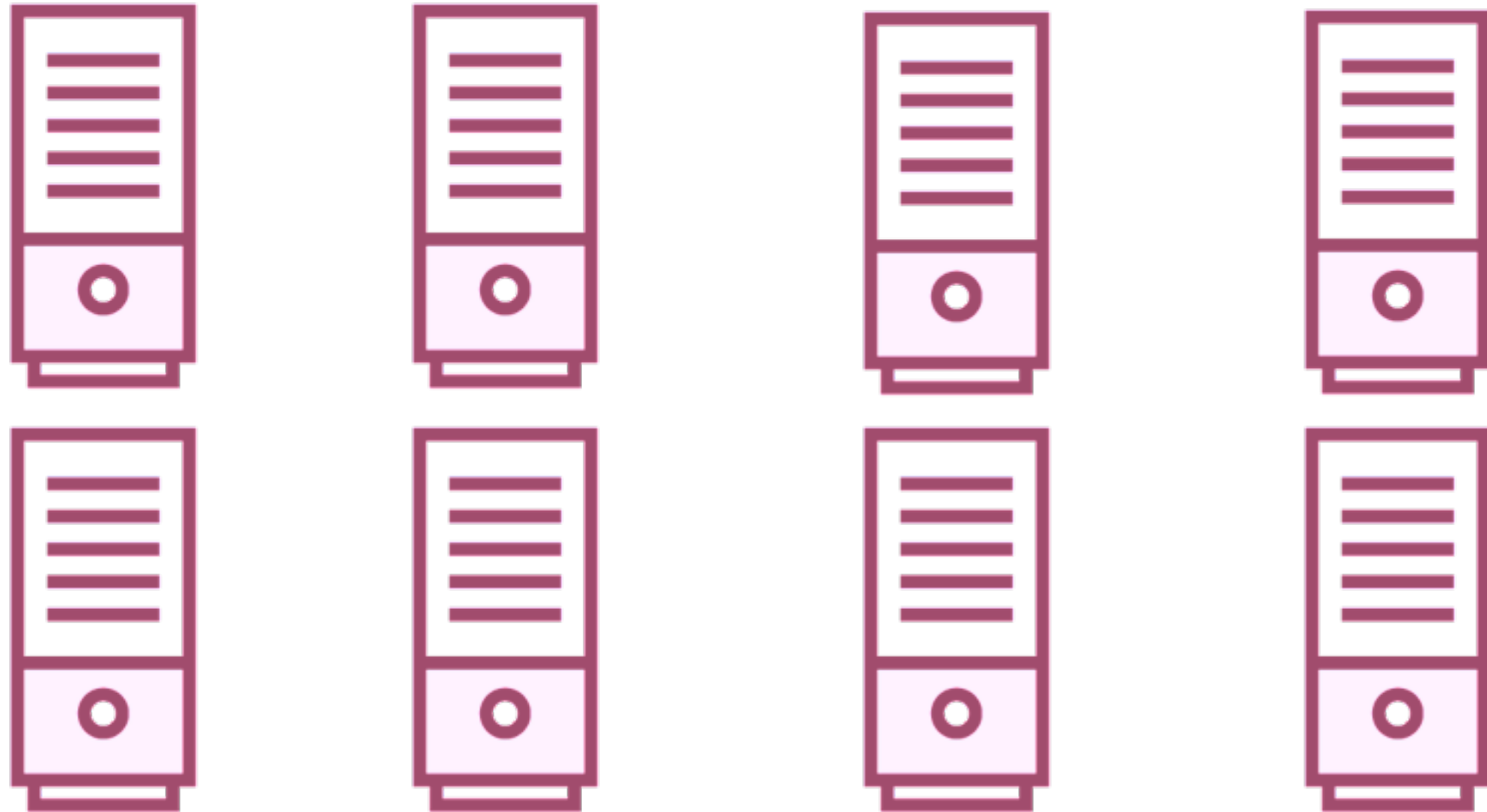
Processing huge amounts of data

MapReduce



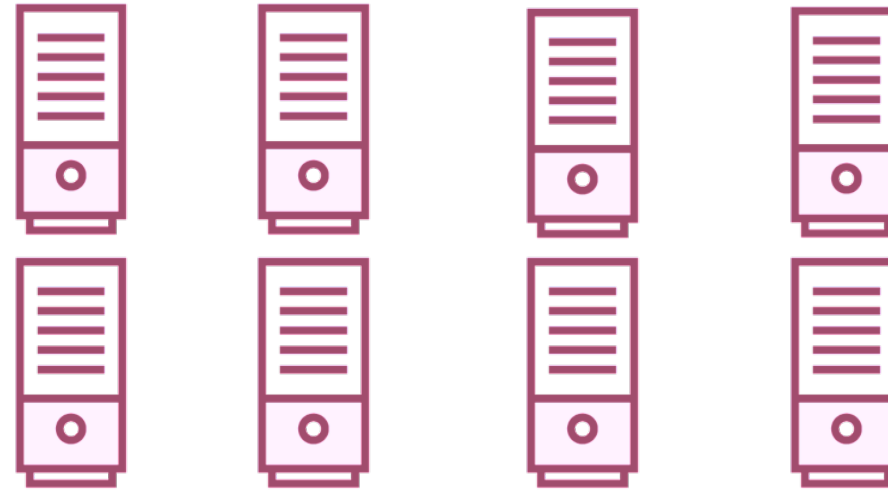
Requires running processes on many machines

MapReduce



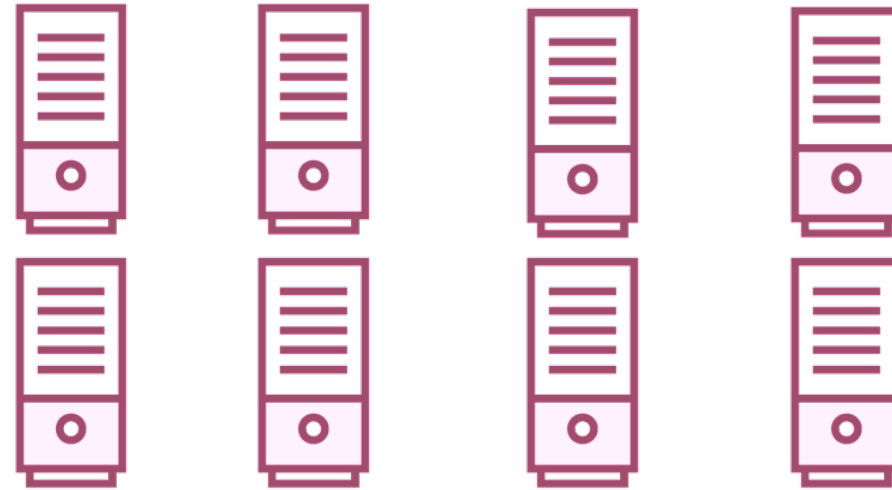
A distributed system

MapReduce



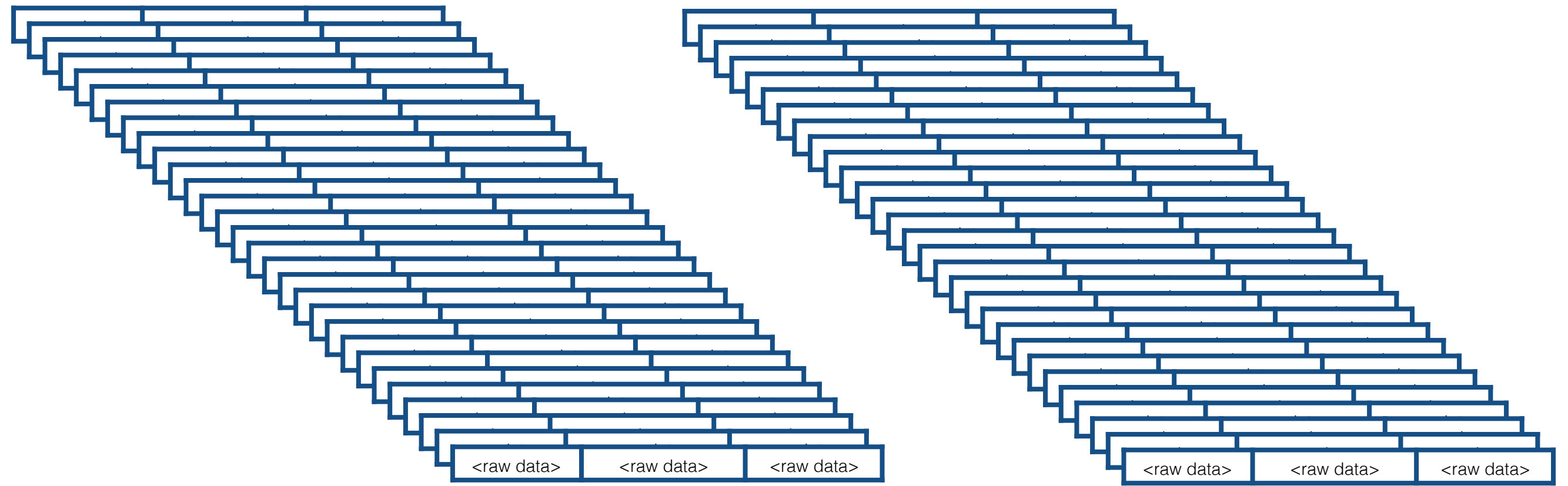
**MapReduce is a programming
paradigm**

MapReduce



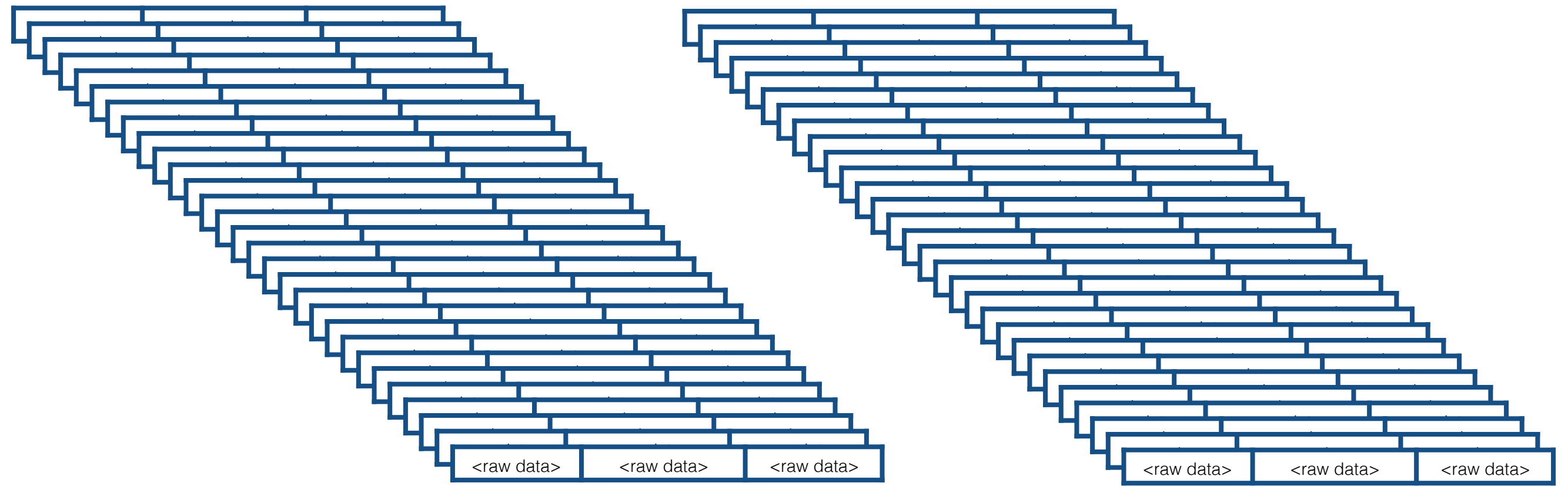
**Takes advantage of the inherent
parallelism in data processing**

MapReduce



**Modern systems generate millions of
records of raw data**

MapReduce

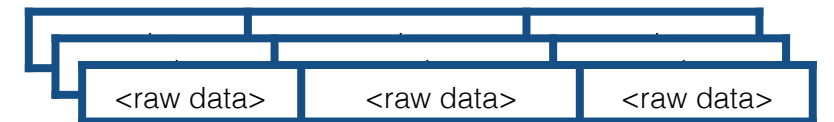
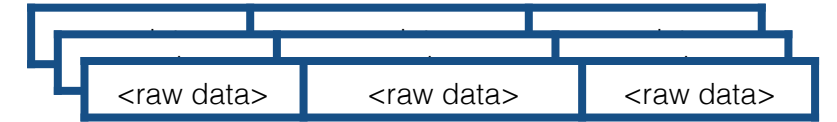
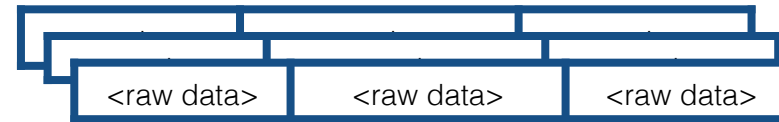
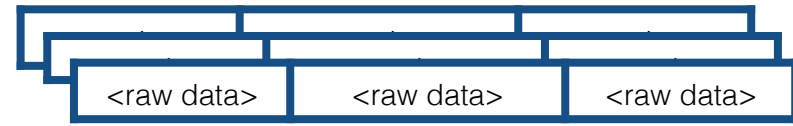


A task of this scale is processed in
two stages

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

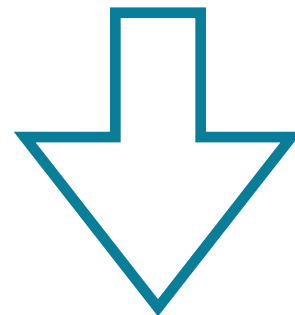
Hadoop does the rest -
behind the scenes

map

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

map

One Record

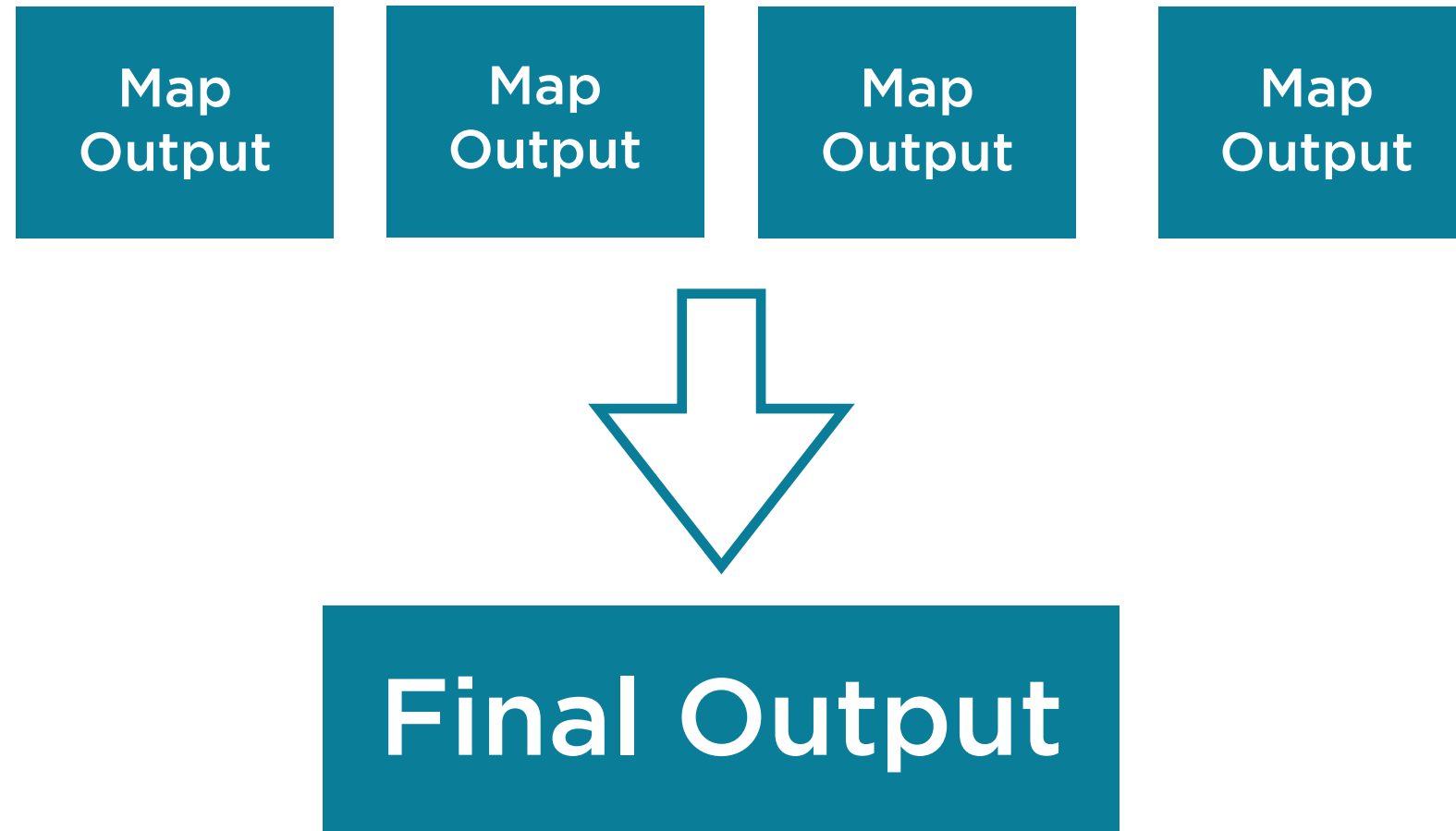


Key-Value Output

reduce

**An operation to
combine the results of
the map step**

reduce



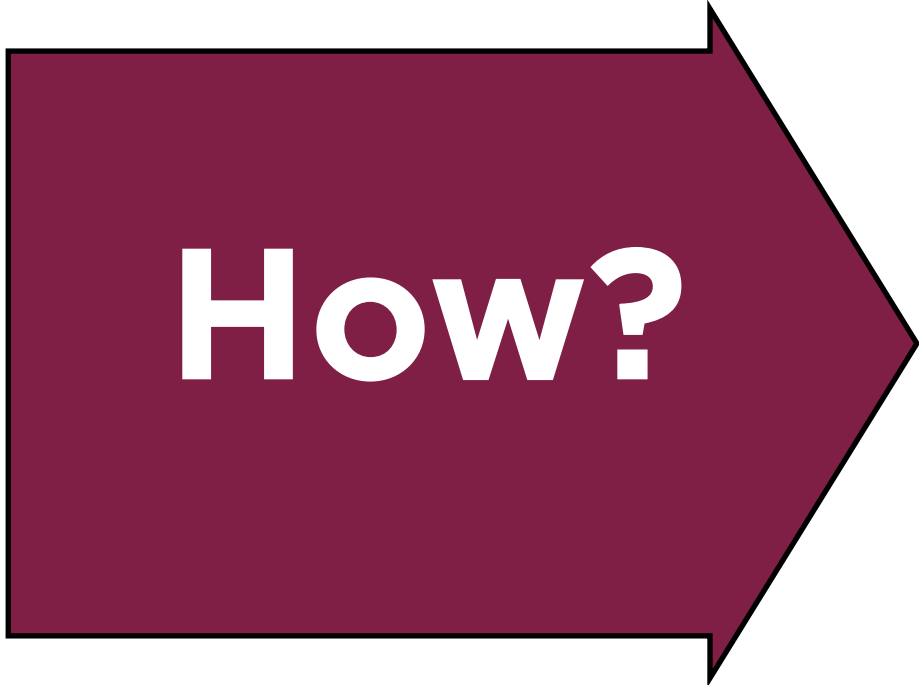
map A step that can be
performed in parallel

reduce A step to combine the
intermediate results

Counting Word Frequencies

Consider a large text file

Twinkle twinkle little star
How I wonder what you are
Up above the world so high
Like a diamond in the sky
Twinkle twinkle little star
How I wonder what you are
.....



Word	Frequency
above	14
are	20
how	21
star	22
twinkle	32
...	..

Twinkle twinkle little star
How I wonder what you are
Up above the world so high
Like a diamond in the sky
Twinkle twinkle little star
How I wonder what you are
.....

MapReduce Flow

**The raw data is really large
(potentially in PetaBytes)**

**It's distributed across many
machines in a cluster**

**Each machine holds a partition of
data**

MapReduce Flow

Twinkle twinkle little star
How I wonder what you are



Up above the world so high
Like a diamond in the sky

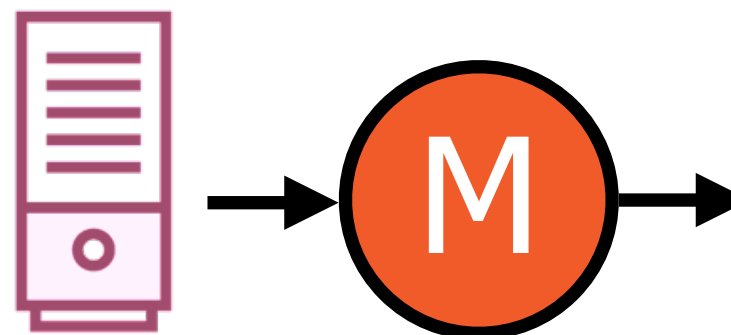
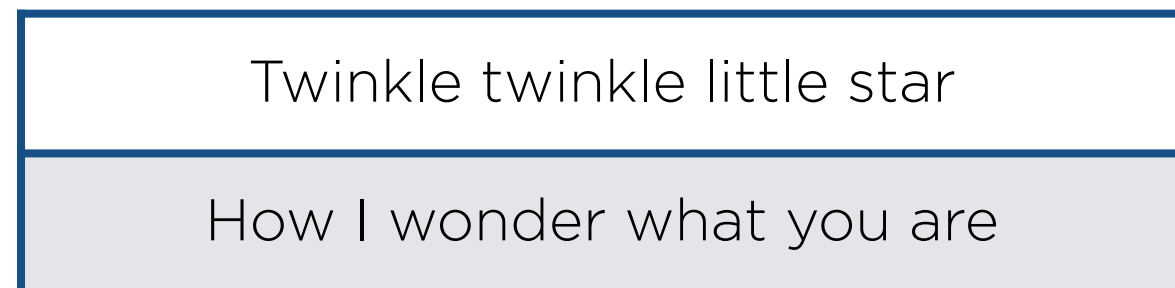
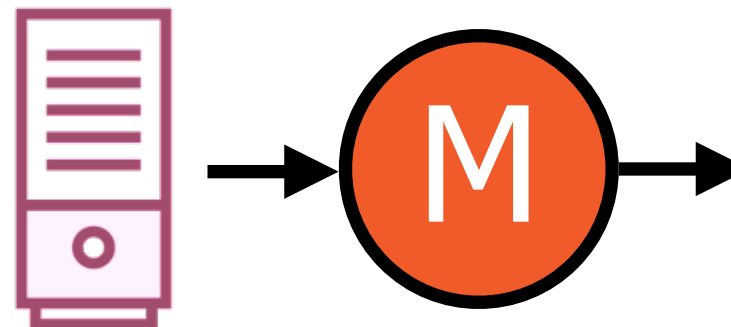
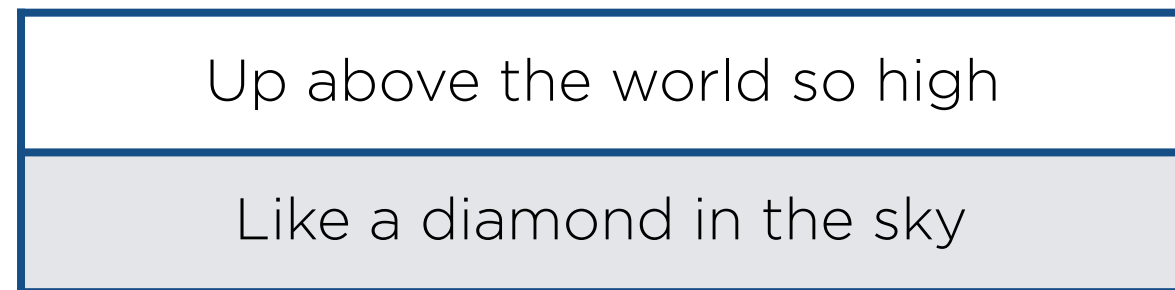
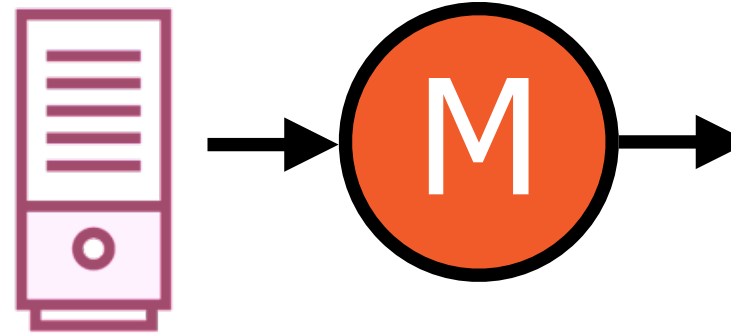
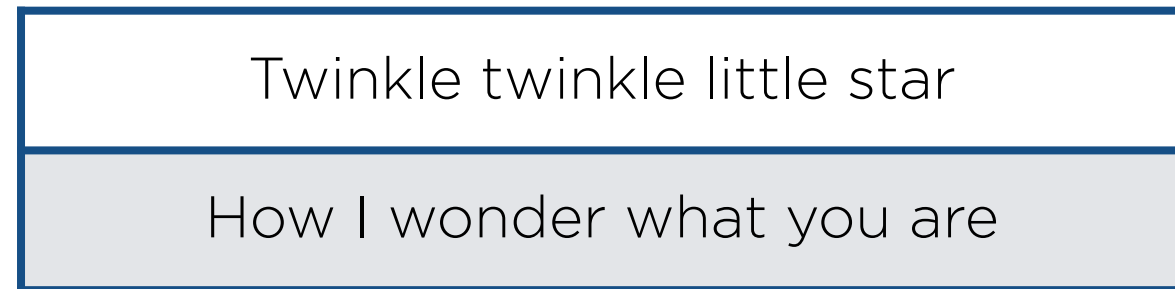


Twinkle twinkle little star
How I wonder what you are



Each partition is given to a different process i.e. to mappers

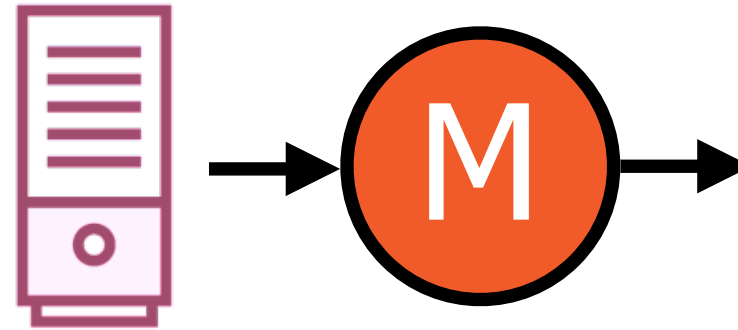
MapReduce Flow



**Each mapper
works in parallel**

Map Flow

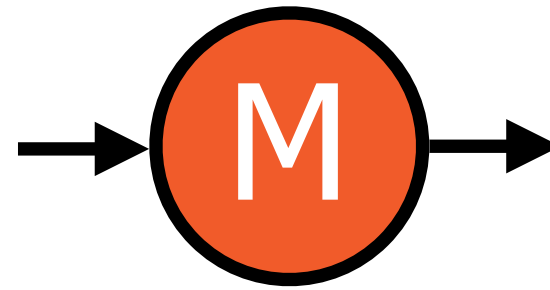
Twinkle twinkle little star
How I wonder what you are



**Within each mapper, the rows
are processed serially**

Map Flow

Twinkle twinkle little star
How I wonder what you are



Word	# Count
------	---------

{twinkle, 1}

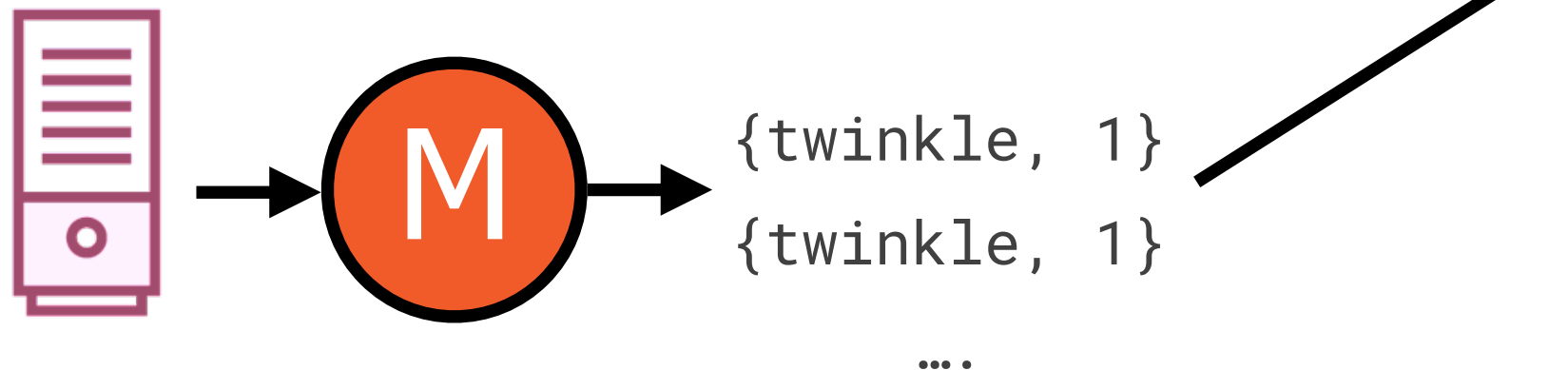
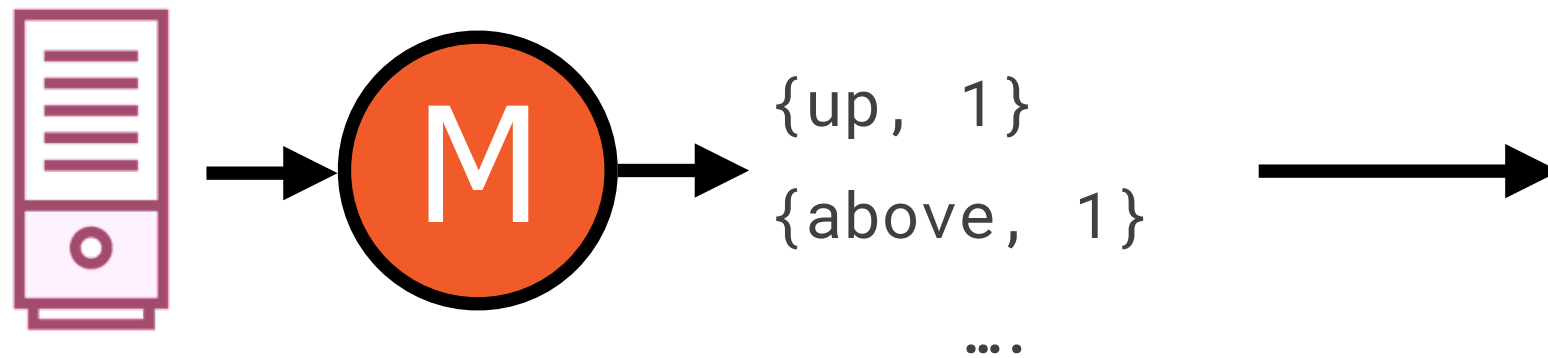
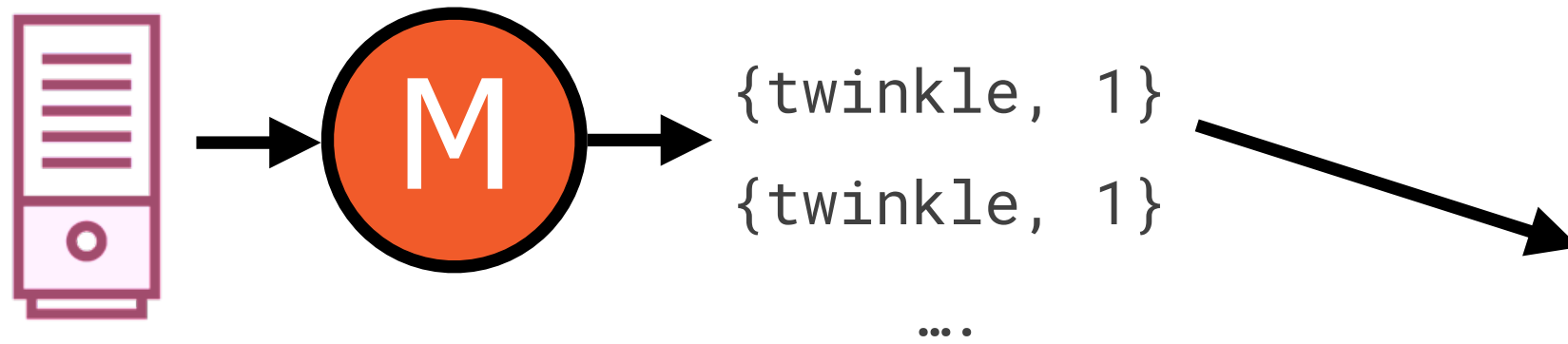
{twinkle, 1}

{little, 1}

{star, 1}

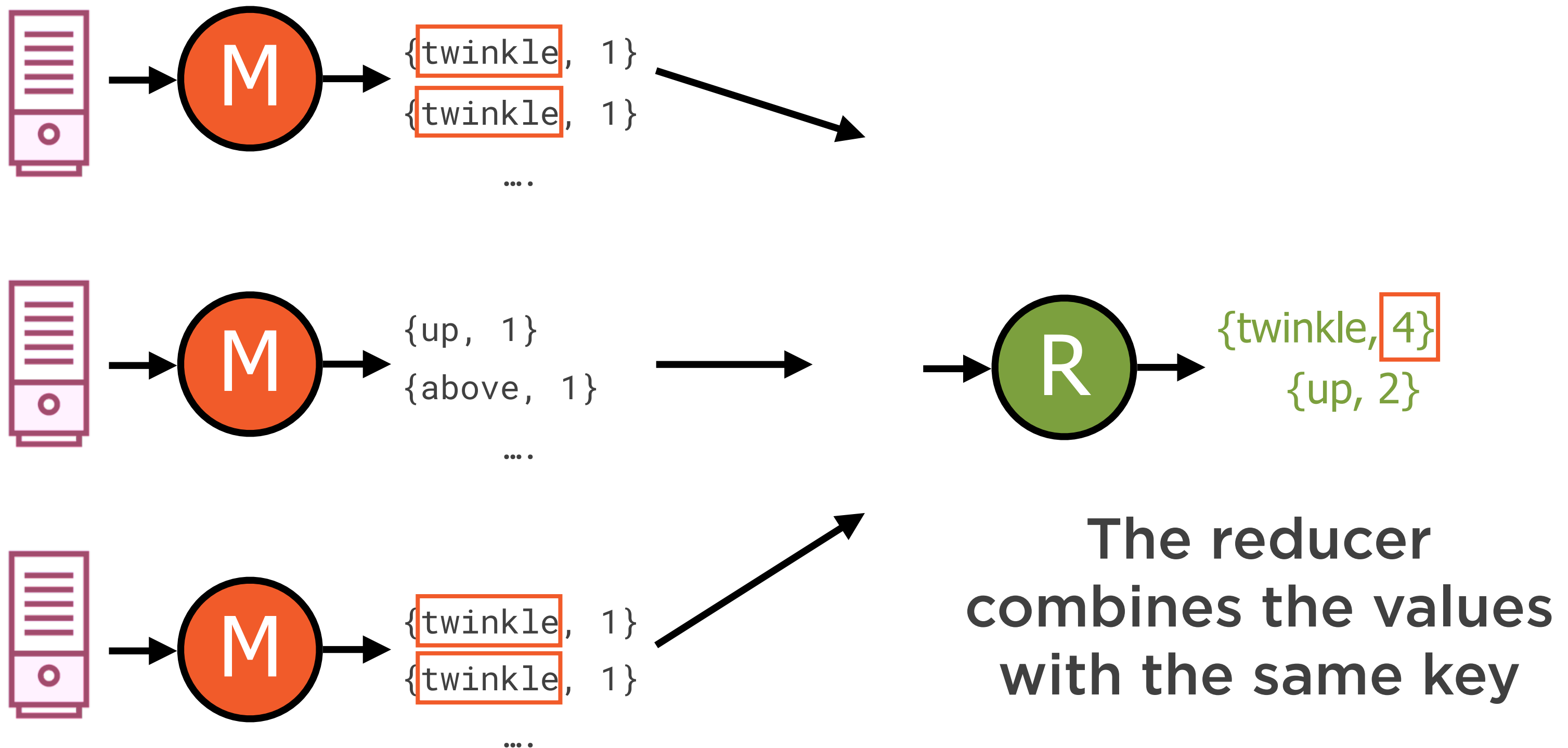
Each row emits {key, value} pairs

Reduce Flow



**The results are
passed on to another
process i.e. a reducer**

Reduce Flow

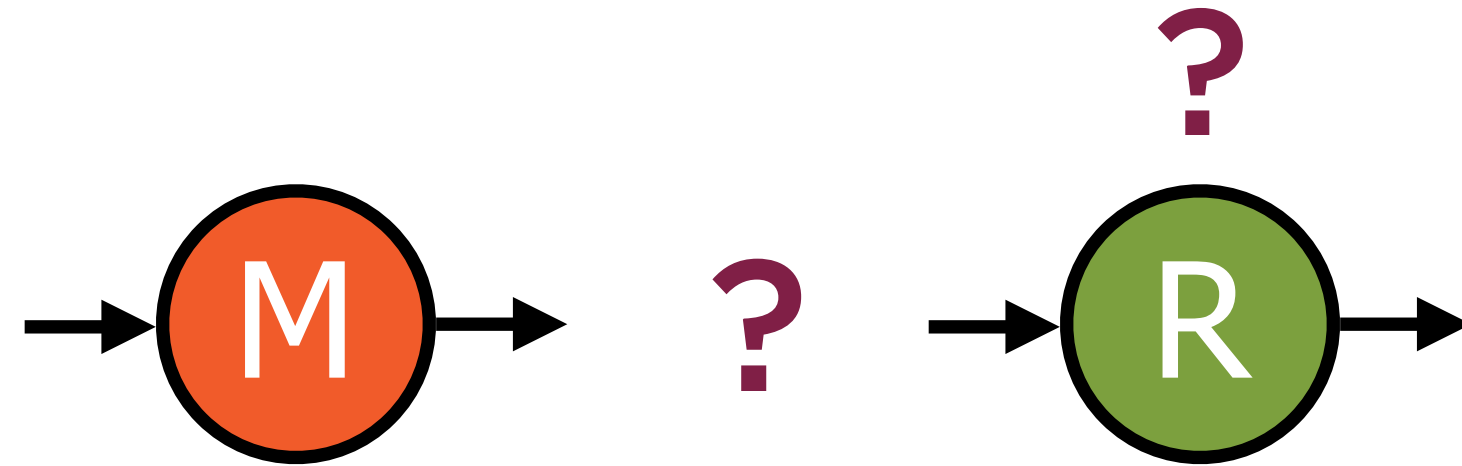


Key Insight Behind MapReduce



Many data processing tasks can be expressed in this form

Answer Two Questions



1. What {key, value} pairs should be emitted in the map step?
2. How should values with the same key be combined?

Counting Word Frequencies

Twinkle twinkle little star
How I wonder what you are
Up above the world so high
Like a diamond in the sky



For each word
in each line

```
{twinkle, 1}  
{twinkle, 1}  
{little, 1}  
{star, 1}  
..  
...
```

Word	Count
twinkle	2
little	1
...	...
...	...
...	...
...	...



Answer these to
parallelize any task :)

Implementing in Java



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 in Java



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

Map Step

Map Class

Mapper Class

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

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 in Java



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 in Java



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

Reduce Step

Reduce Class

Reducer Class

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

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

**The output types of the Mapper should
match the input types of the Reducer**

Implementing in Java



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 in Java



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



```
graph TD; subgraph Main_Class [Main Class]; Job_Object [Job Object]; end
```

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

Running a MapReduce job

Monitoring progress in the UI

**Understanding the information
presented in the UI**

Summary

Setting up a MapReduce job for a simple counting task

Submitting a MapReduce job to Hadoop and monitoring it