

Controlling Parallelism in Map and Reduce Phases



Janani Ravi

CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Understand what happens behind the scenes in a MapReduce task

Control the number of mappers and reducers

Improve parallelism by using a Combiner after the map phase

Know the constraints involved in using combiners

Key Insight Behind MapReduce

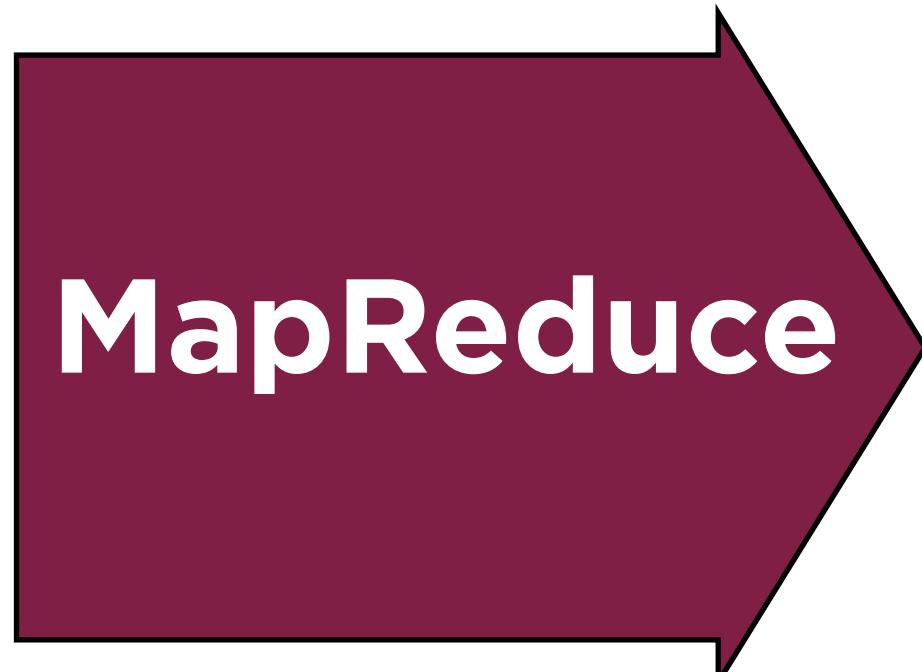


Many data processing tasks can be expressed in this form

Building a User-ViewCount Map

Data is originally captured in this form

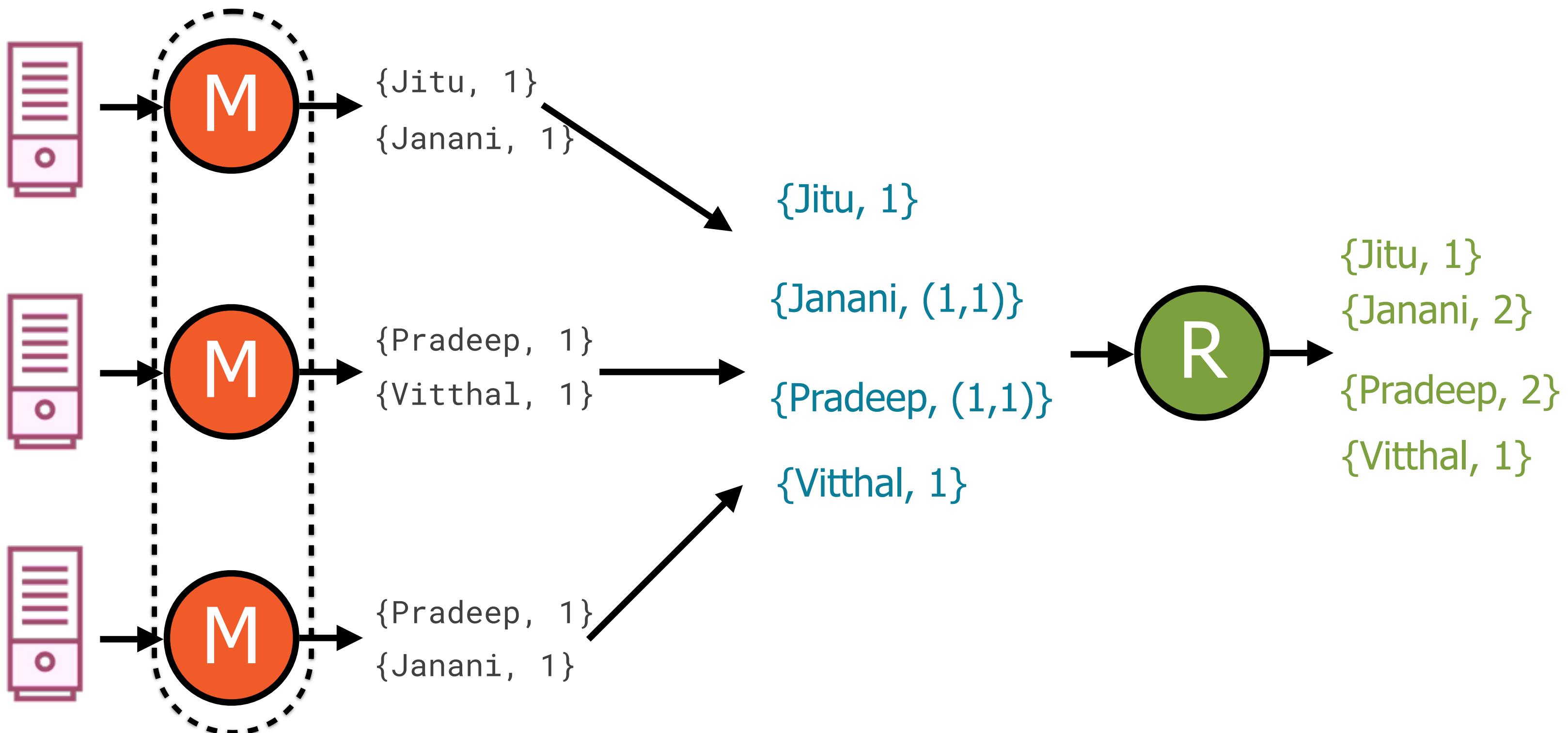
View ID	From Member	To Member
1	Janani	Jitu
2	Swetha	Janani
3	Shreya	Pradeep
4	Jitu	Vitthal
5	Shreya	Janani
6	Jitu	Pradeep



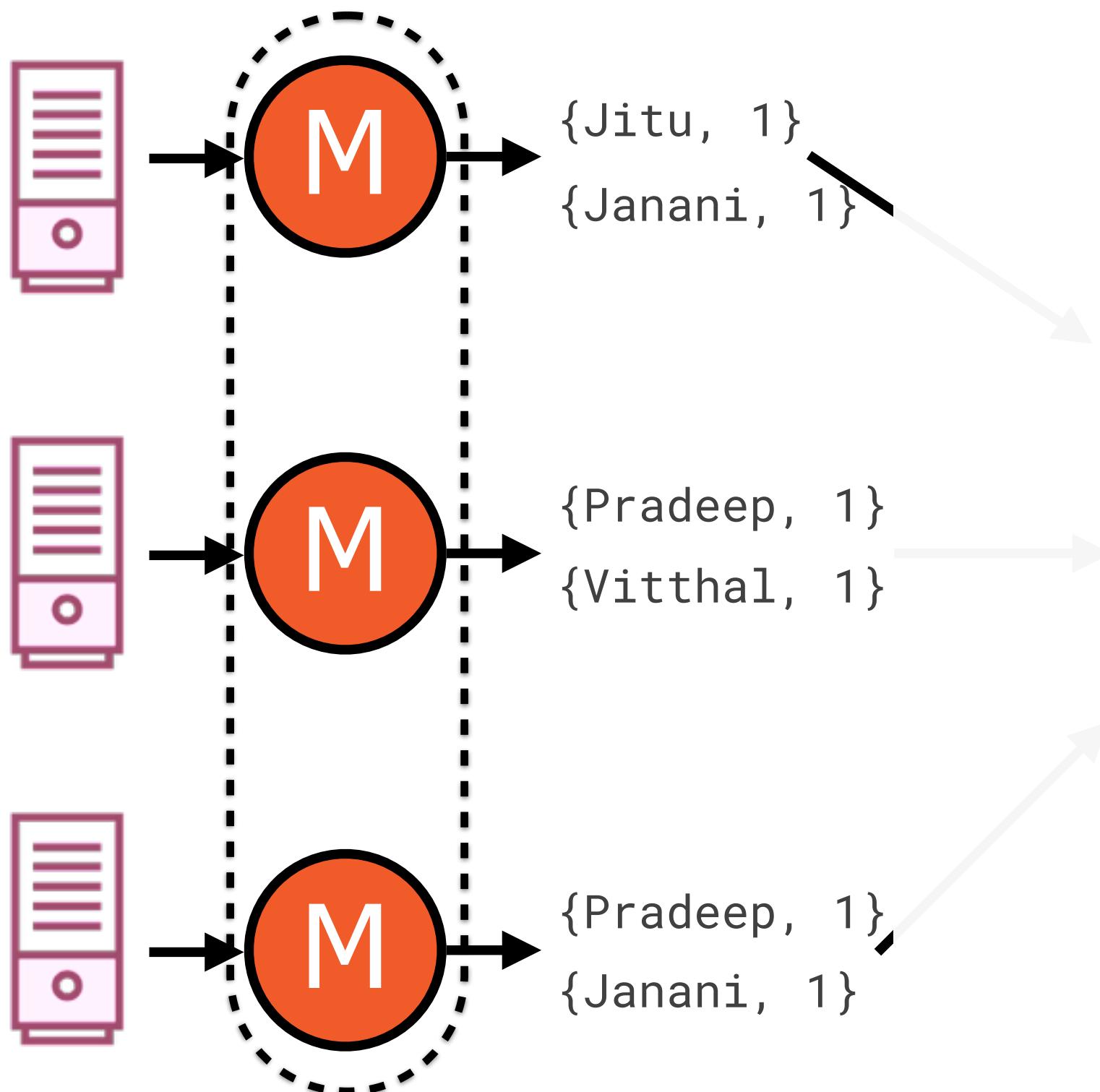
User-ViewCount Map

Member	# Profile Views
Janani	50
Swetha	15
Vitthal	22
Shreya	23
Jitu	32
Pradeep	10

Typical MapReduce Flow

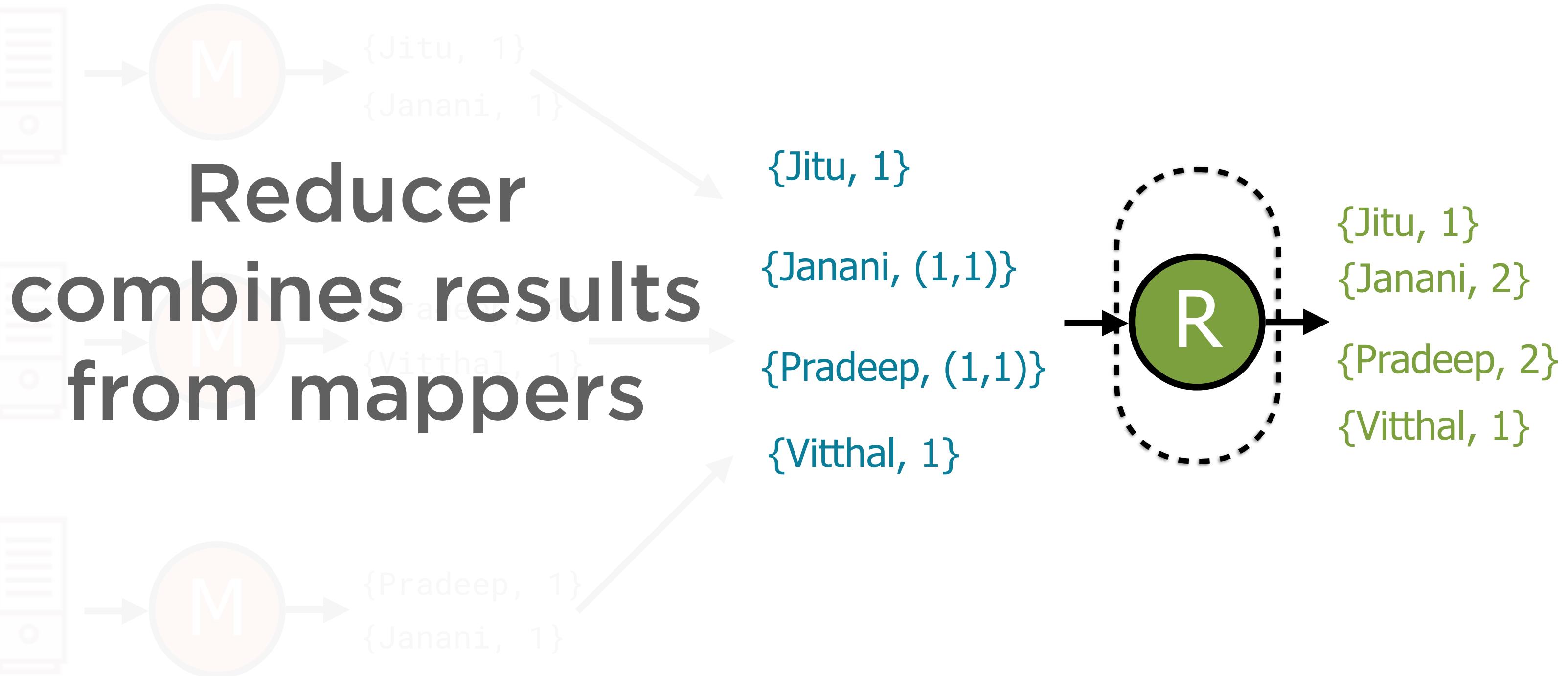


Typical MapReduce Flow



**Mappers that
work in parallel
on data
partitions**

Typical MapReduce Flow



Typical MapReduce Flow



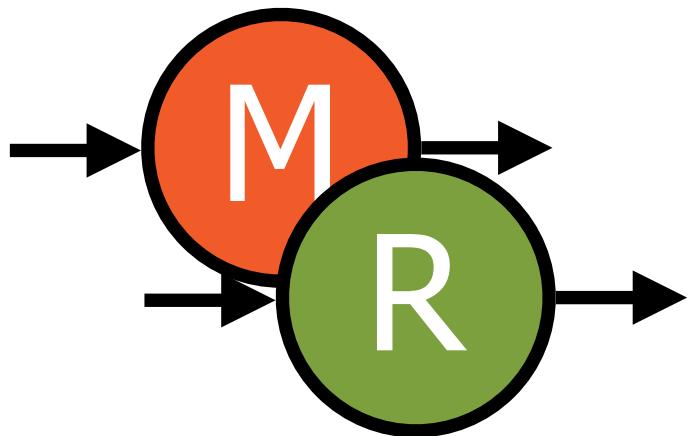
Mappers and reducers are
“processes” that run on
individual nodes in a cluster

Typical MapReduce Flow



The programmer defines the logic of the Map and Reduce steps in a MapReduce Job

Typical MapReduce Flow



The MapReduce job is submitted to a cluster

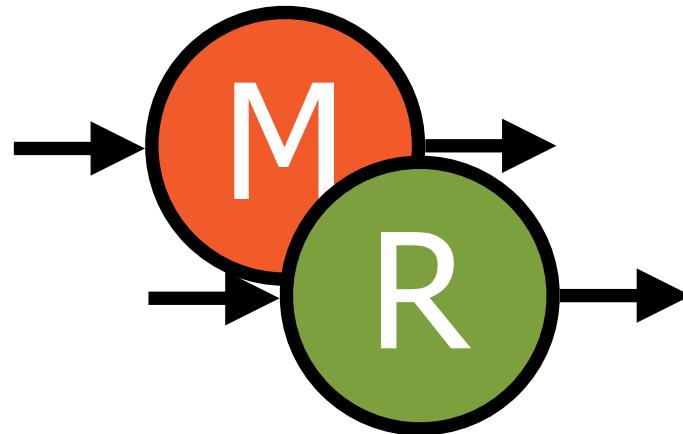
Typical MapReduce Flow

The cluster is managed by a piece of software called a Cluster Manager

In the case of Hadoop 2.0, YARN is the cluster manager



Typical MapReduce Flow



The cluster manager launches
mapper and reducer processes...

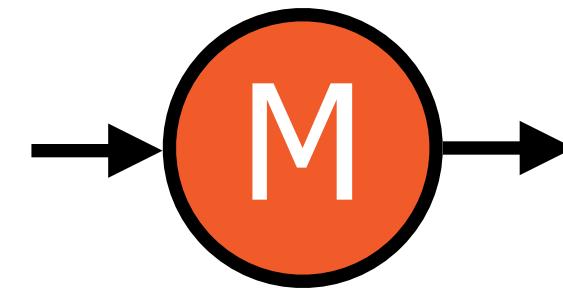
... on different nodes in the cluster



How many mappers
and reducers are
launched?

Number of Mappers/Reducers

Number of mappers = number of data partitions



Number of Mappers/Reducers

View ID	From Member	To Member
1	Janani	Jitu
2	Swetha	Janani
3	Shreya	Pradeep
4	Jitu	Vithal
5	Shreya	Janani
6	Jitu	Pradeep

Number of Mappers/Reducers

View ID	From Member	To Member
1	Janani	Jitu
2	Swetha	Janani
3	Shreya	Pradeep
4	Jitu	Vithal
5	Shreya	Janani
6	Jitu	Pradeep

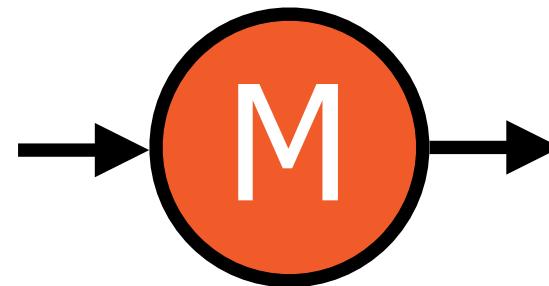
View ID	From Member	To Member
1	Janani	Jitu
2	Swetha	Janani

View ID	From Member	To Member
3	Shreya	Pradeep
4	Jitu	Vithal

View ID	From Member	To Member
5	Shreya	Janani
6	Jitu	Pradeep

Number of Mappers/Reducers

Number of mappers = number of data partitions



number of data partitions

Decided by the storage system

Cannot be controlled by the user

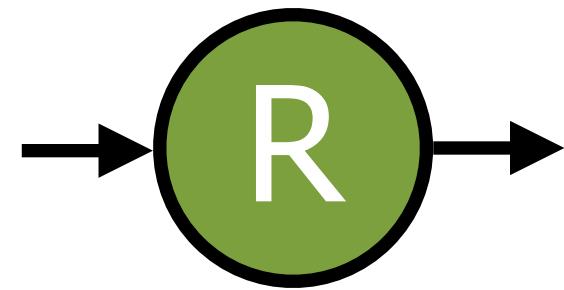
Storage system for Hadoop is HDFS

Number of Mappers/Reducers

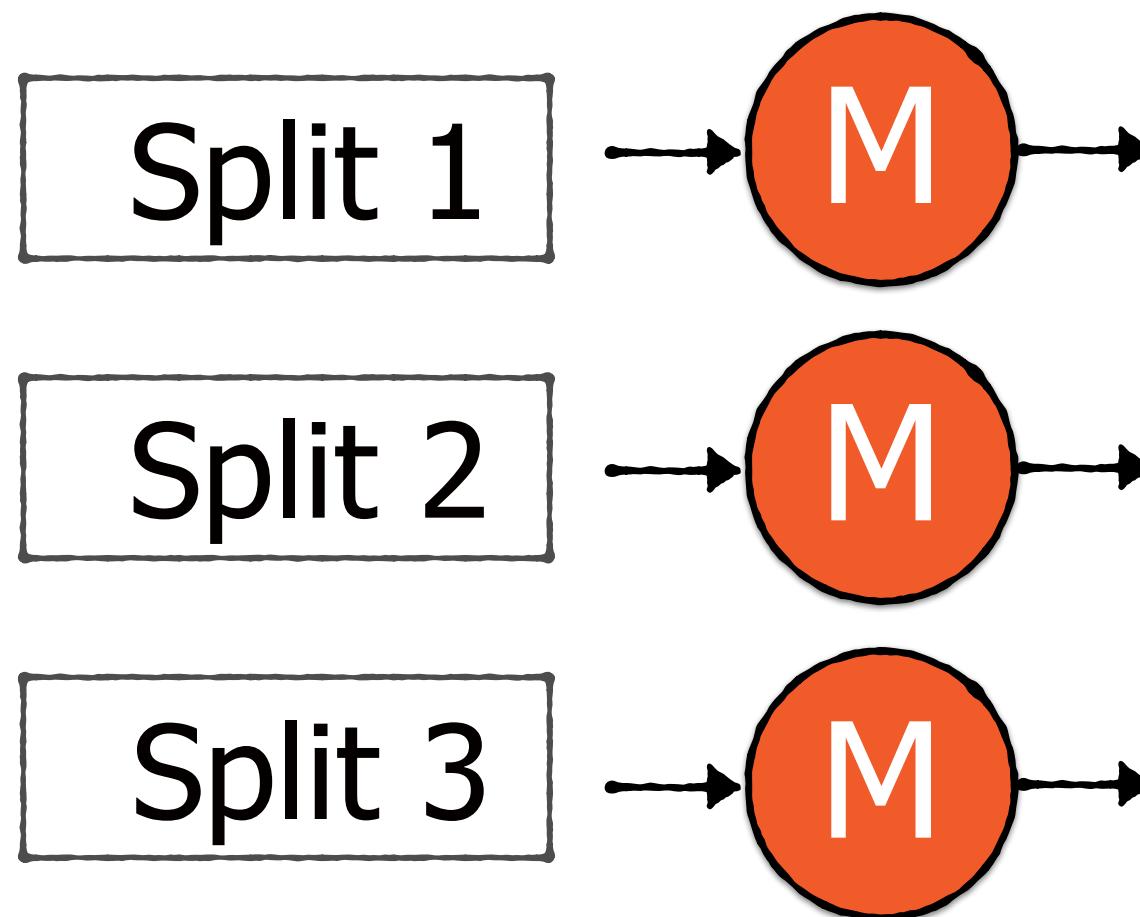
Number of reducers = 1 (by default)

Controlled by the user

Optimizes performance



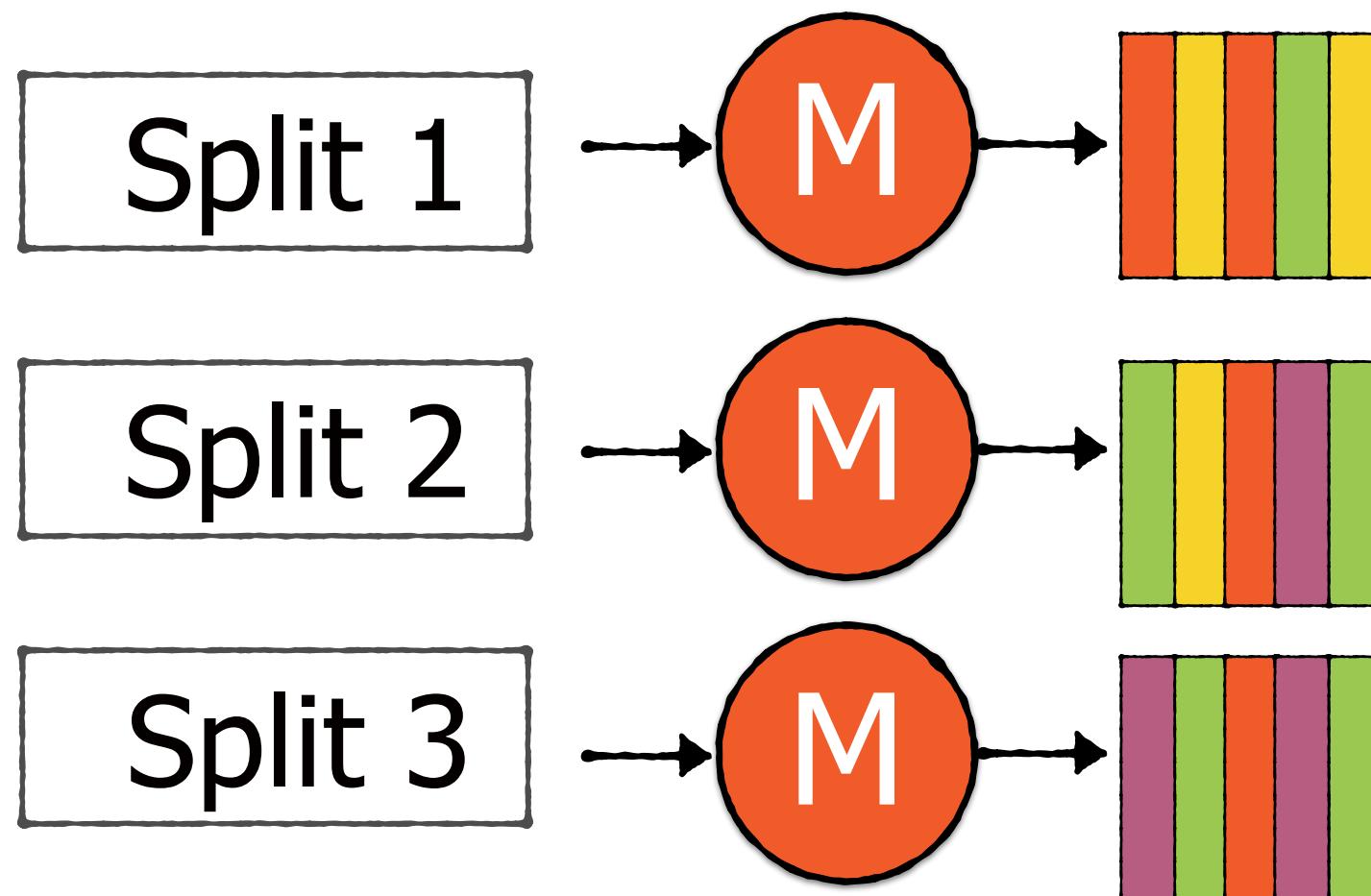
Behind the Scenes



Say our input dataset
had been split into 3
blocks by the underlying
storage system

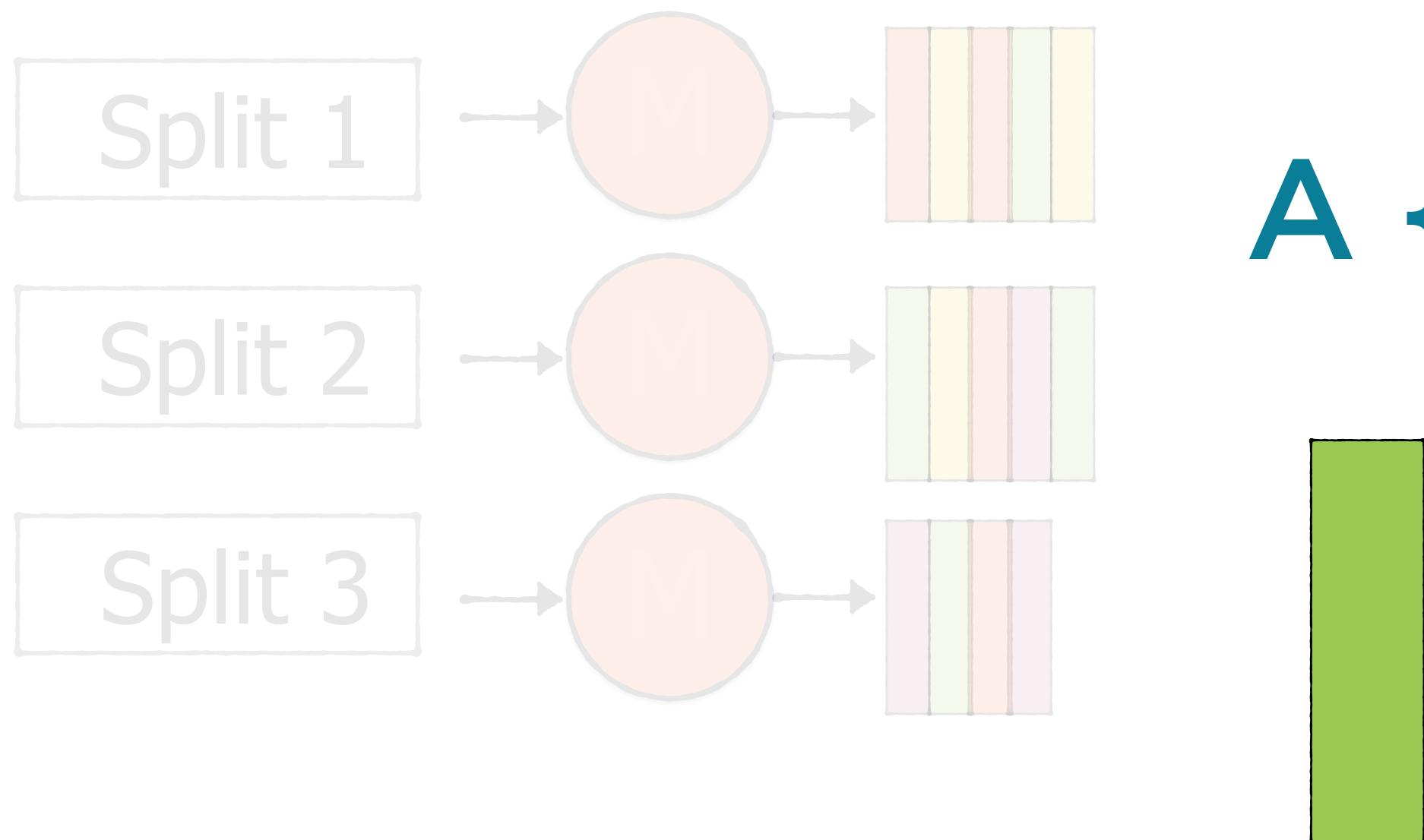
There will 3 mappers

Behind the Scenes



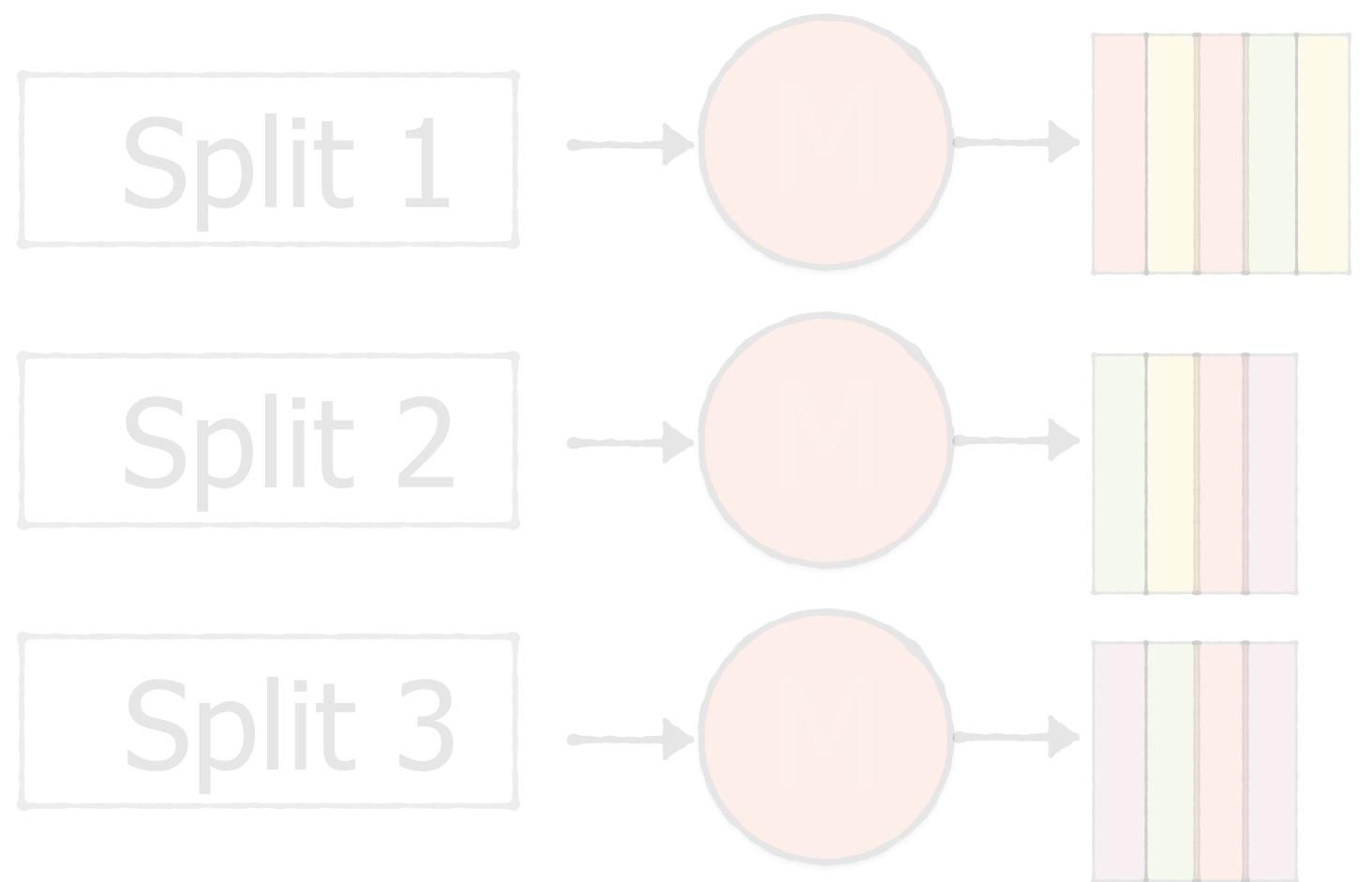
Mappers emit
{key, value} pairs

Behind the Scenes

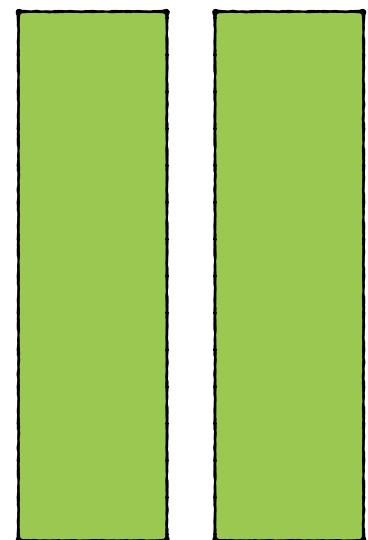


A **{key, value} pair**

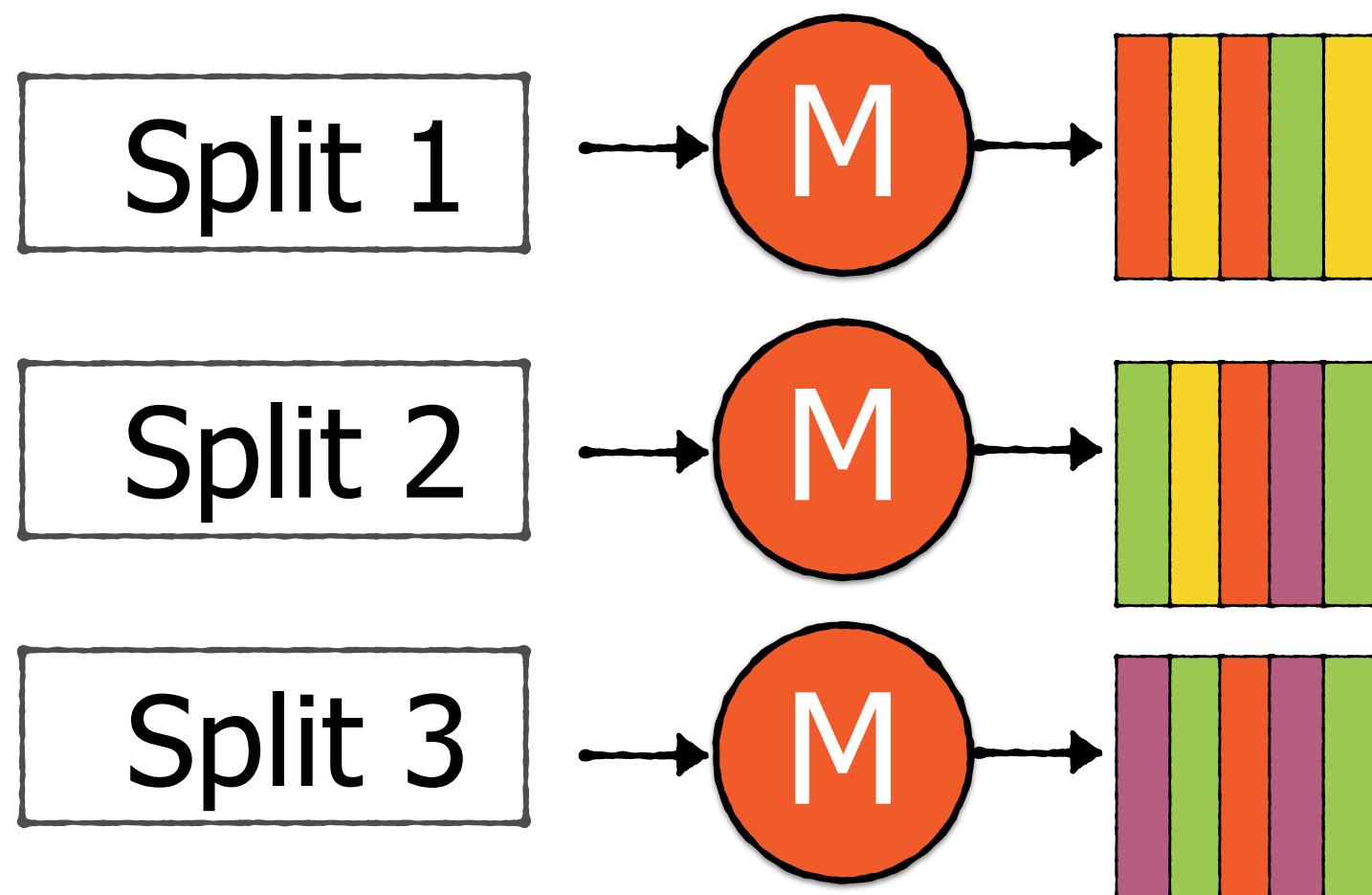
Behind the Scenes



**Pairs with the
same key**

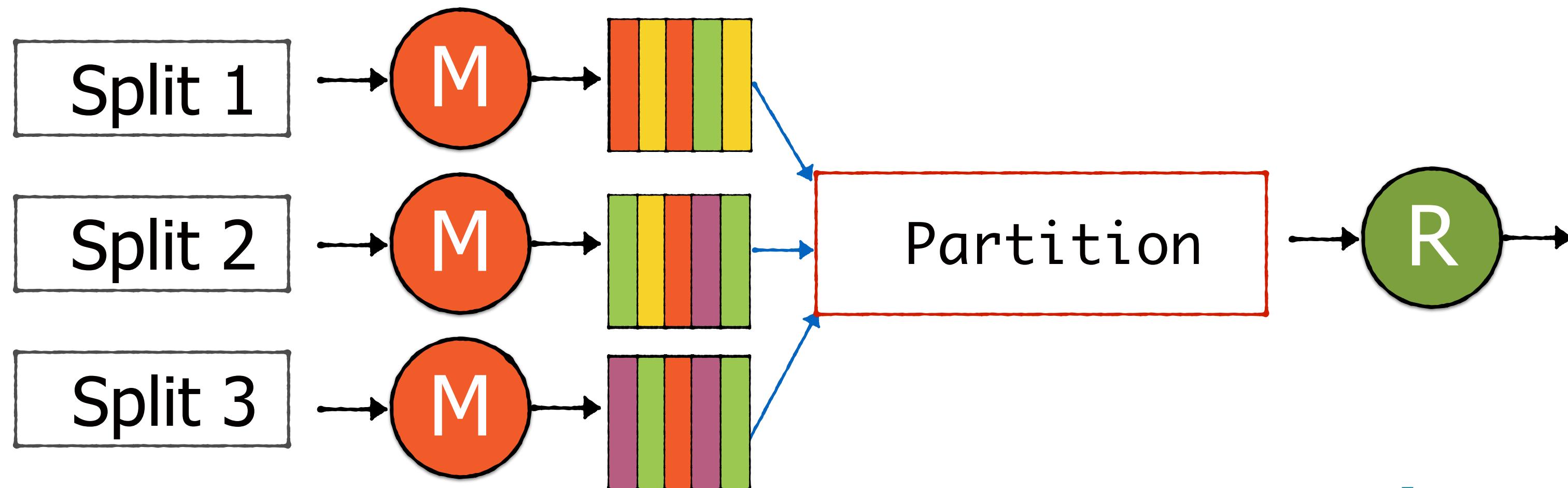


Behind the Scenes



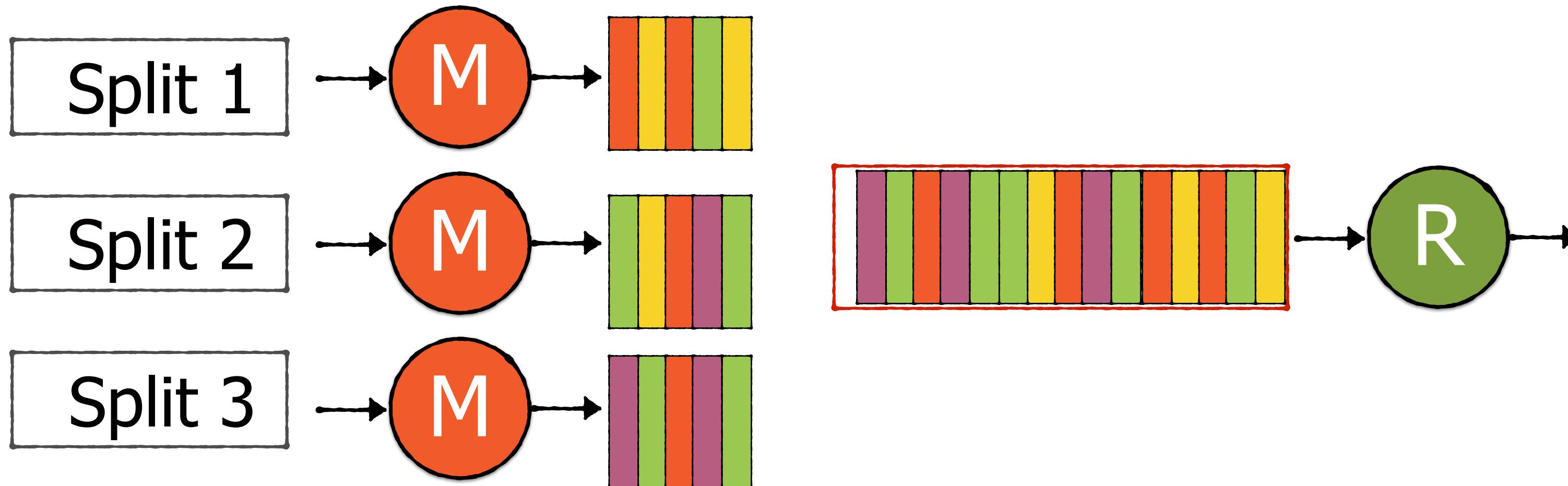
What happens next
depends on the
number of reducers

Behind the Scenes

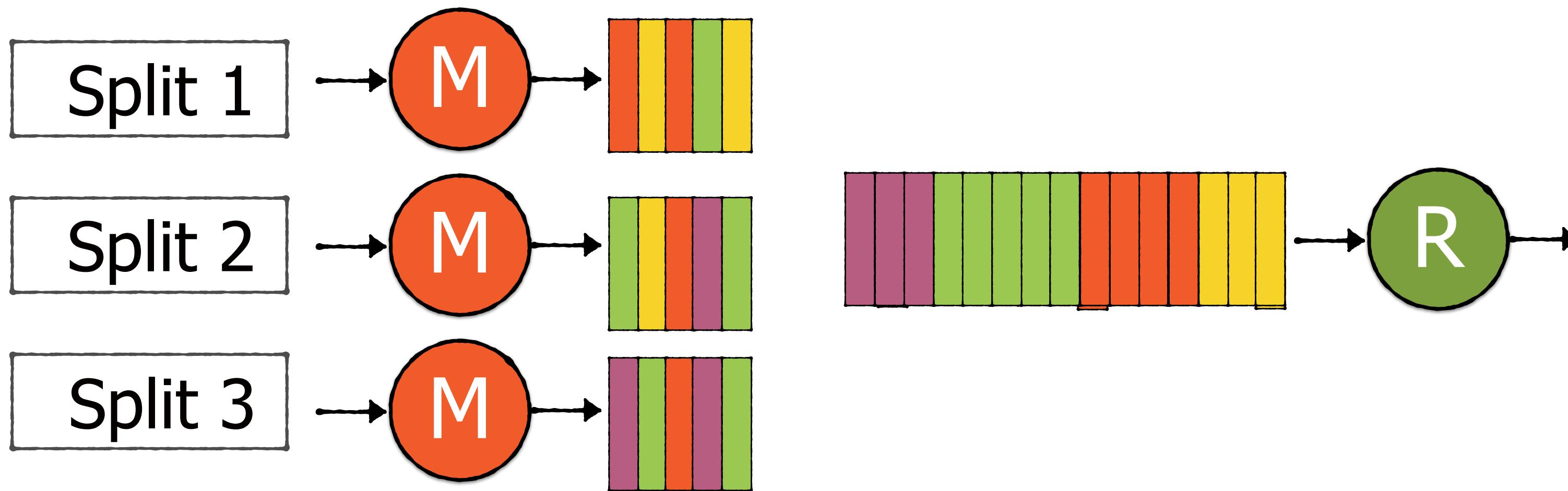


1 Reducer

Behind the Scenes

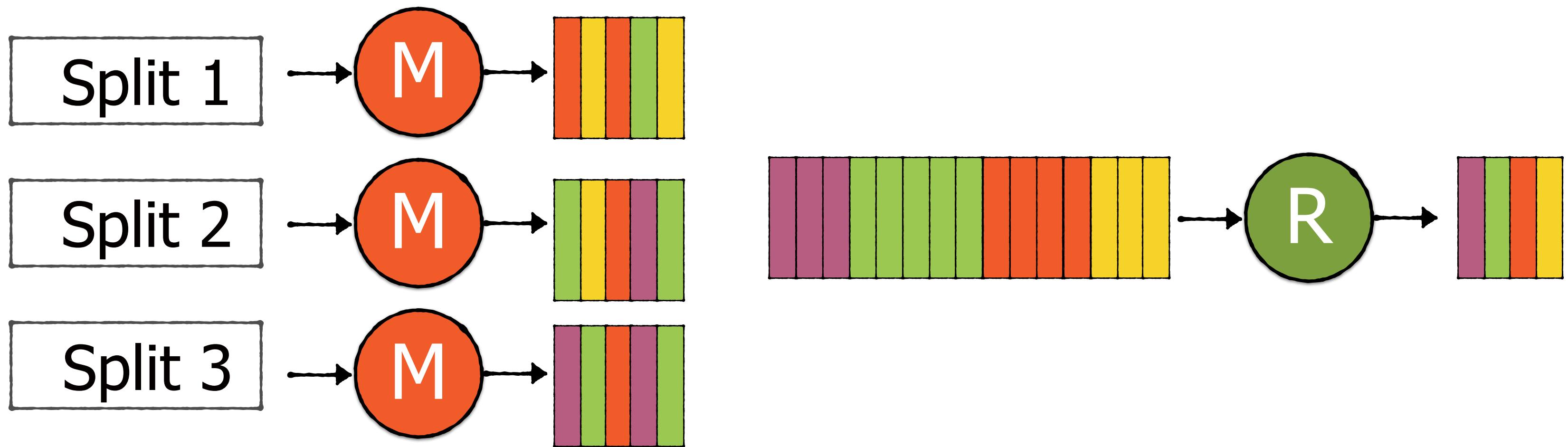


Behind the Scenes



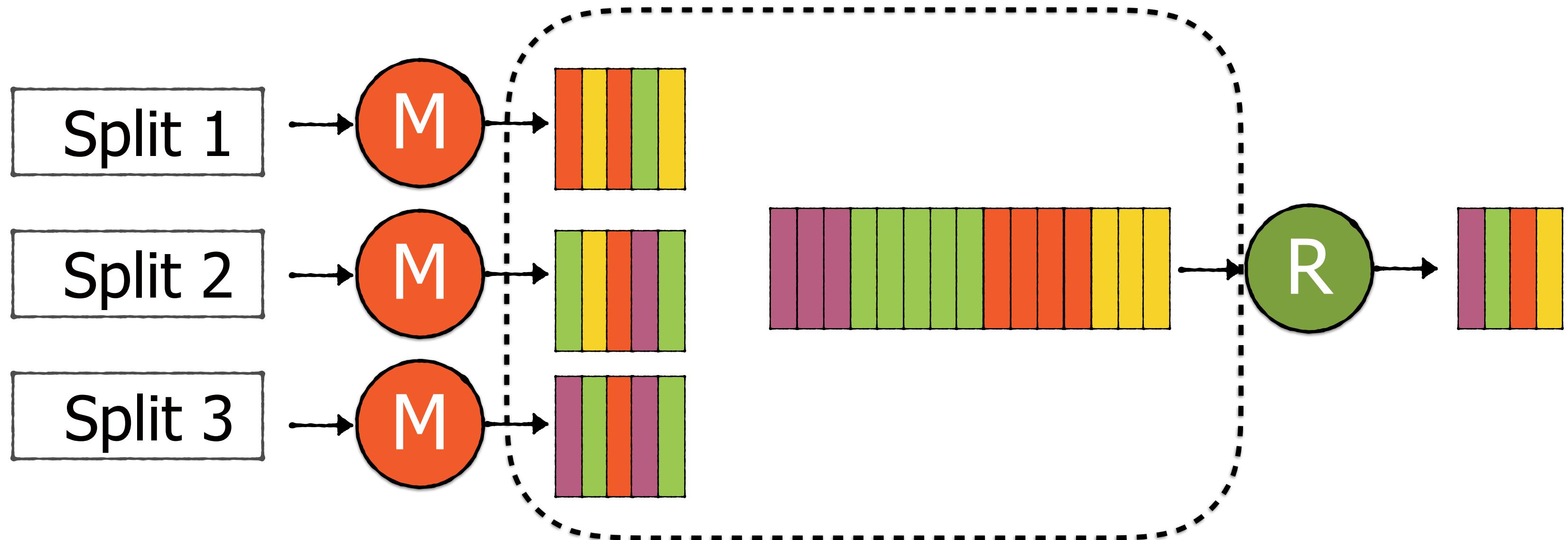
**Sort the pairs
based on the key**

Behind the Scenes



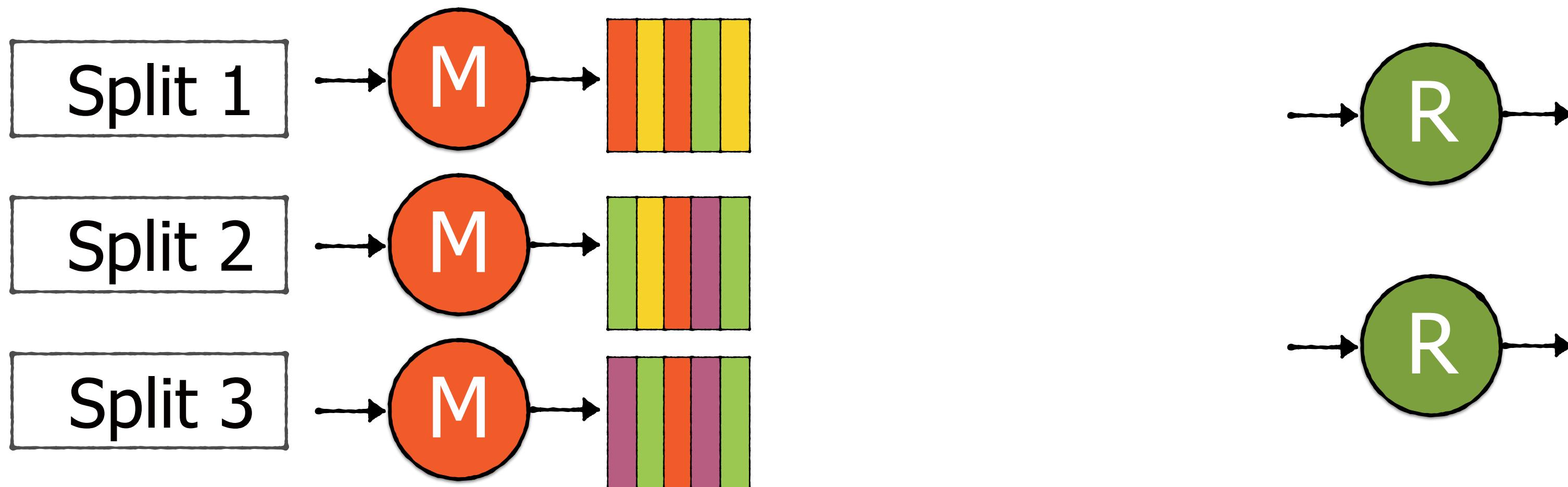
Combine values
with the same key

Behind the Scenes



Handled by the
framework completely
behind the scenes

Behind the Scenes

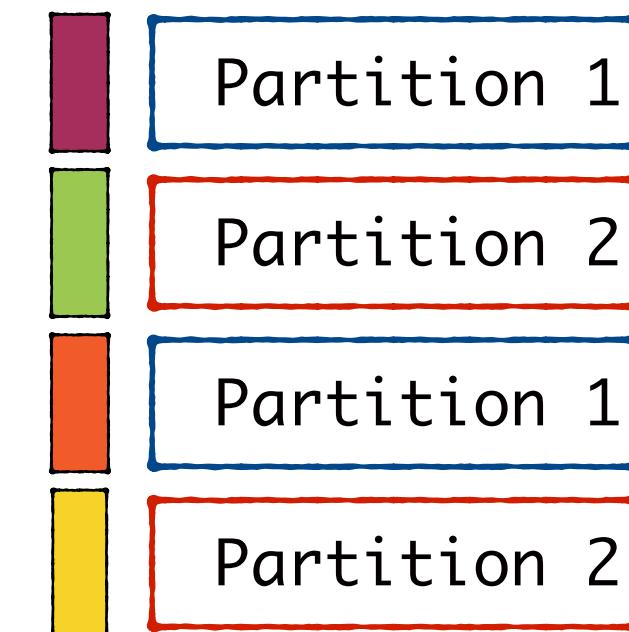


2 Reducers

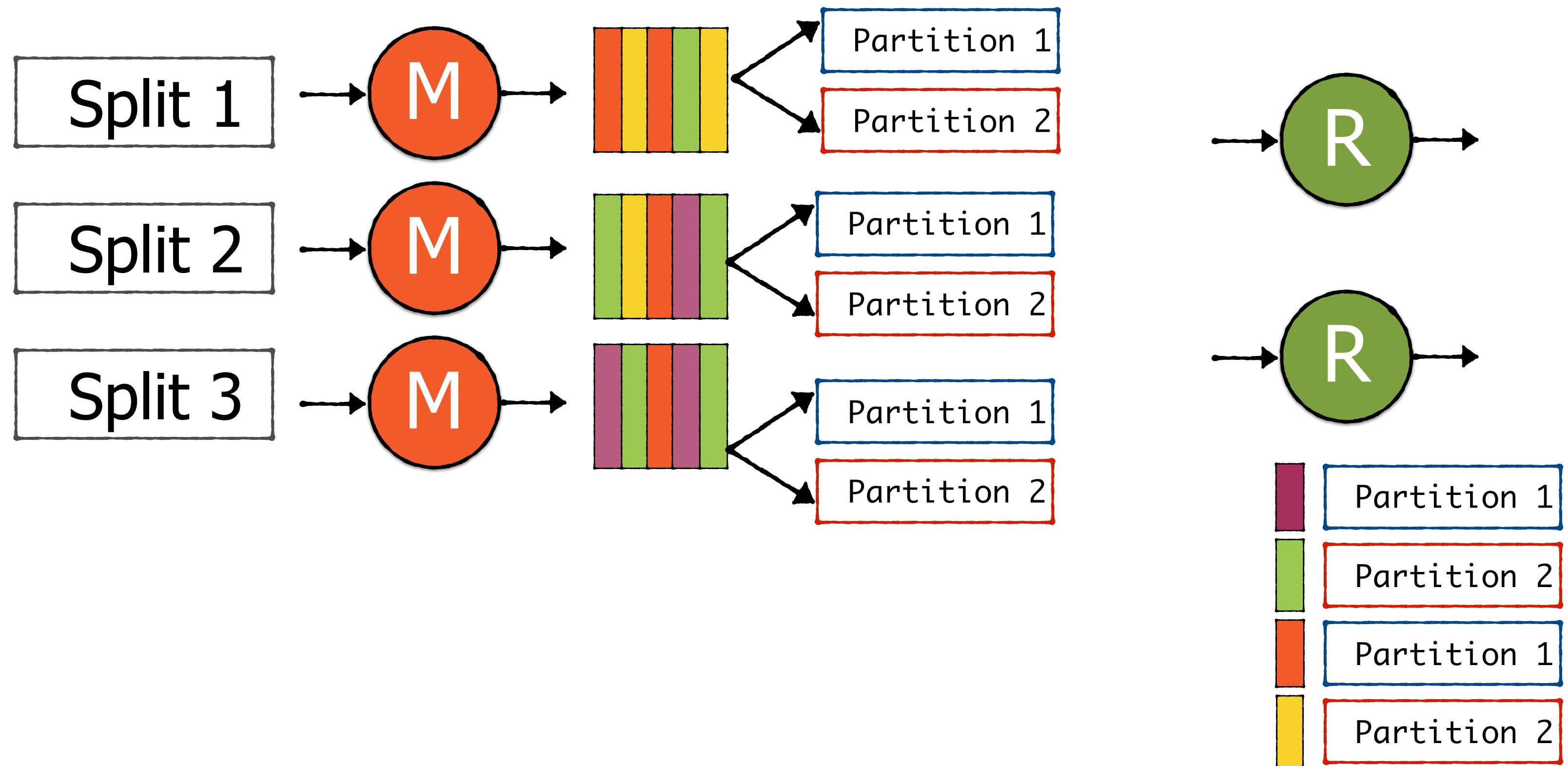
Assigning Partitions

After the map step, each key is assigned to a partition

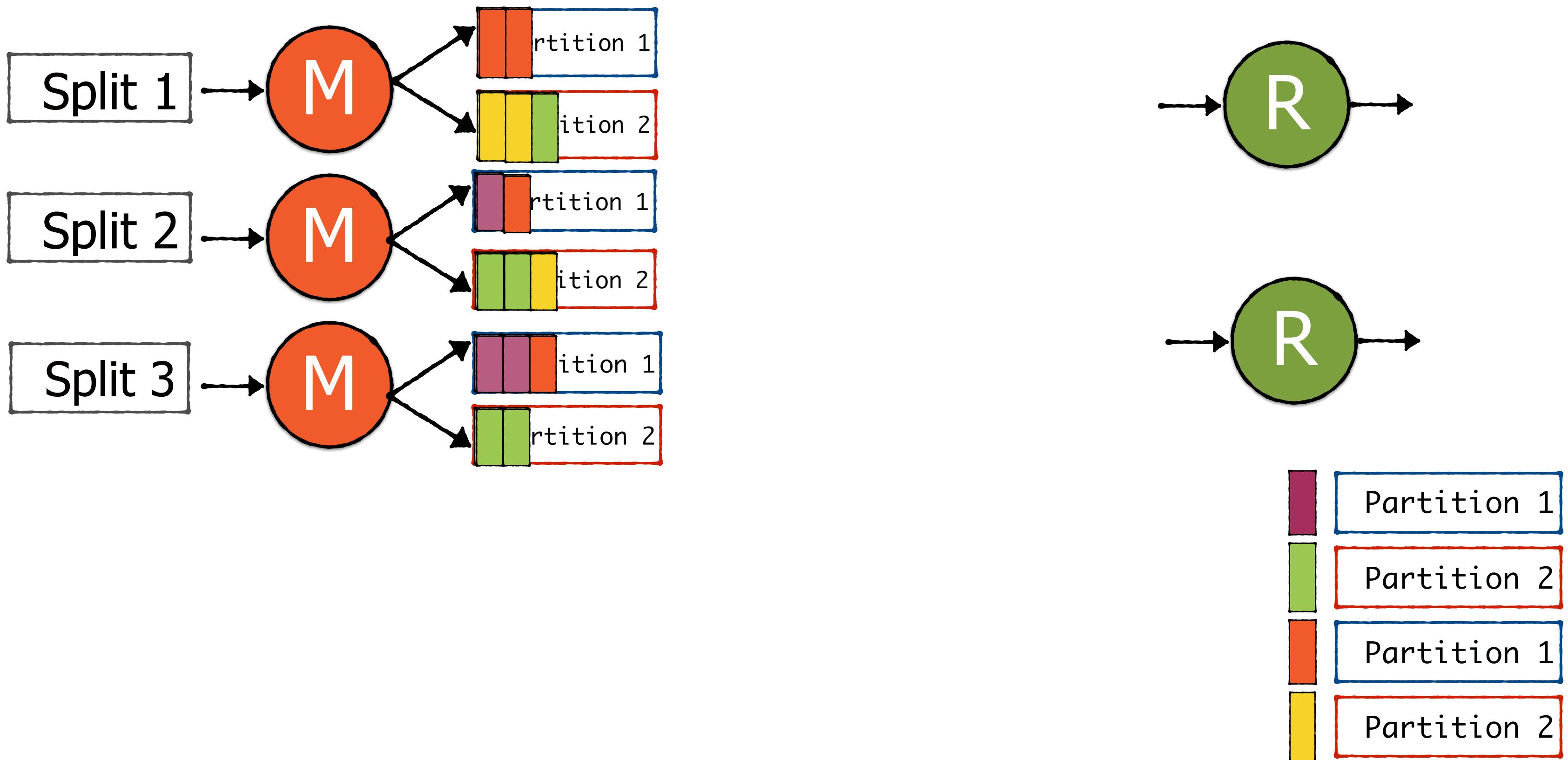
Partitions = # Reducers



Behind the Scenes



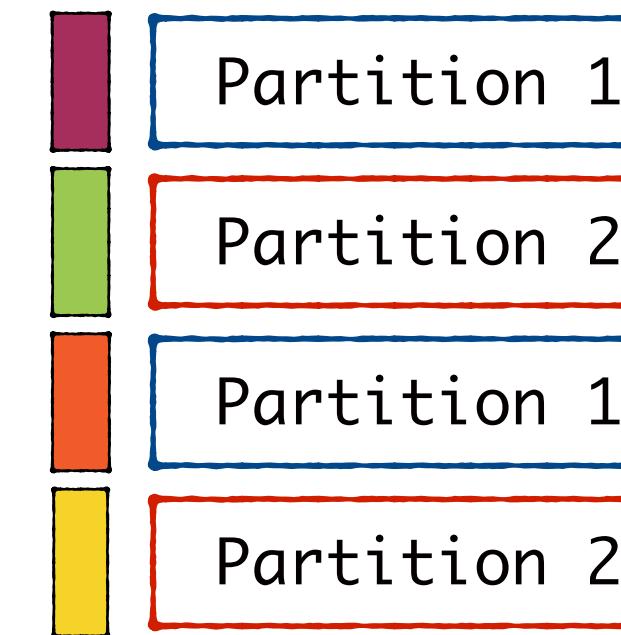
Behind the Scenes



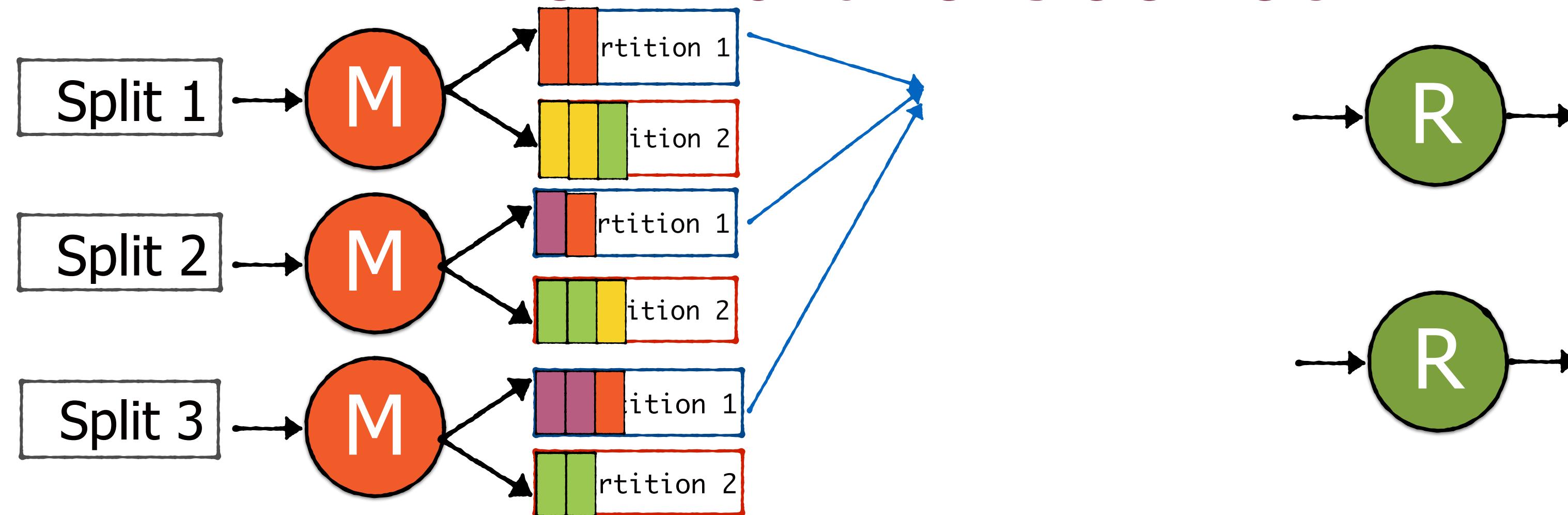
Assigning Partitions

Partitions = # Reducers

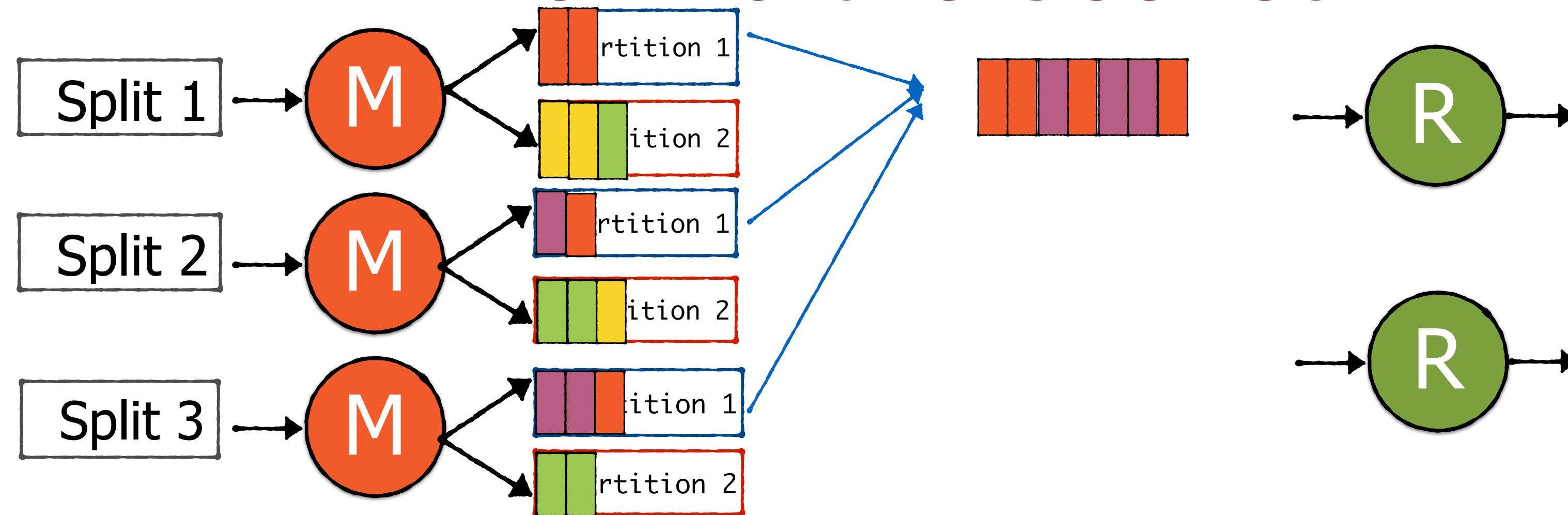
All the results in 1 partition
will go to 1 reducer



Behind the Scenes

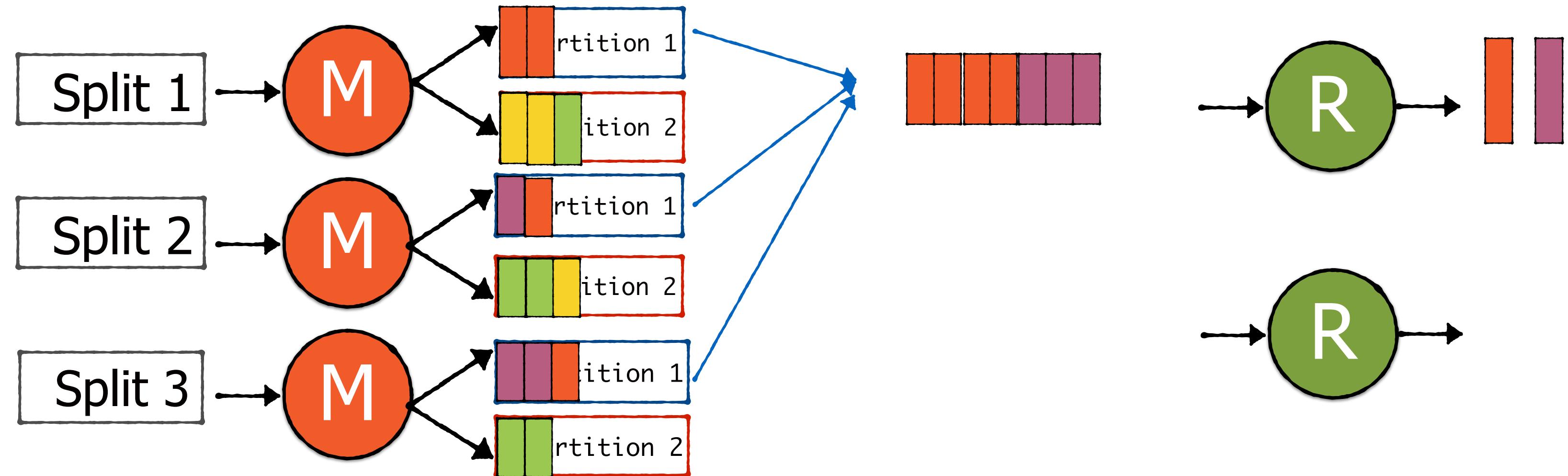


Behind the Scenes



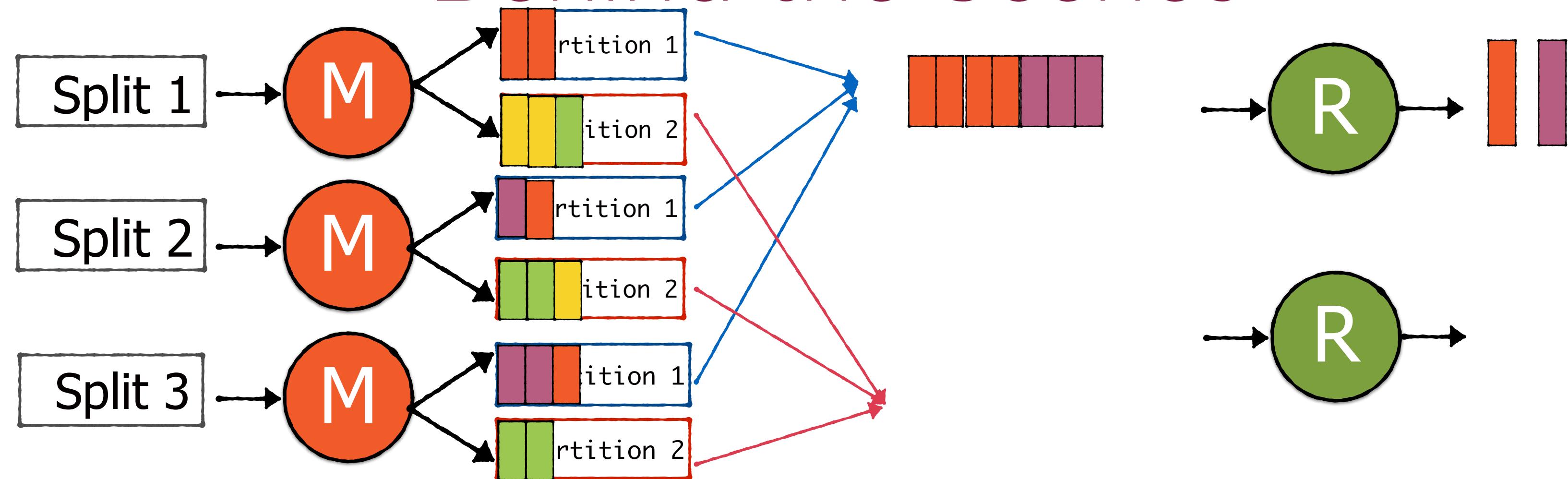
The pairs are also
sorted by key

Behind the Scenes



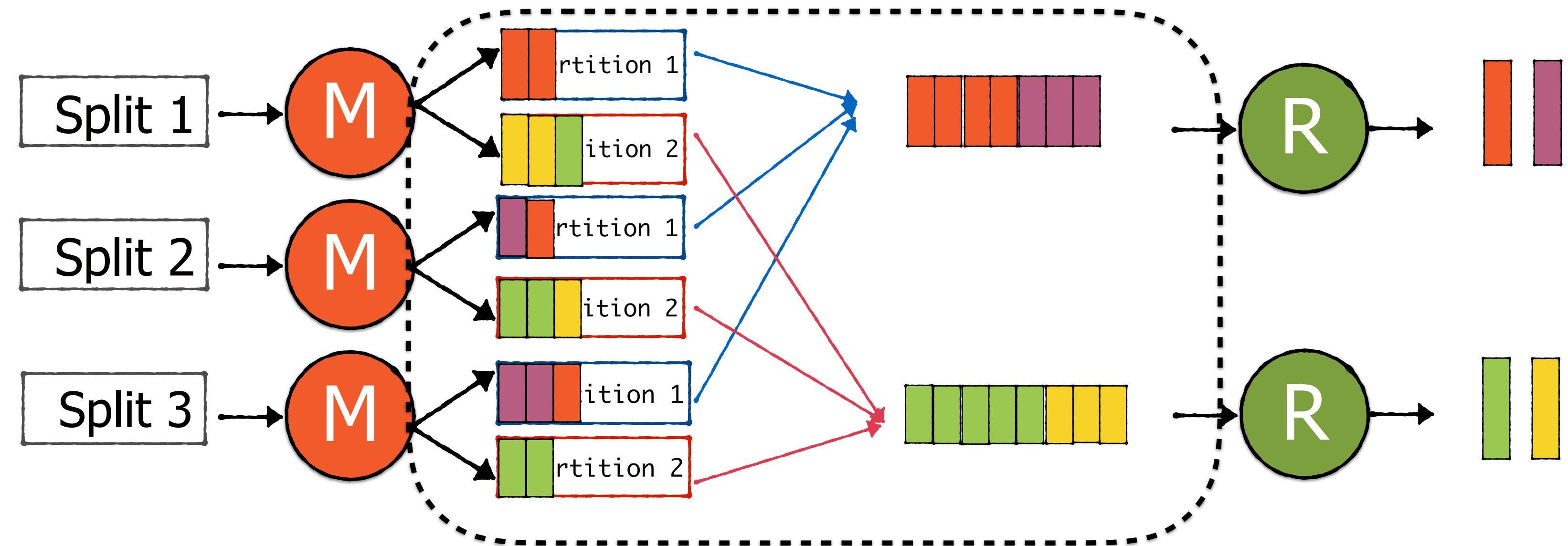
The reducer combines
values based on the
programmer supplied logic

Behind the Scenes



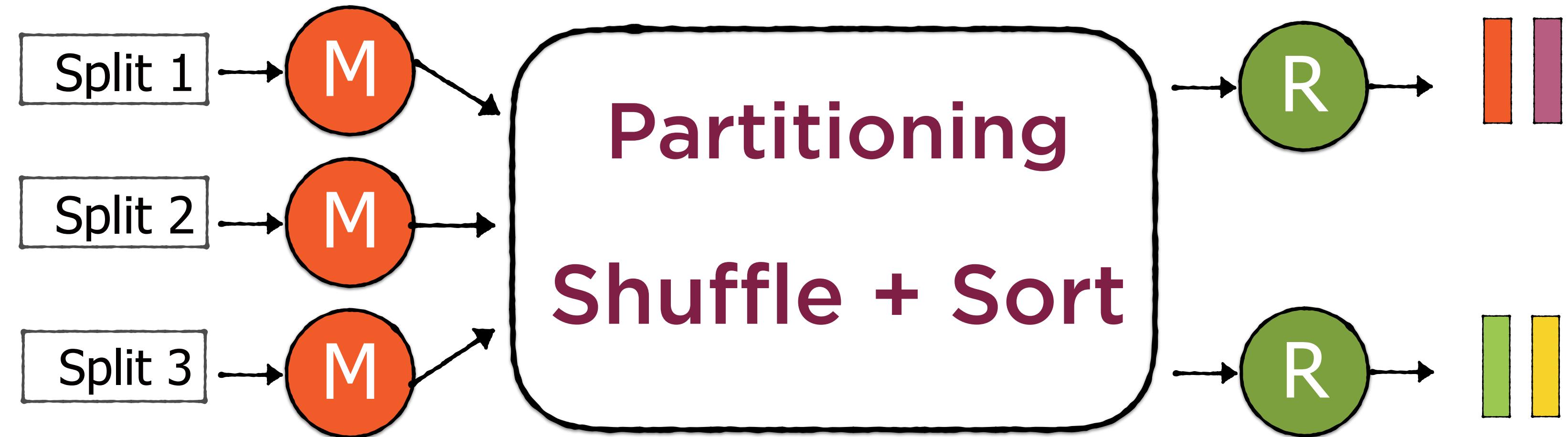
All of this occurs
in parallel for each
partition, reducer

Behind the Scenes



These 2 steps are managed
completely behind the scenes

Behind the Scenes





These framework provided features which allow us to control parallelism

Between the Map and Reduce Phases

Partition

Shuffle

Sort

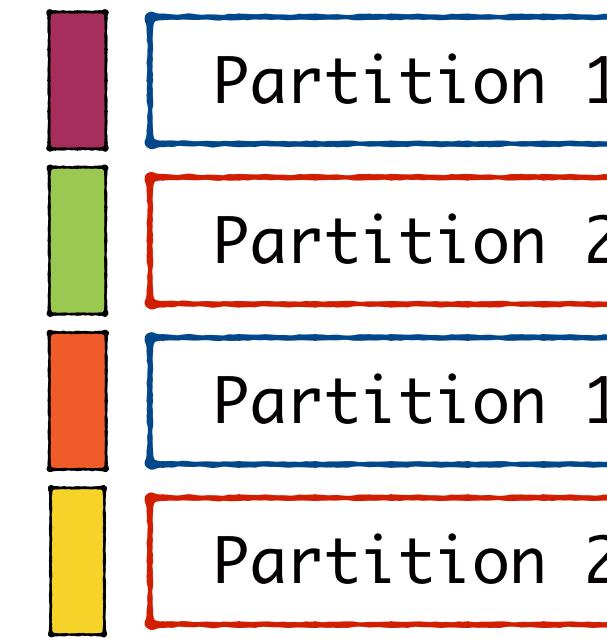
Three operations performed by the
Hadoop MapReduce framework

Partition

This refers to determining which map output key goes to which partition

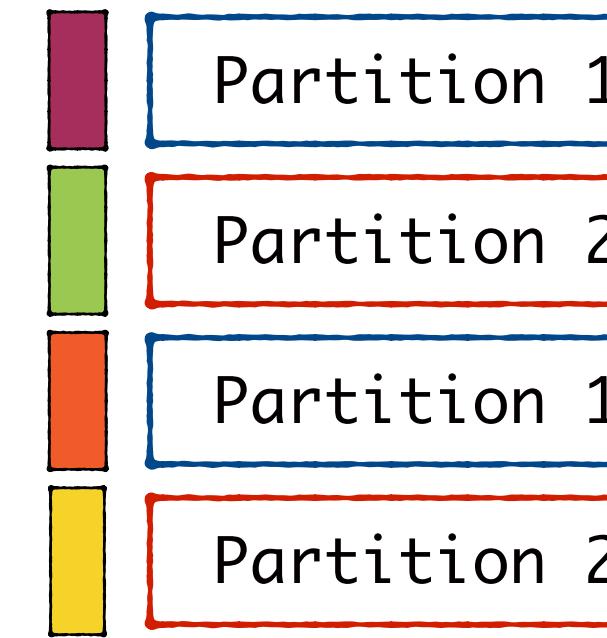
Used with multiple reducers

Partition



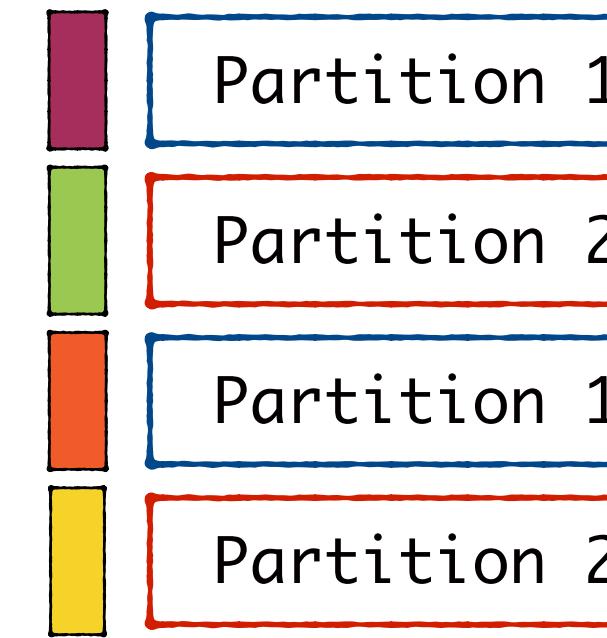
By default Hadoop uses a
hash function on the keys
to split them across
partitions

Partition



All the results in one partition go to one reducer

Partition

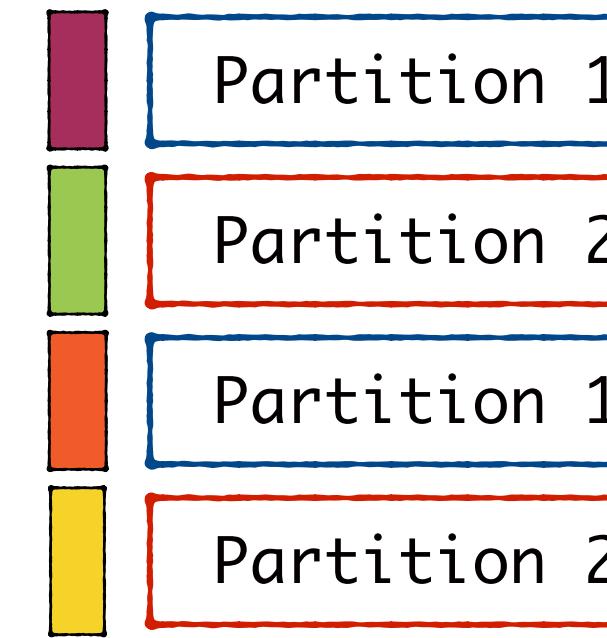


The hash function should
be consistent, the same
key is always assigned the
same partition



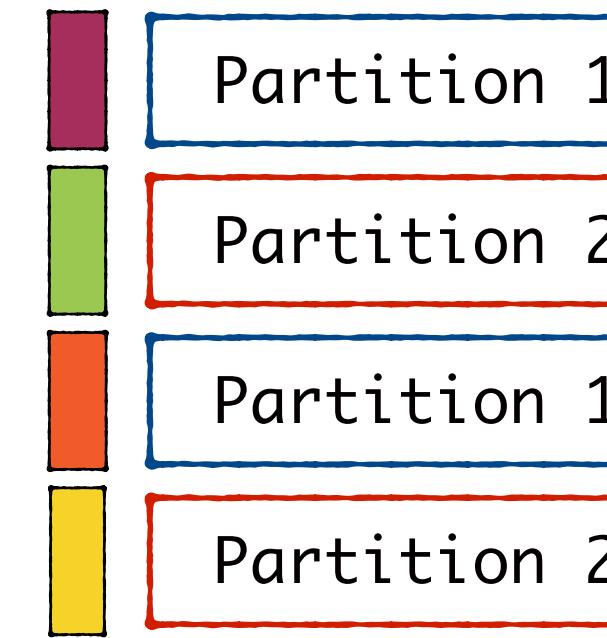
How should we choose
the number of
reducers?

Partition



Choose reducers such that
keys are evenly distributed
across partitions

Partition

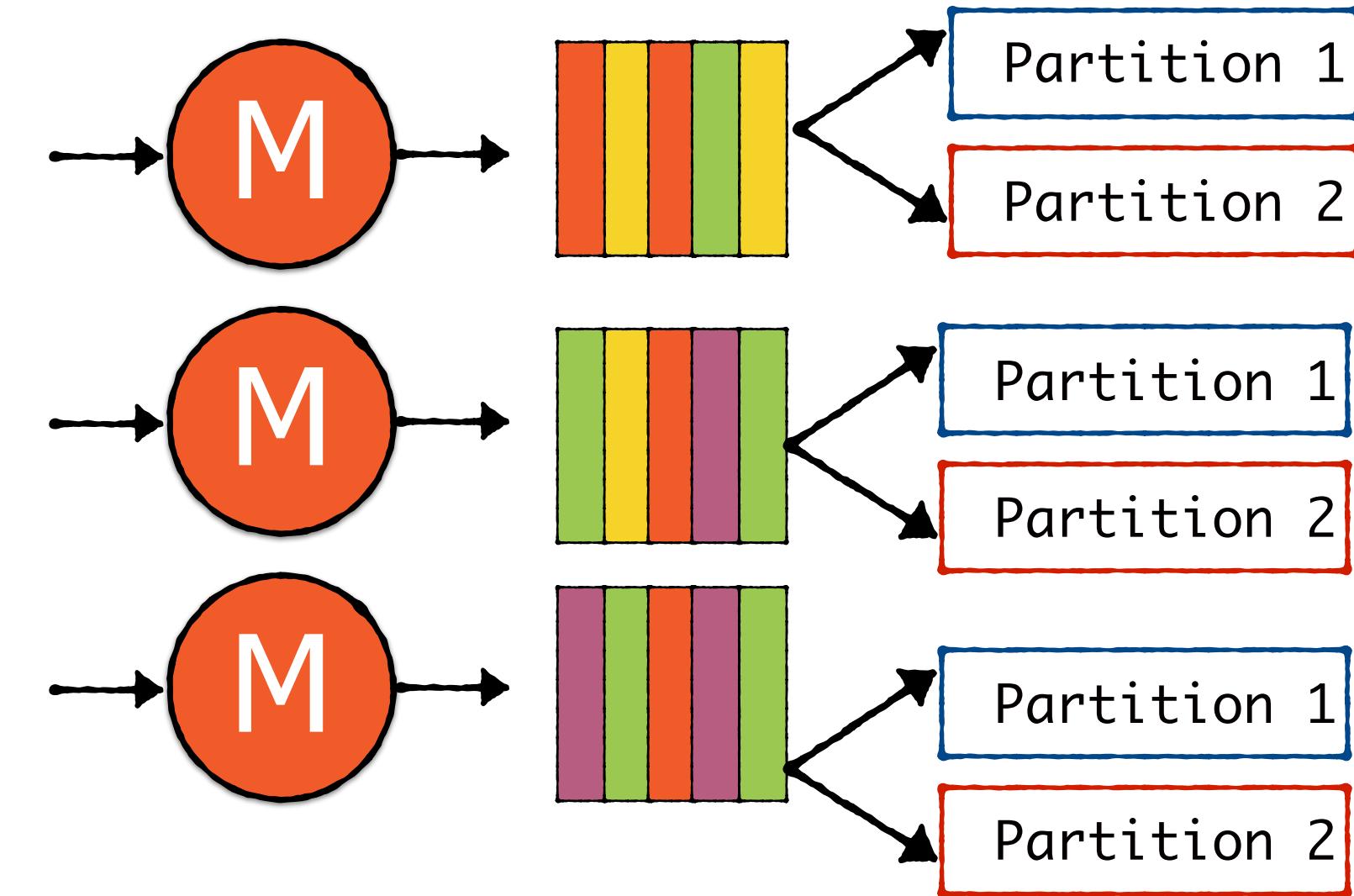


Hadoop allows you to
customize how you
partition the data as well

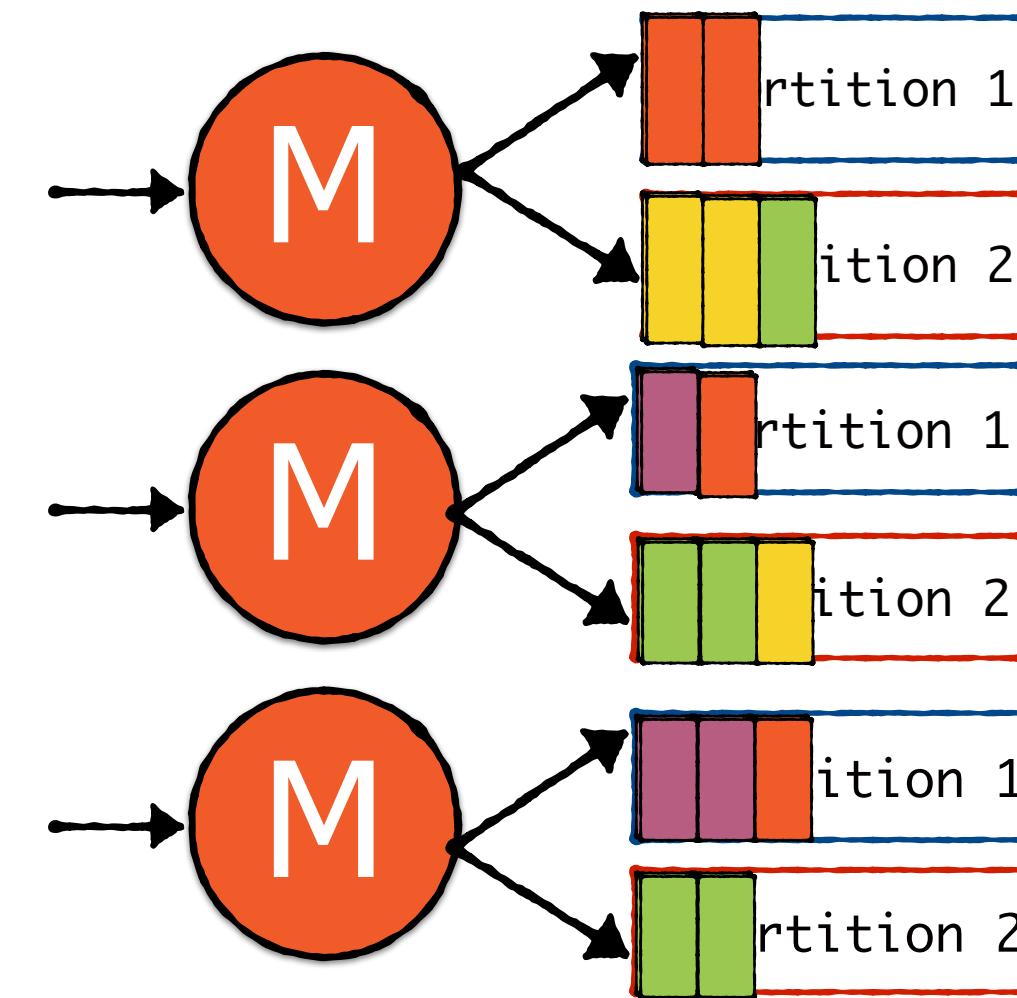
Shuffle

The process of moving
data from the mappers
to the reducers

Shuffle



Shuffle



Shuffle

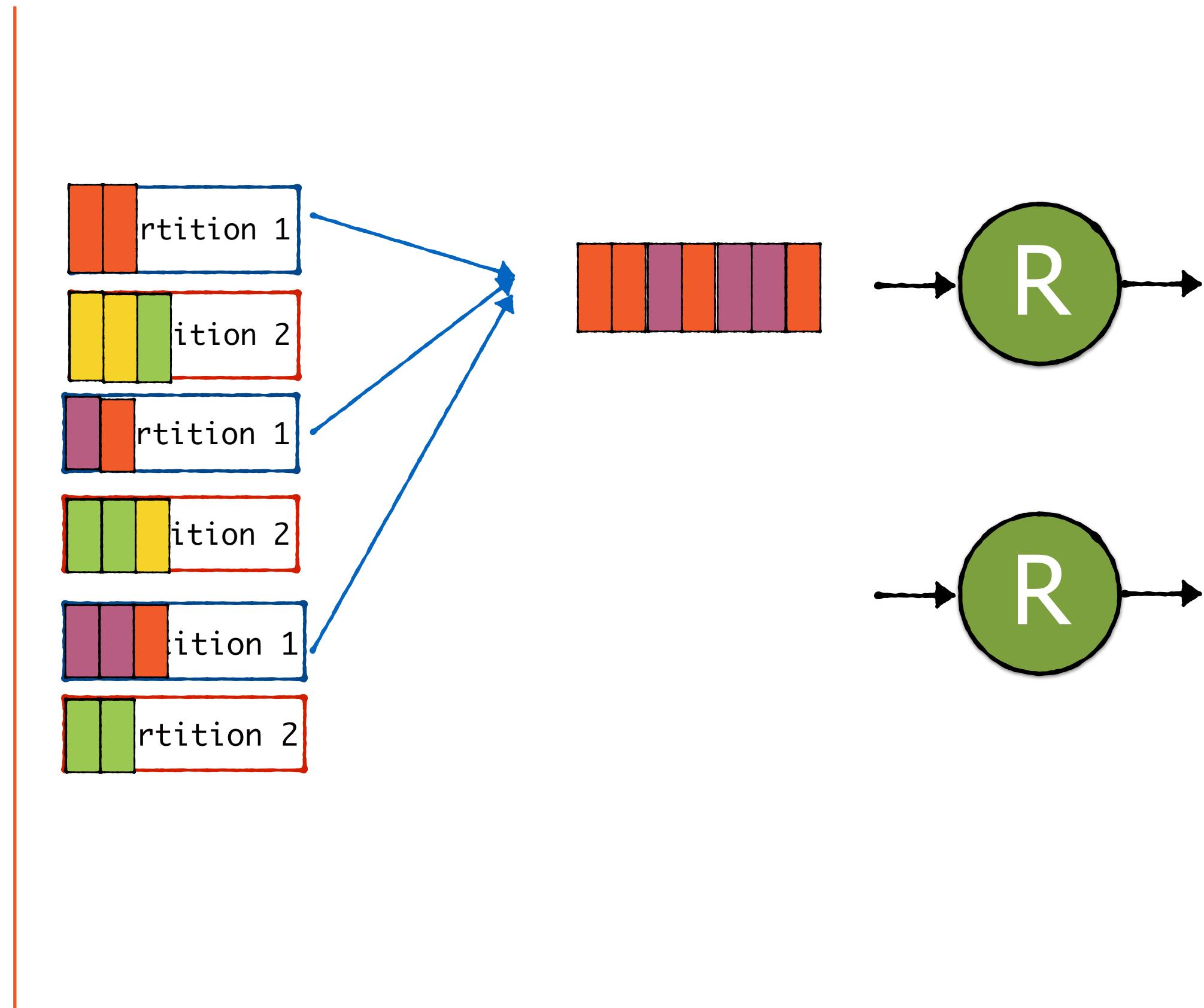
Required so reducers
get the data to
operate on

Sort

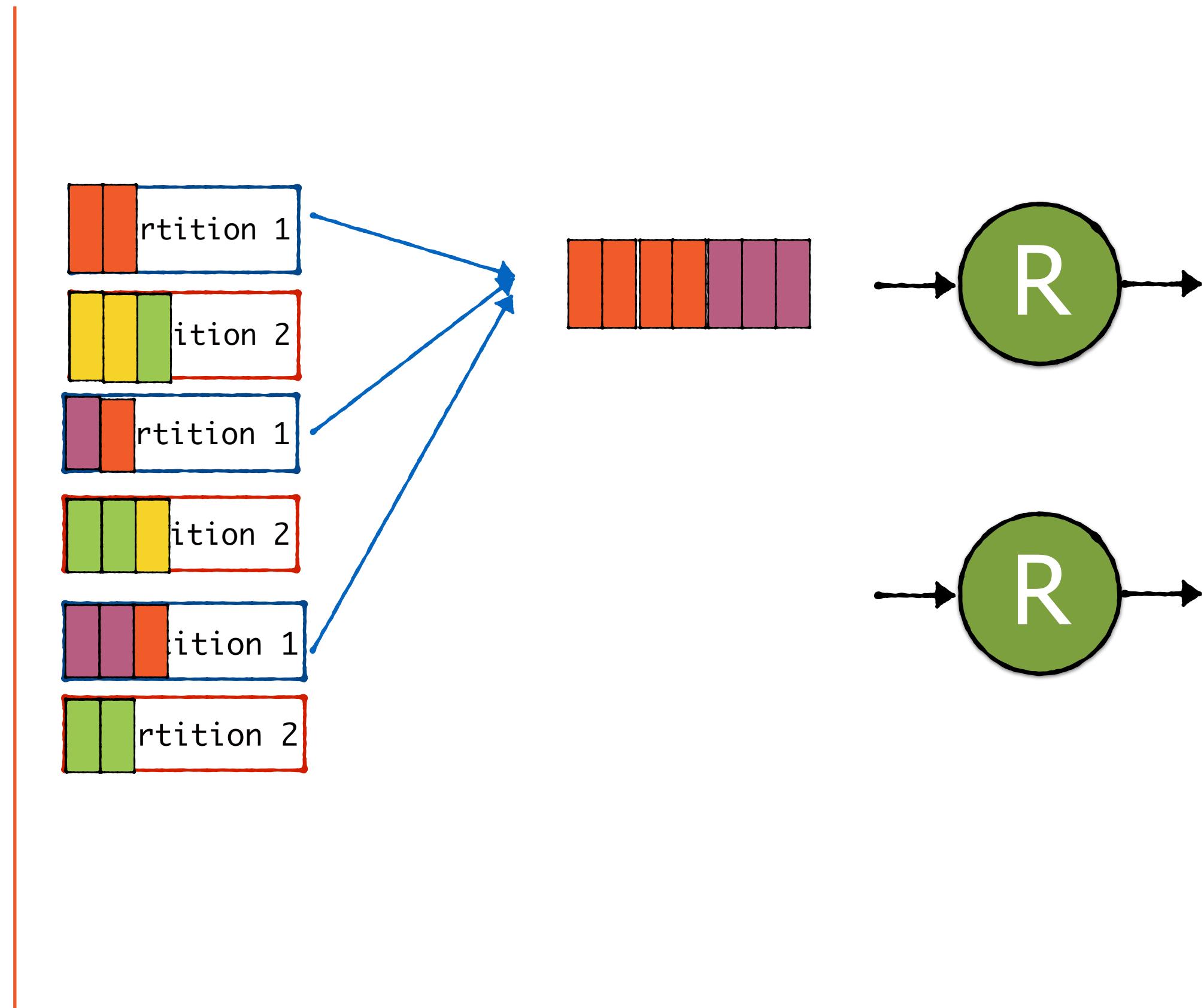
**Operation done on the
reducer to get all the same
keys together in a group**

**Values with the same
key appear as a
collection to the reducer**

Sort



Sort



Demo

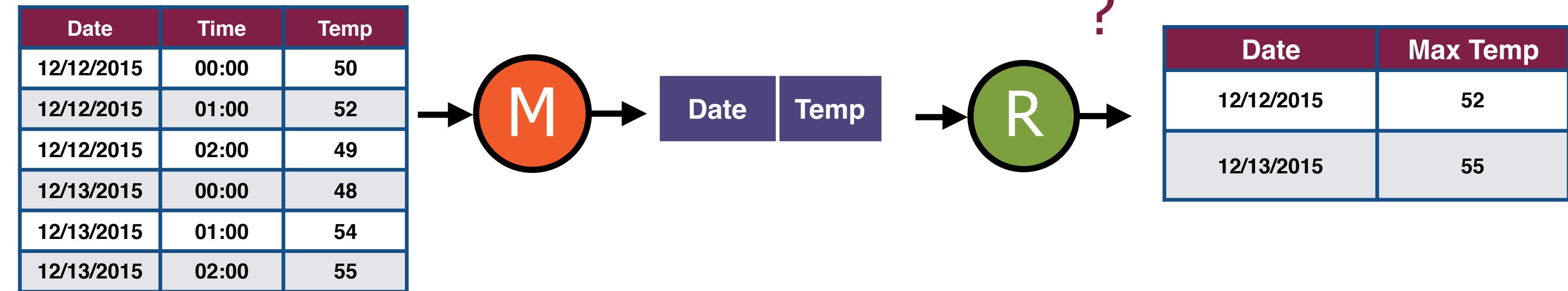
**Specify the number of Reducers to run
on our MapReduce job**

Max Temperature in a Day

Date	Time	Temp
12/12/2015	00:00	50
12/12/2015	01:00	52
12/12/2015	02:00	49
12/13/2015	00:00	48
12/13/2015	01:00	54
12/13/2015	02:00	55

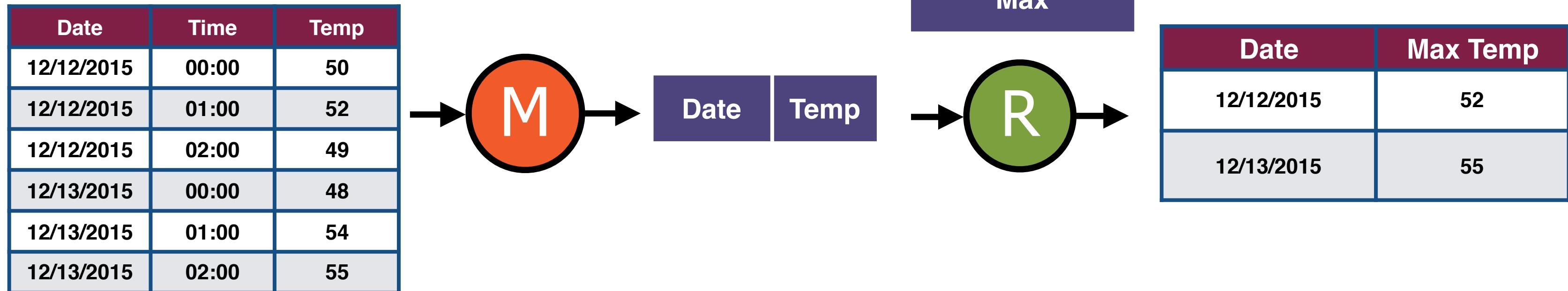
What is the maximum recorded temperature on a given day?

Max Temperature in a Day



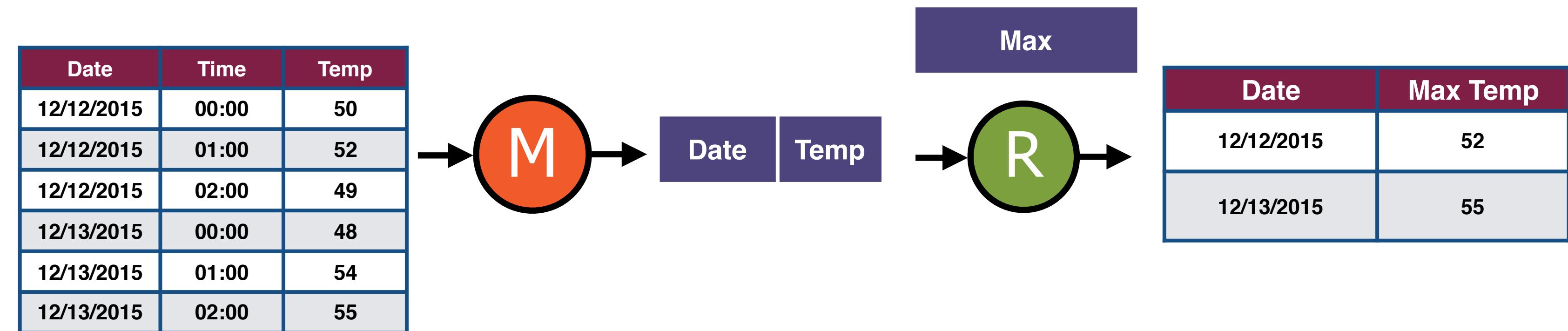
From each record
extract the date and
the temperature

Max Temperature in a Day



The reduce phase
finds the maximum
temperature for a
particular date

Max Temperature in a Day



The original data is likely to be split across multiple machines - the input splits

Max Temperature in a Day

Date	Time	Temp
12/12/2015	00:00	50
12/12/2015	01:00	52
12/12/2015	02:00	49
12/13/2015	00:00	48
12/13/2015	01:00	54
12/13/2015	02:00	55

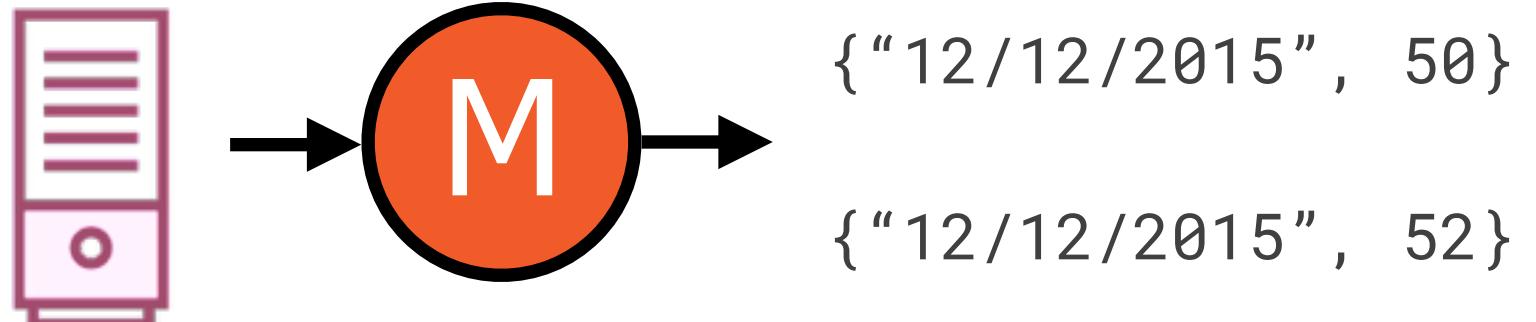
Date	Time	Temp
12/12/2015	00:00	50
12/12/2015	01:00	52

Date	Time	Temp
12/12/2015	02:00	49
12/13/2015	00:00	48

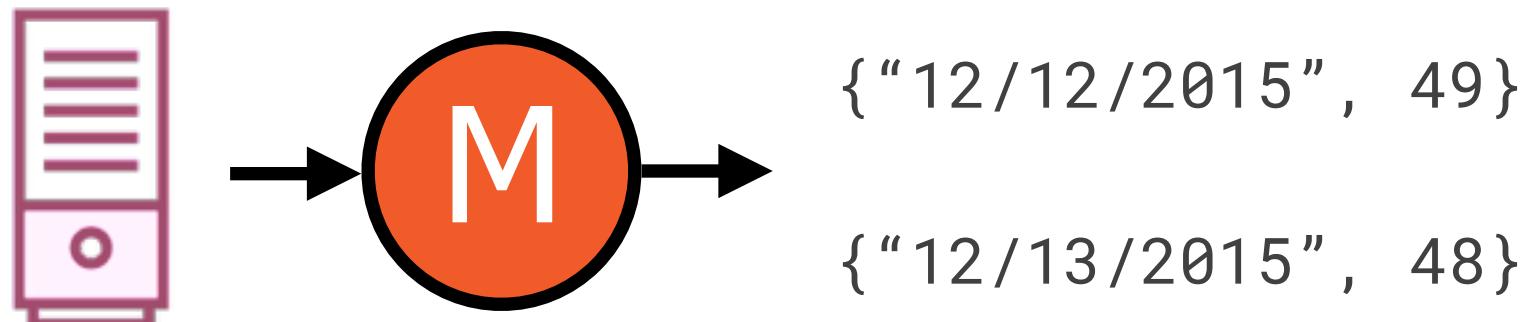
Date	Time	Temp
12/13/2015	01:00	54
12/13/2015	02:00	55

Max Temperature in a Day

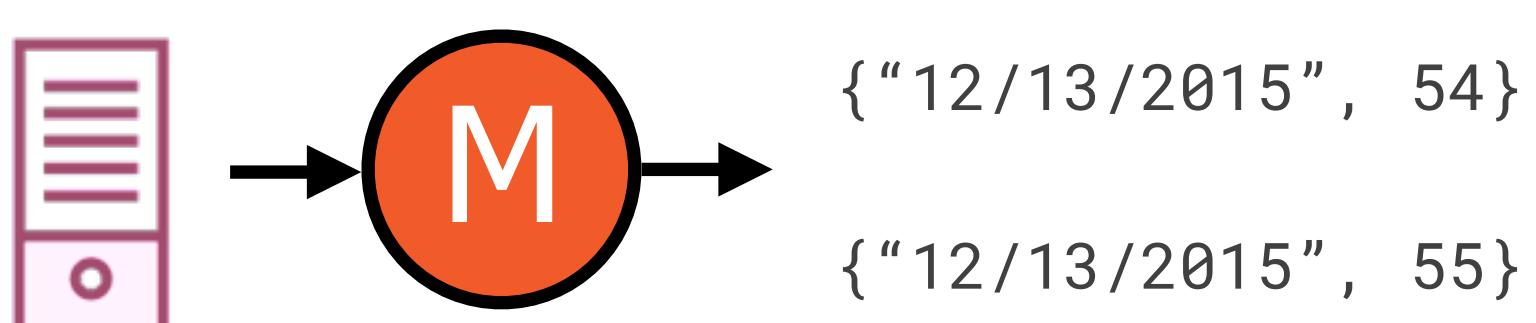
Date	Time	Temp
12/12/2015	00:00	50
12/12/2015	01:00	52



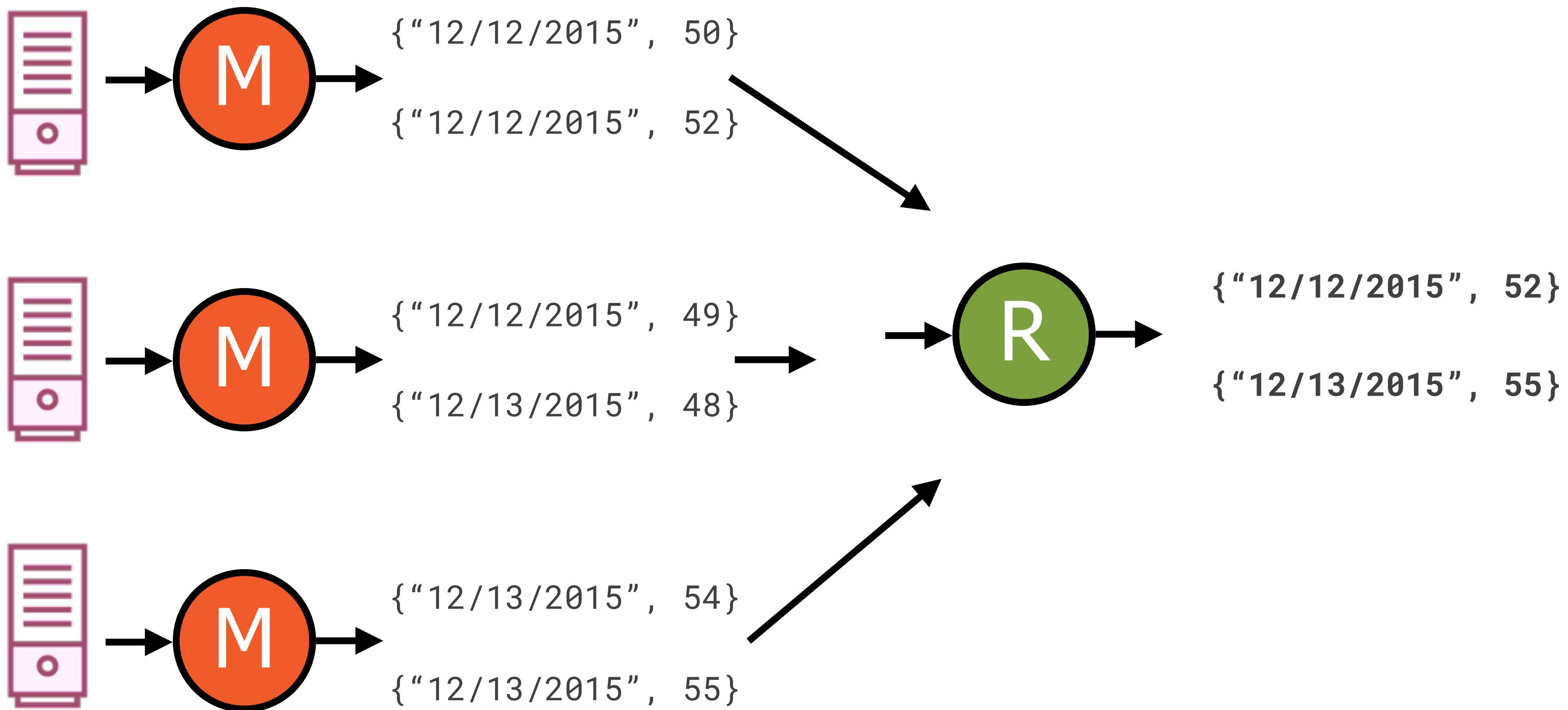
Date	Time	Temp
12/12/2015	02:00	49
12/13/2015	00:00	48



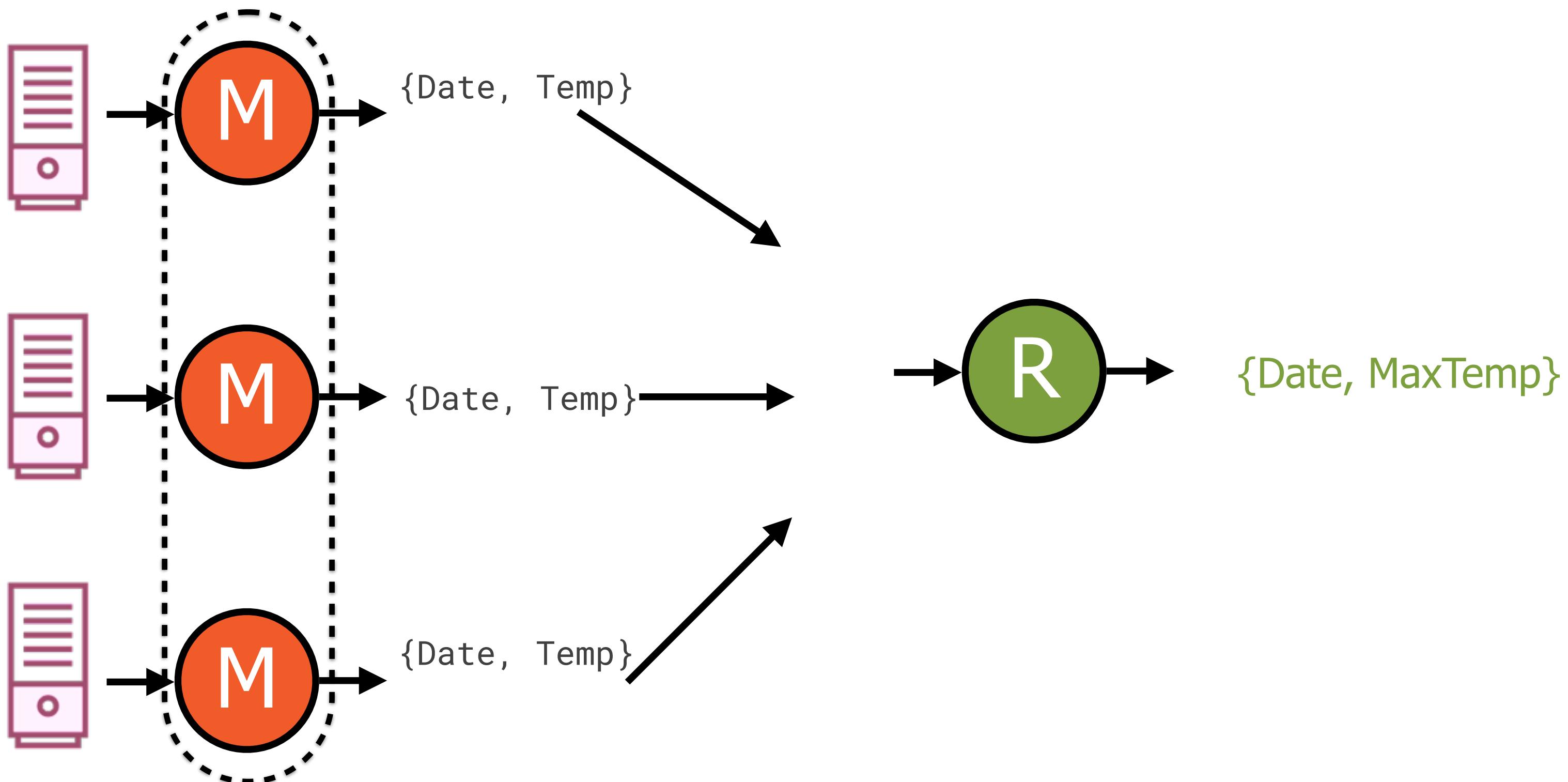
Date	Time	Temp
12/13/2015	01:00	54
12/13/2015	02:00	55



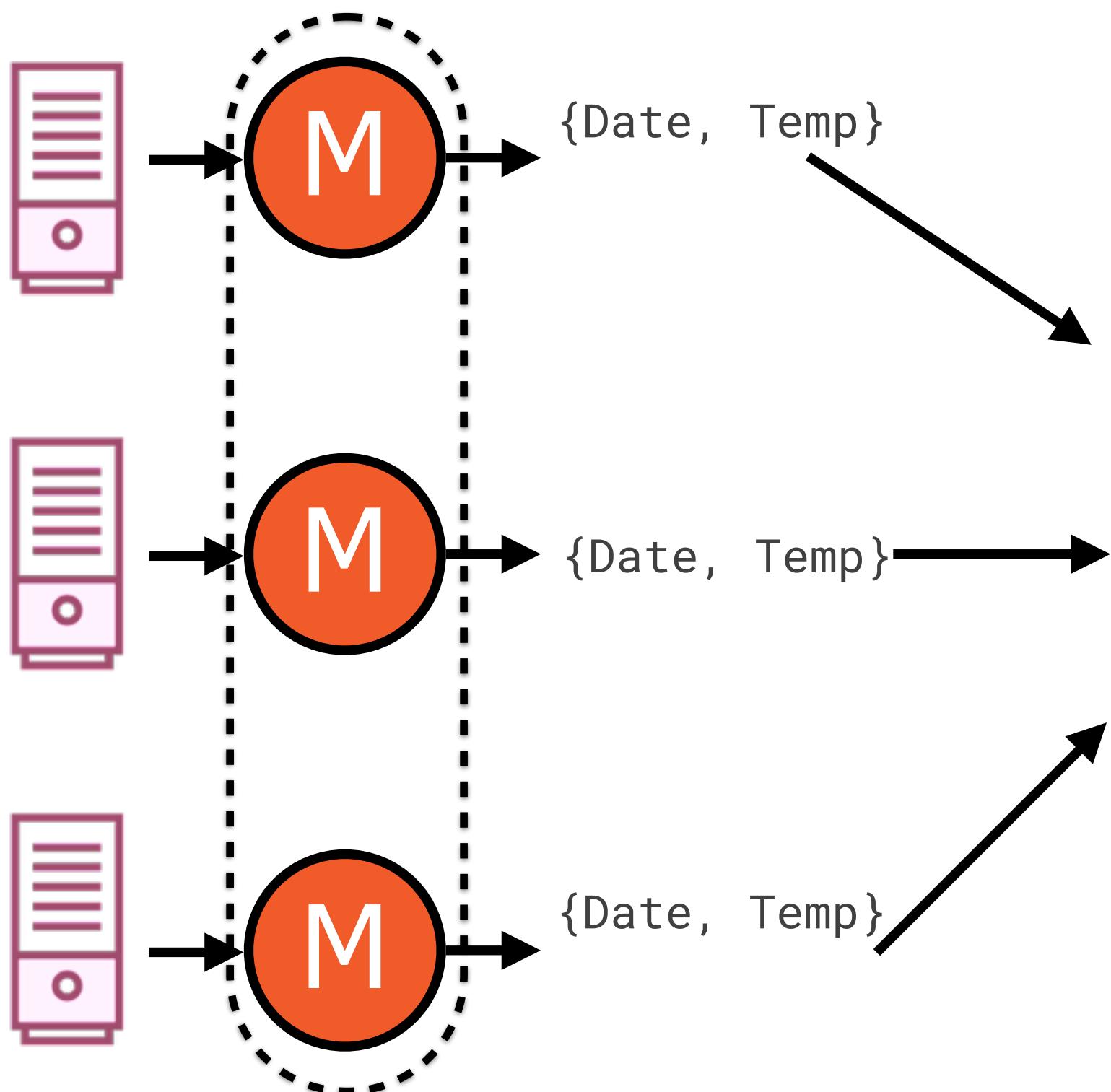
Max Temperature in a Day



Typical MapReduce Flow

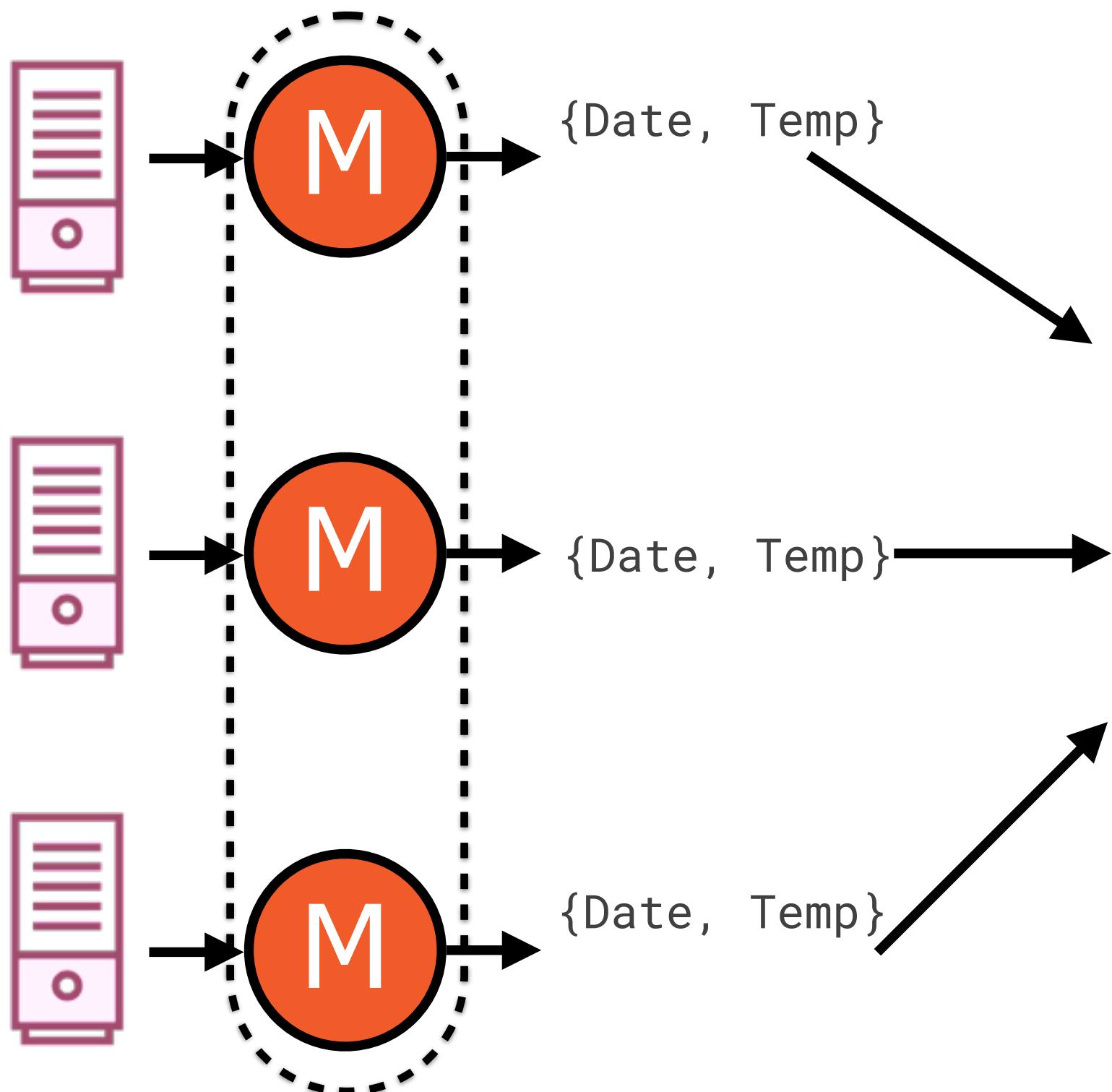


Typical MapReduce Flow



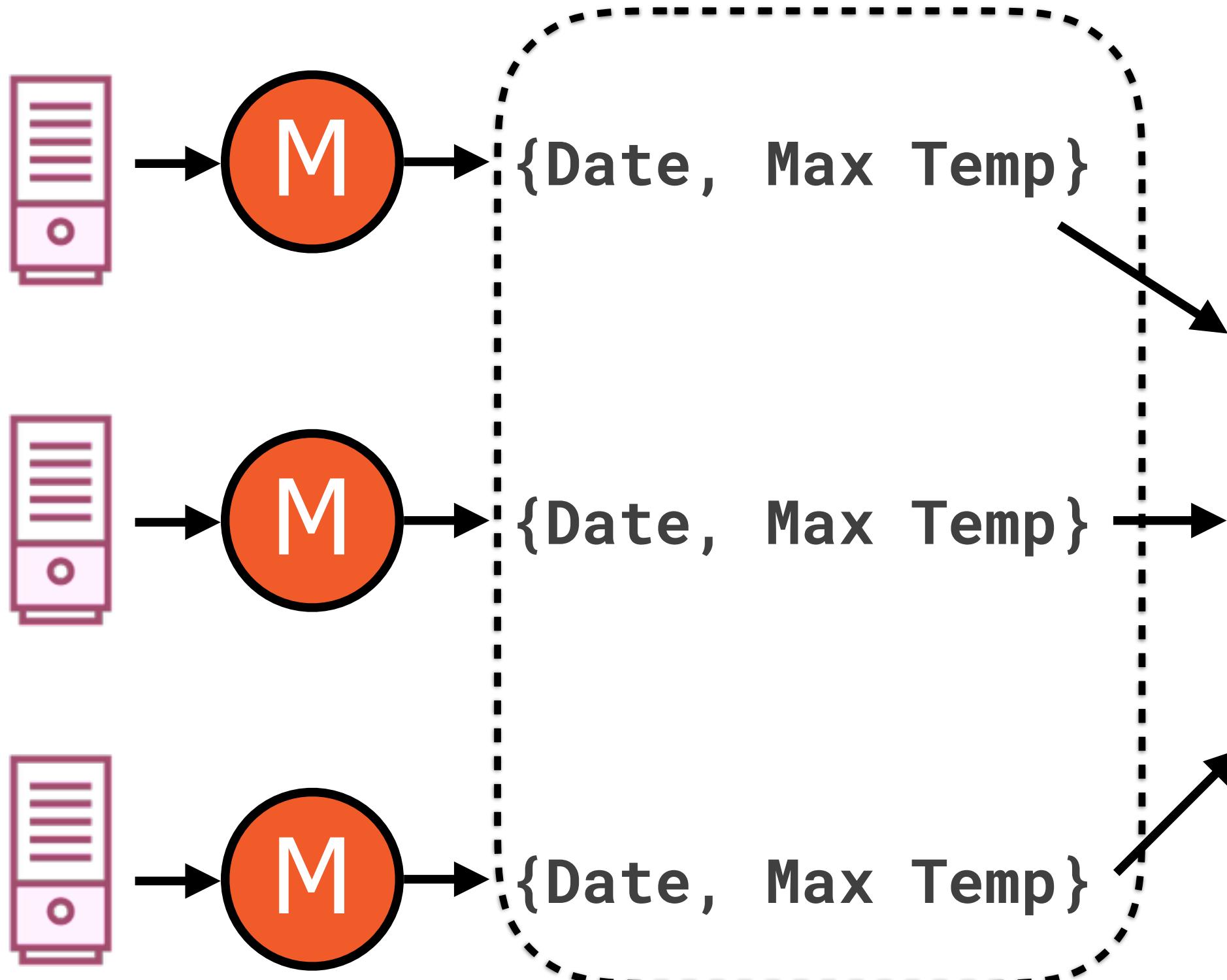
**The map phase
does the bare
minimum needed to
get the maximum
temperature**

Typical MapReduce Flow



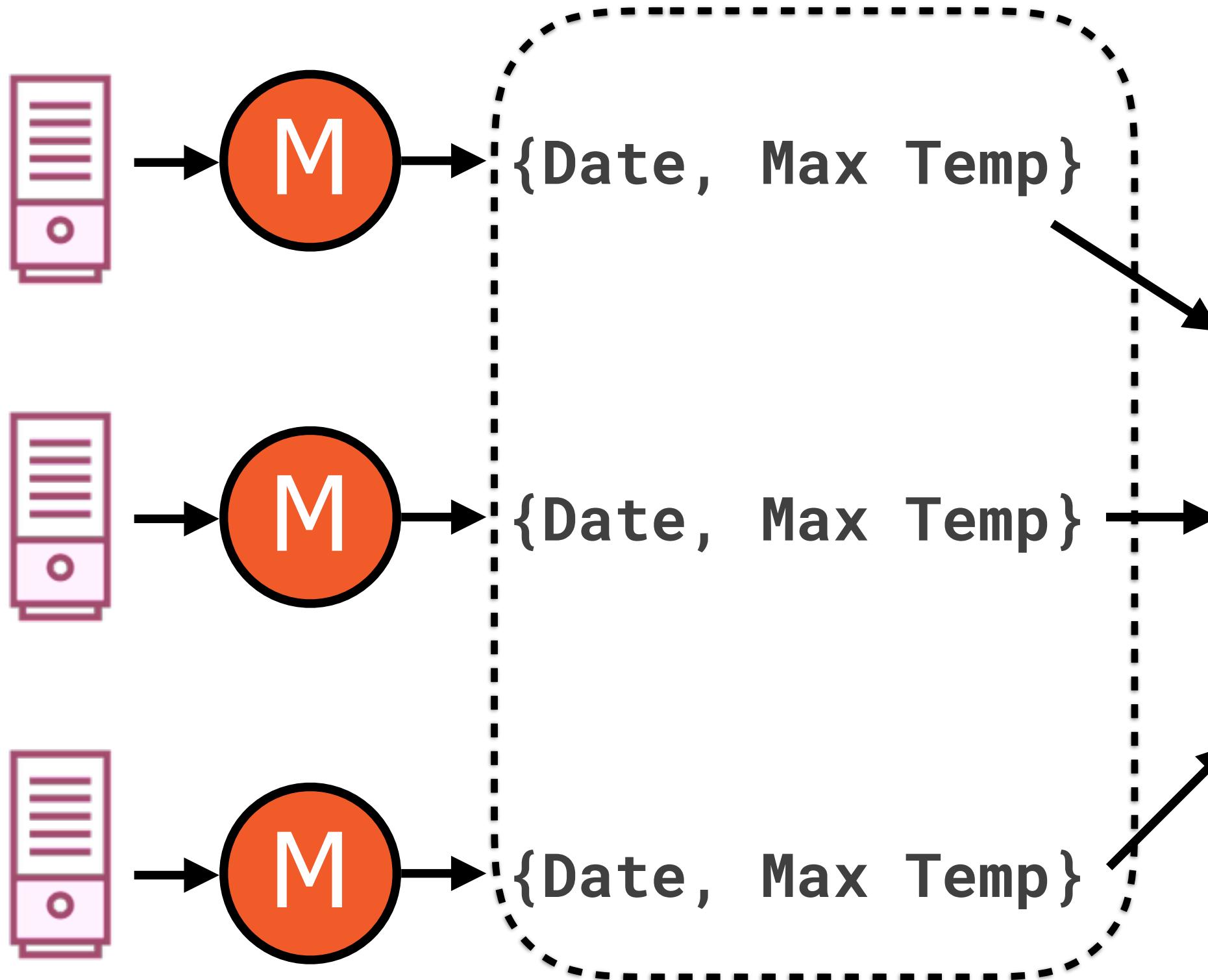
It could do more

Typical MapReduce Flow



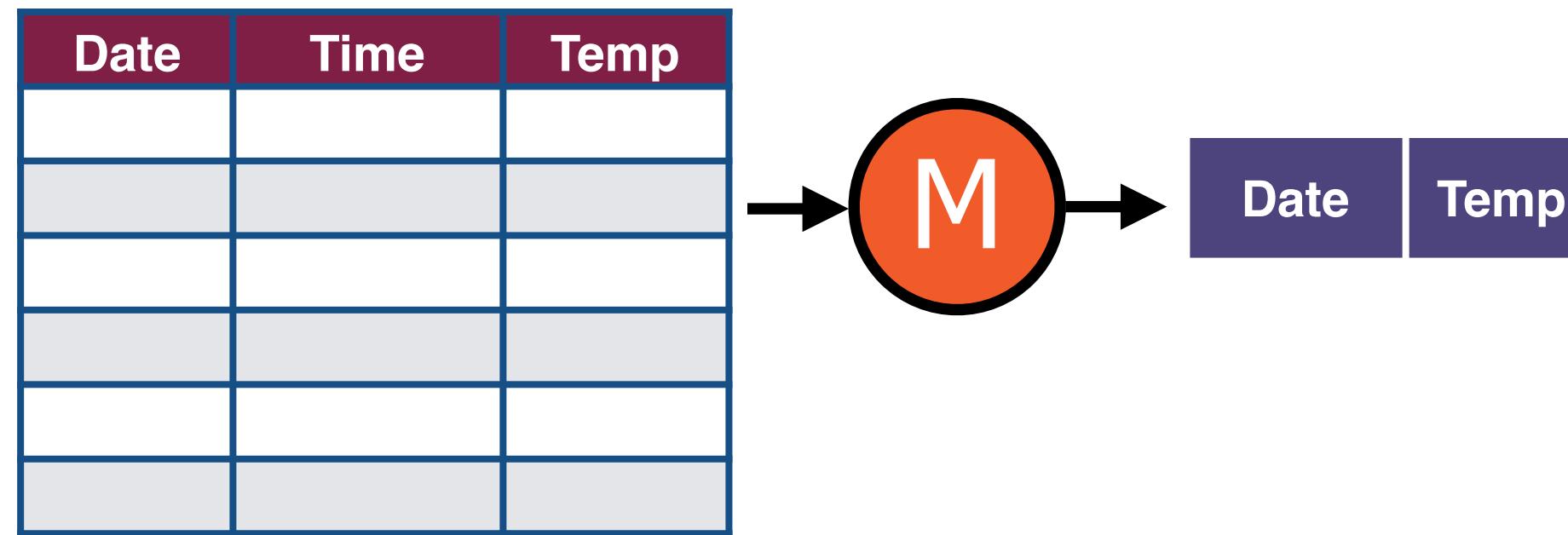
What if we could implement some combining logic at the map stage itself?

Typical MapReduce Flow



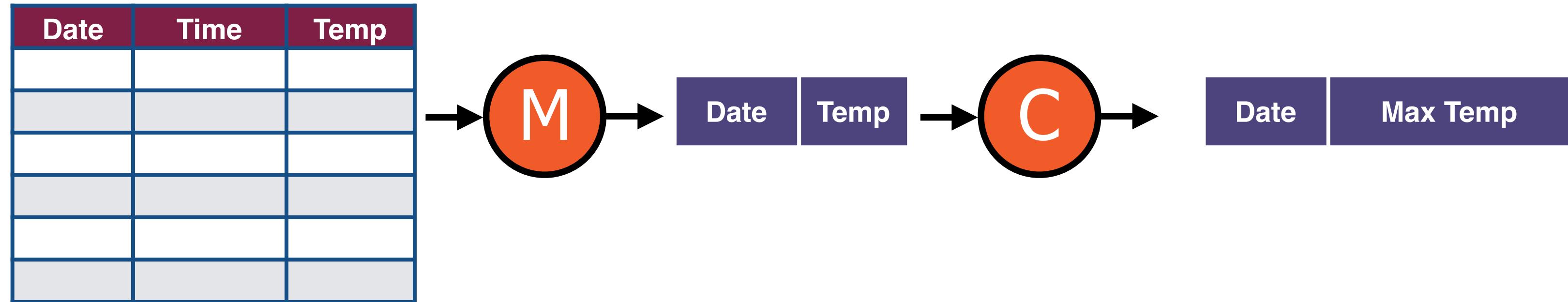
A combiner
performs a part of
the reduce step
on the map node

Combiner Function



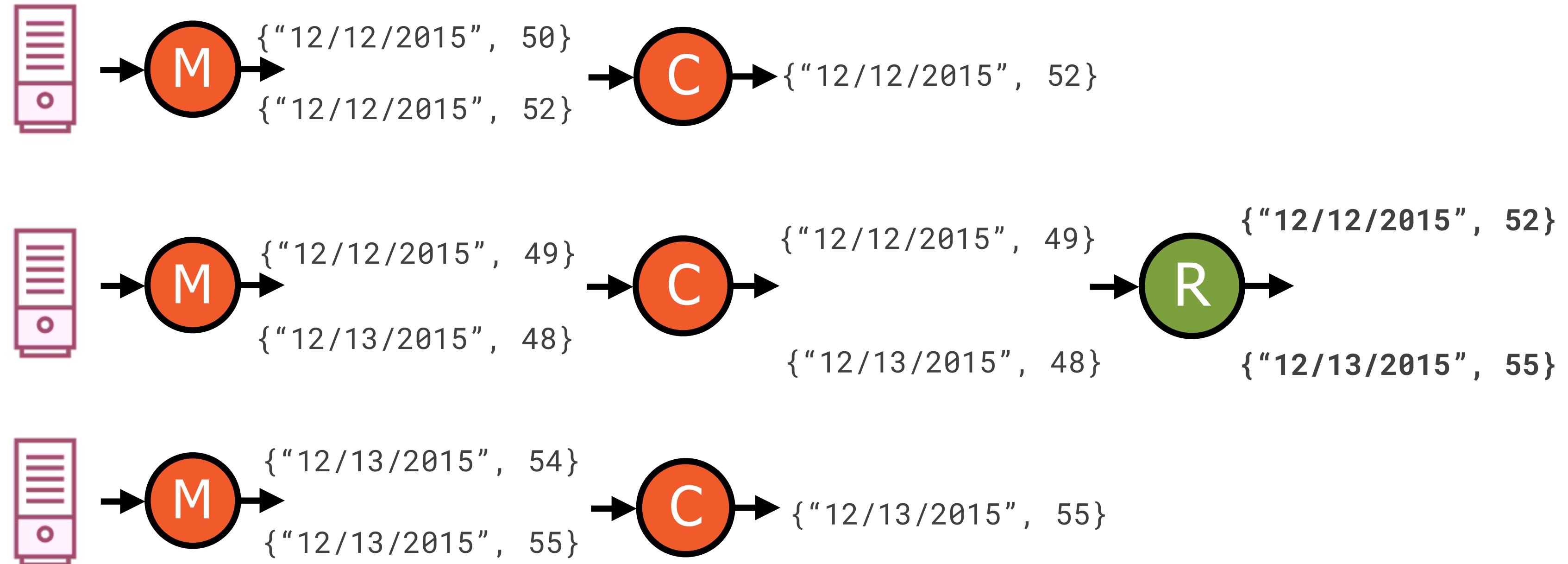
1 Mapper

Combiner Function

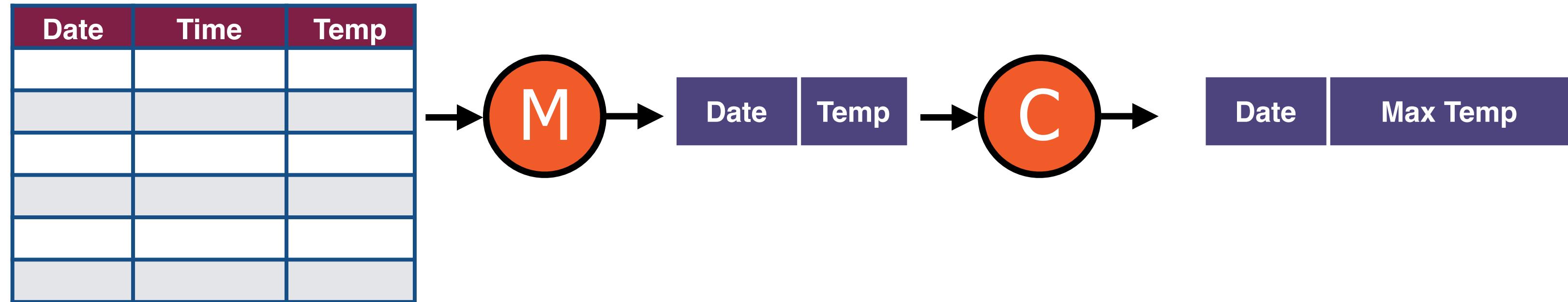


Combine values with the
same key before they are
copied over to the reducer

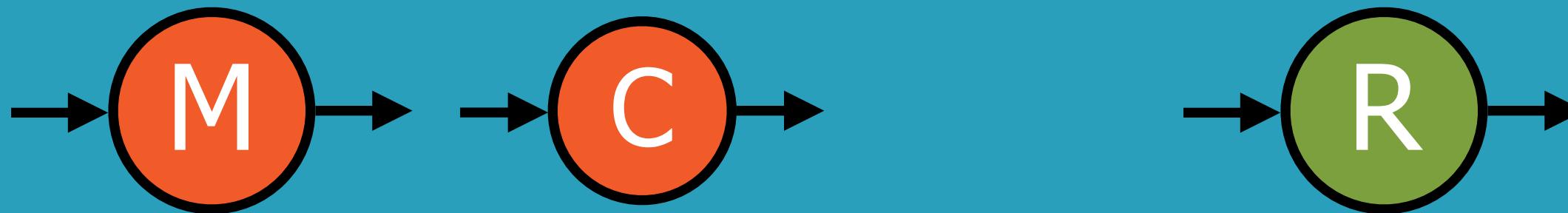
Combiner Function



Combiner Function



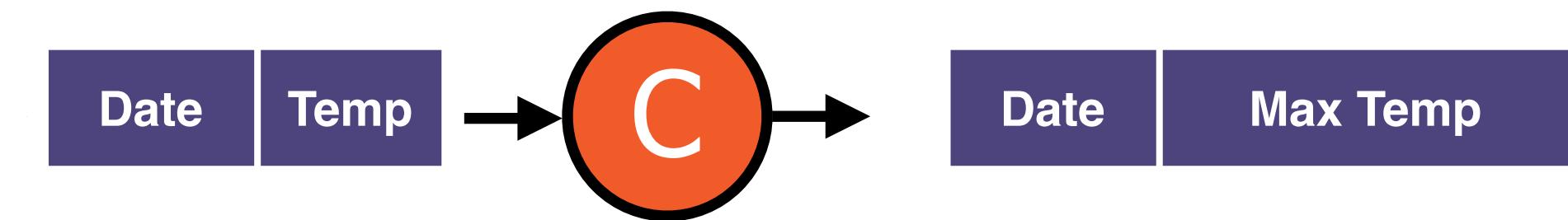
Combine values with the same key before they are copied over to the reducer



A combiner works on the mapper output before it is sent to the reducer

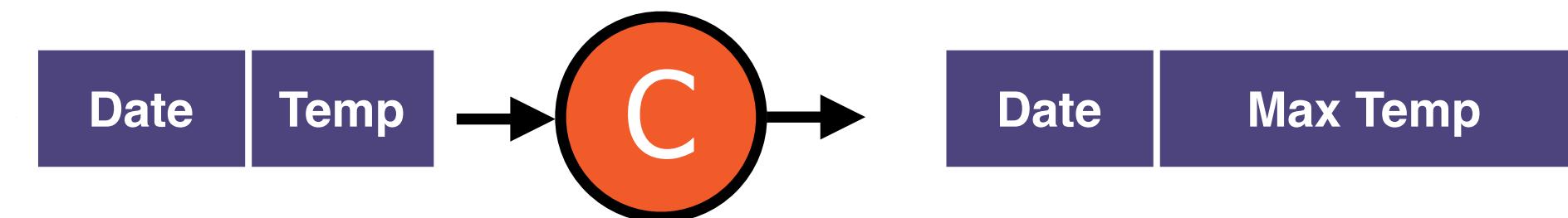
Combiner Function

This is the same logic that happens in the Reducer!



Combiner Function

We can use the Reducer class directly as the Combiner



Demo

**Use the Reduce class as the combiner
in our ViewCount MapReduce**

What Are the Advantages of Combiners?

Improve
parallelism

Reduce
data
transfer

Improve
parallelism

More processing
on the map node

Multiple map nodes
means greater
processing in parallel

Reduce
data
transfer

Results in fewer key, value
pairs in the map output

Less data transfer
to the reducer node

But...

Use combiners with caution

No impact
on the final
result

The result of the MapReduce
with and without combiners
should be the same

The combiners should not
change the logic of the
processing

Reducers as Combiners

Maximum

Minimum

Sum

The combiner performs the same operation as the reducer

Maximum

We've seen a detailed example of how the Combiner process is exactly the same as the Reducer

Maximum

7

4

5

3

7

0

11

2

11

16

2

14

16

16

Minimum

7

4

5

3

3

0

11

2

0

2

14

16

2

0

Sum

7
4
5
3

0
11
2

2
14
16

19

13

32

64

Reducers as Combiners

Maximum

Minimum

Sum

The combiner performs the same operation as the reducer

Reducers as Combiners

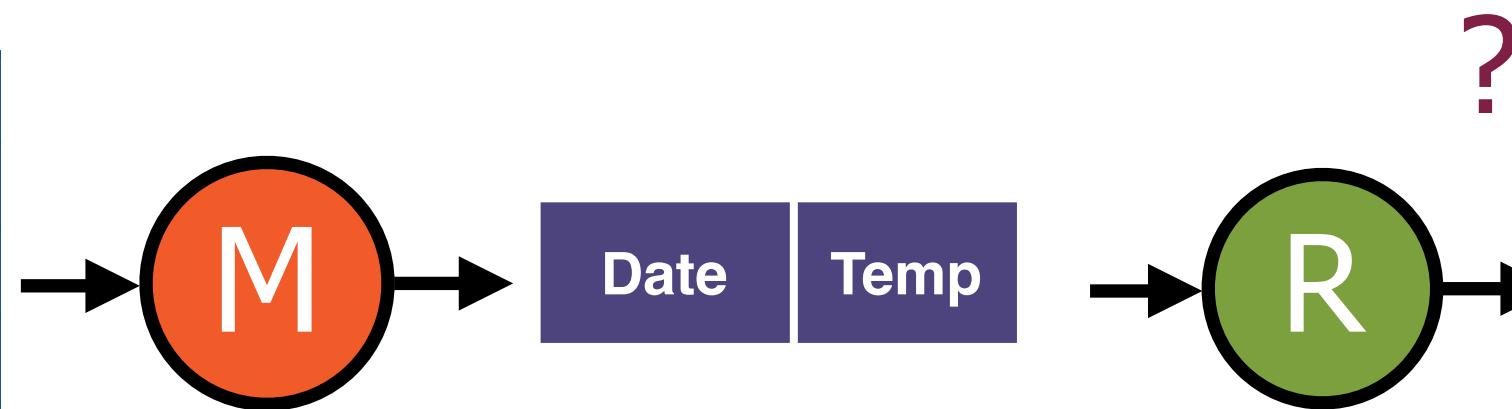


Average

**Combiner and reducer functions have
to be different!**

Average Temperature in a Day

Date	Time	Temp
12/12/2015	00:00	50
12/12/2015	01:00	52
12/12/2015	02:00	49
12/13/2015	00:00	48
12/13/2015	01:00	54
12/13/2015	02:00	55

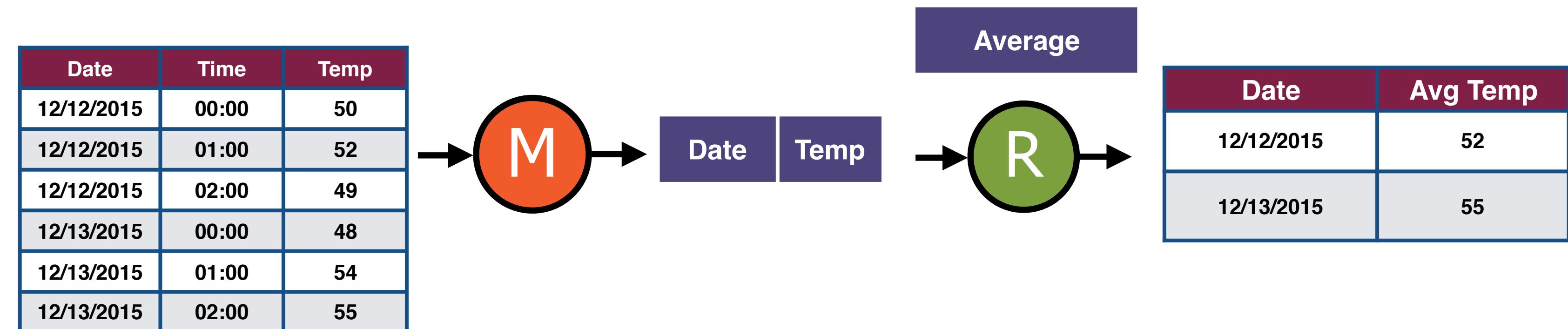


Date	Avg Temp
12/12/2015	50.3
12/13/2015	52.3

From each record
extract the date and
the temperature

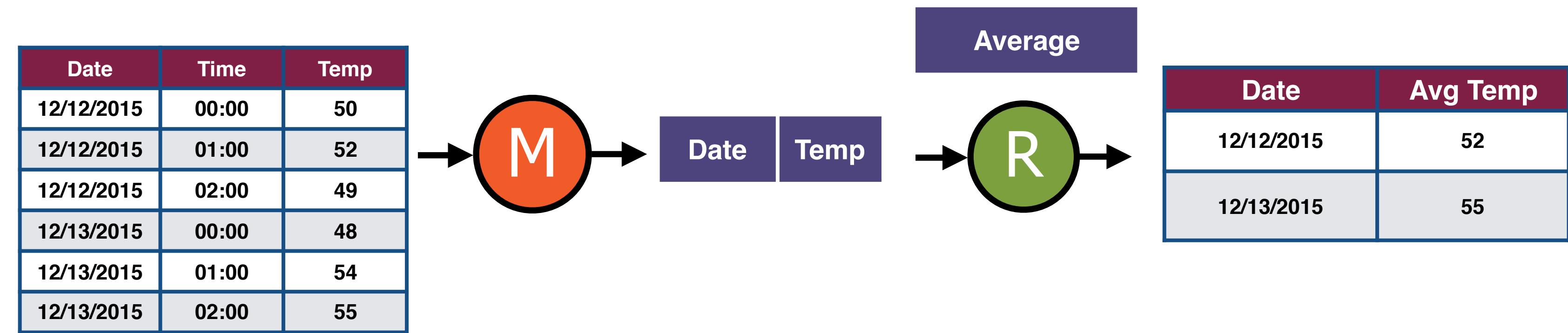
Without a combiner

Average Temperature in a Day



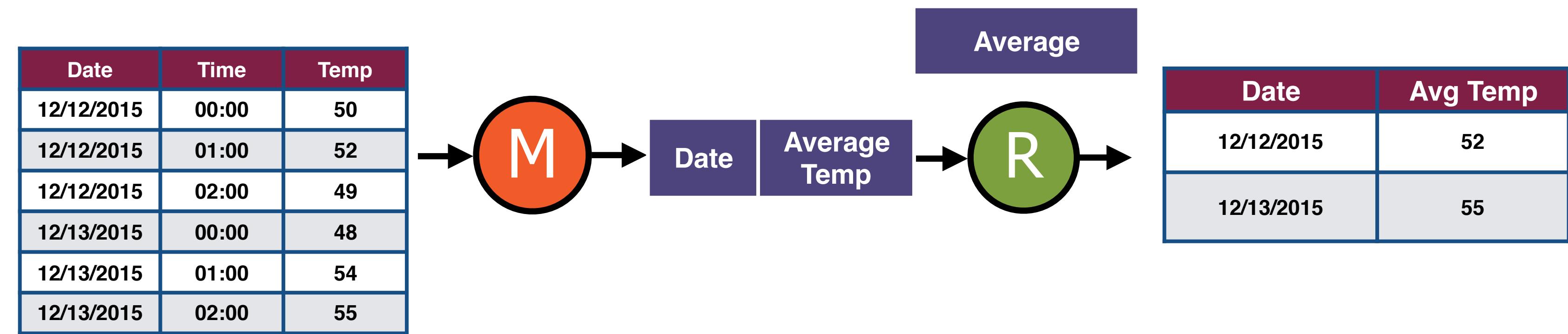
The reduce phase
finds the average
temperature for a
particular date

Average Temperature in a Day



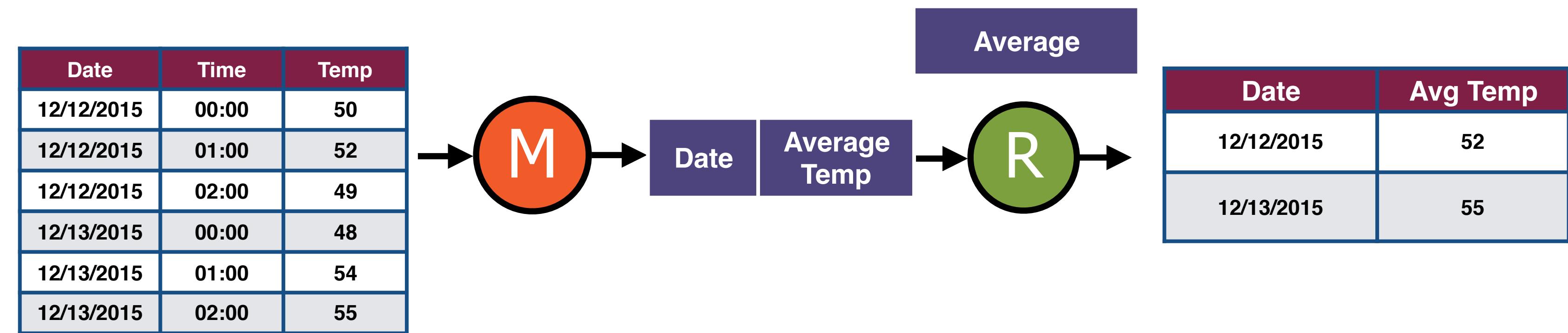
What if we used the
Reducer as the combiner
function?

Average Temperature in a Day



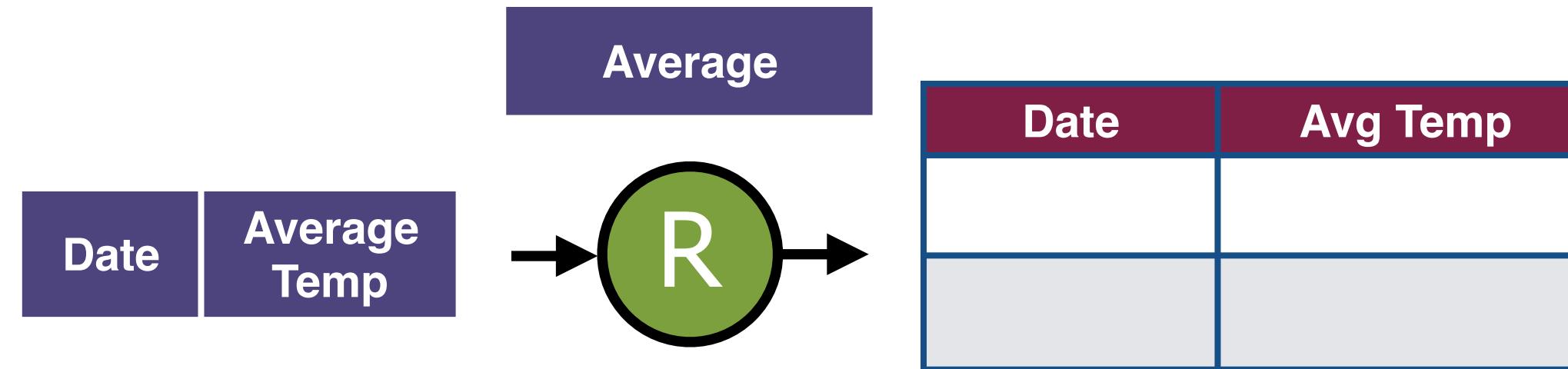
What if we used the
Reducer as the combiner
function?

Average Temperature in a Day



This is wrong!

Average Temperature in a Day



Average of a set
of numbers

<>

Average (Averages
of subsets)

Reducers as Combiners



Average

**Combiner and reducer functions have
to be different!**

Average

7
4
6
3

3
2
4

3
11
16

5

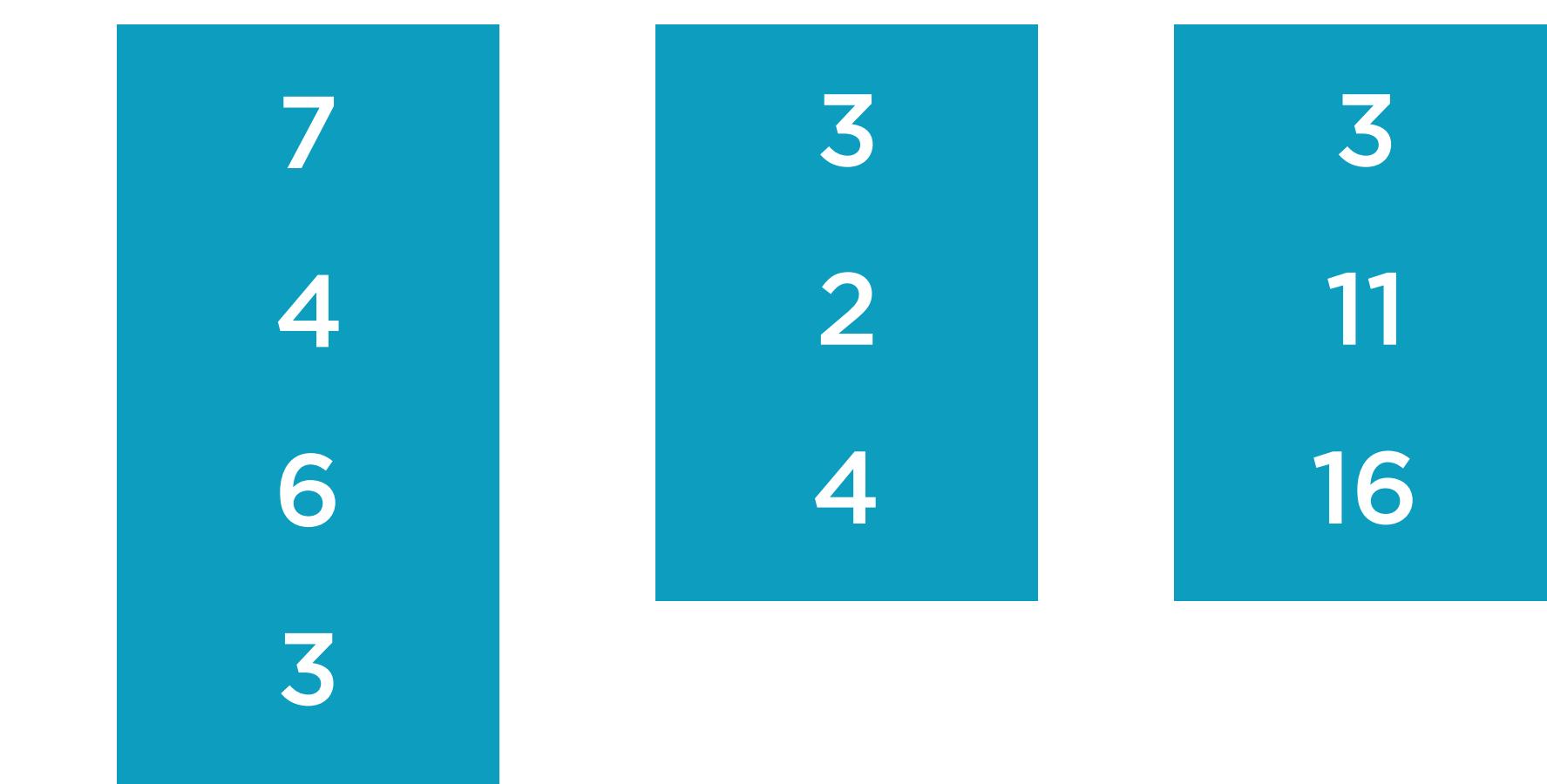
3

10

6

Wrong!

Average



Correct answer is

5.9

Summary

Control parallelism by changing the number of reducers

Understand the role of partitioning, shuffle and sort in a MapReduce task

Add a combiner to improve parallelism and reduce data transfer

Ensure that the combiner does not change the final result