Microsoft

MapReduce

October 2020

Brolinskyi Sergii

Plan of presentation

- Why do we need MapReduce
- Types of systems (monolithic vs distributed)
- The power of MapReduce
- Map
- Shuffle
- Reduce
- Typical operations of MapReduce
- Demo
- How to MapReduce
- Summary

Why do we need MapReduce

Billions of rows of raw data

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

7 people viewed your profile in the past 3 days

▲8% profile rank in the past 30 days

Two ways to build a system

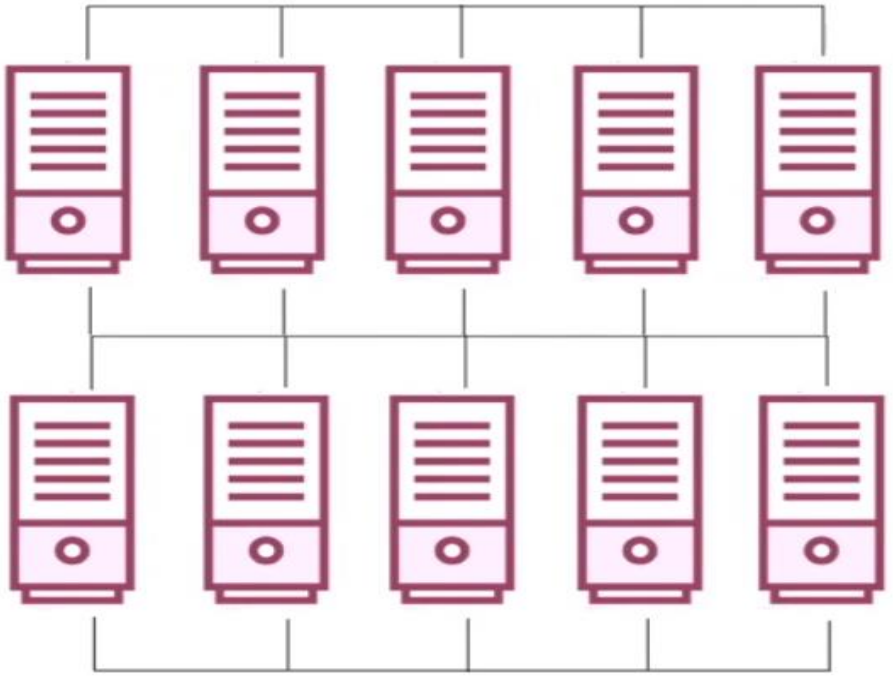
Monolithic

Distributed

Two Ways to Build a Team

A star who dribbles and shoots

A team of good players who know how to pass



Monolithic – only option until late 80s

- Advantage – easy to use
- Disadvantages – expensive; not scalable

Distributed:

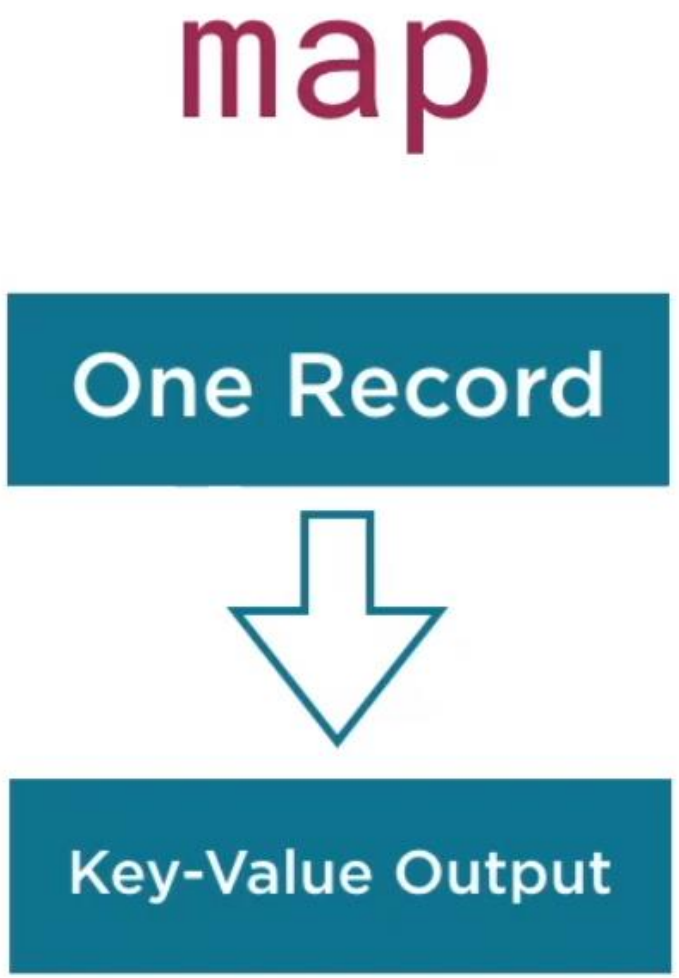
- Advantage – cheap,linear scale
- Disadvantages - partitioning; fault tolerance and recovery (what if some node goes down); parallel processing (that is where MapReduce is used)

The power of MapReduce

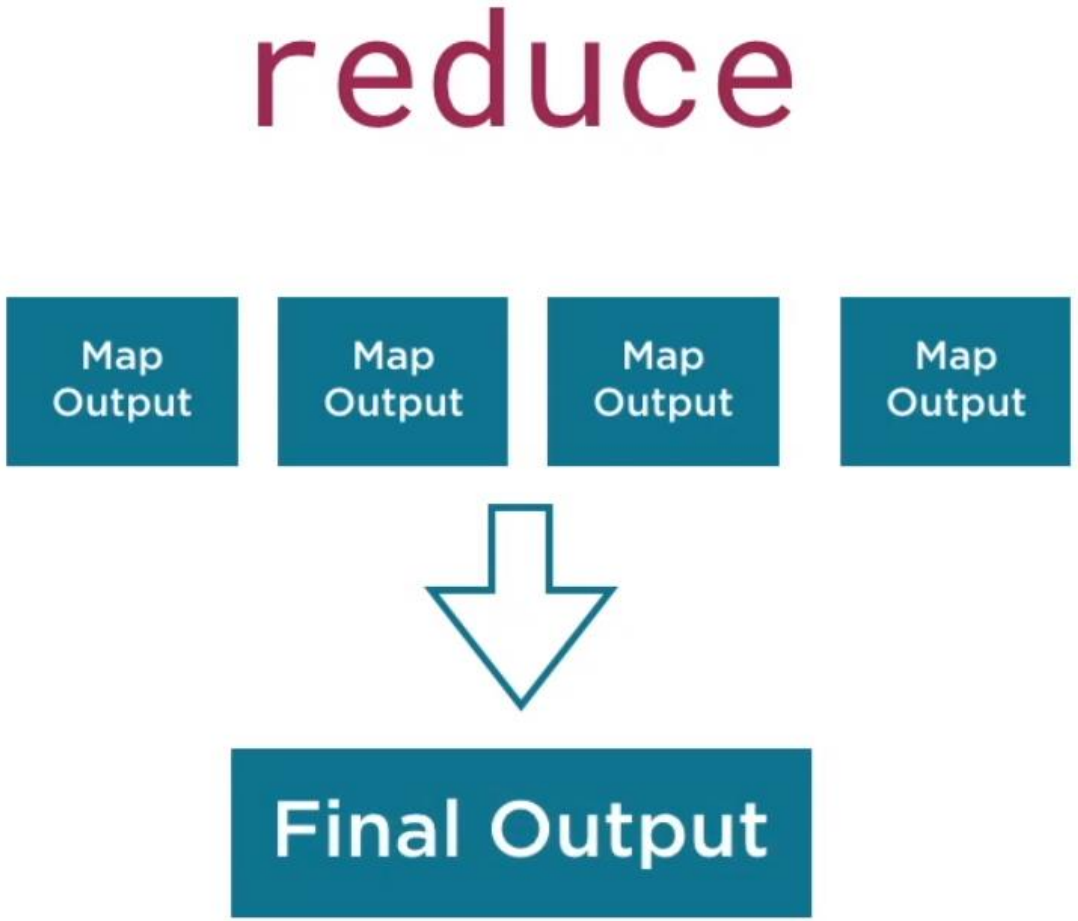
Abstraction

Parallelization

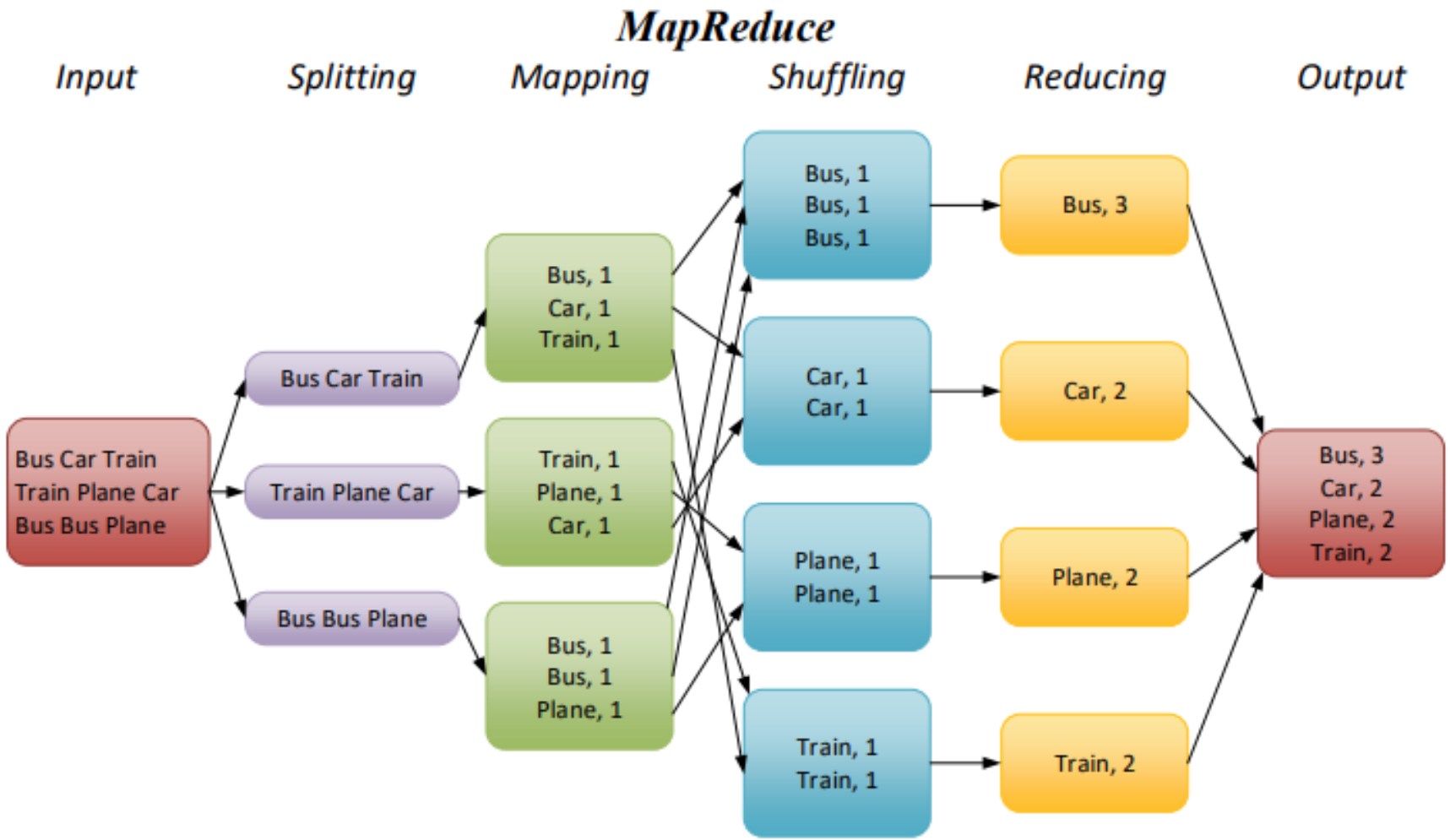
Map step



Reduce step


reduce
Map Output
Map Output
Map Output
Map Output
Final Output

Shuffle step



Why Shuffle and not Sort?

Big O complexity

SHUFFLE	**SORT**
O(N)	O(N log(N))

Why Shuffle and not Sort?

Items	O(N)	O(N LogN)	Overhead
10	10	33	3.3
1 000	1 000	9 965	9.9
1 000 000	1 000 000	19 931 568	19.9
1 000 000 000	1 000 000 000	29 897 352 854	29.9

Typical operations for MapReduce

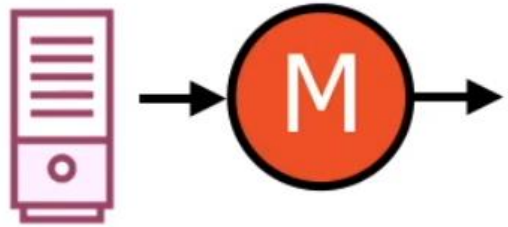
- Filtering
- Counting
- Ranking (like to 50%, bottom 5%)
- Min/Max/Avg
- Any other task that can be done in two stages (split into independent pieces and combine intermediate results)

Profile views

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

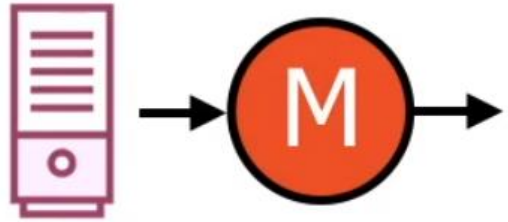
Map Flow

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



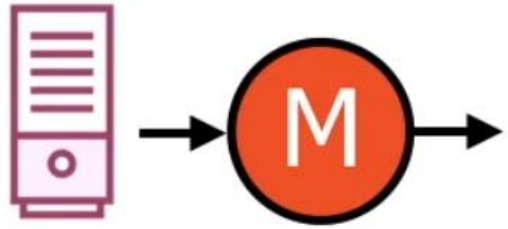
{Jitu, 1}
{Janani, 1}

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



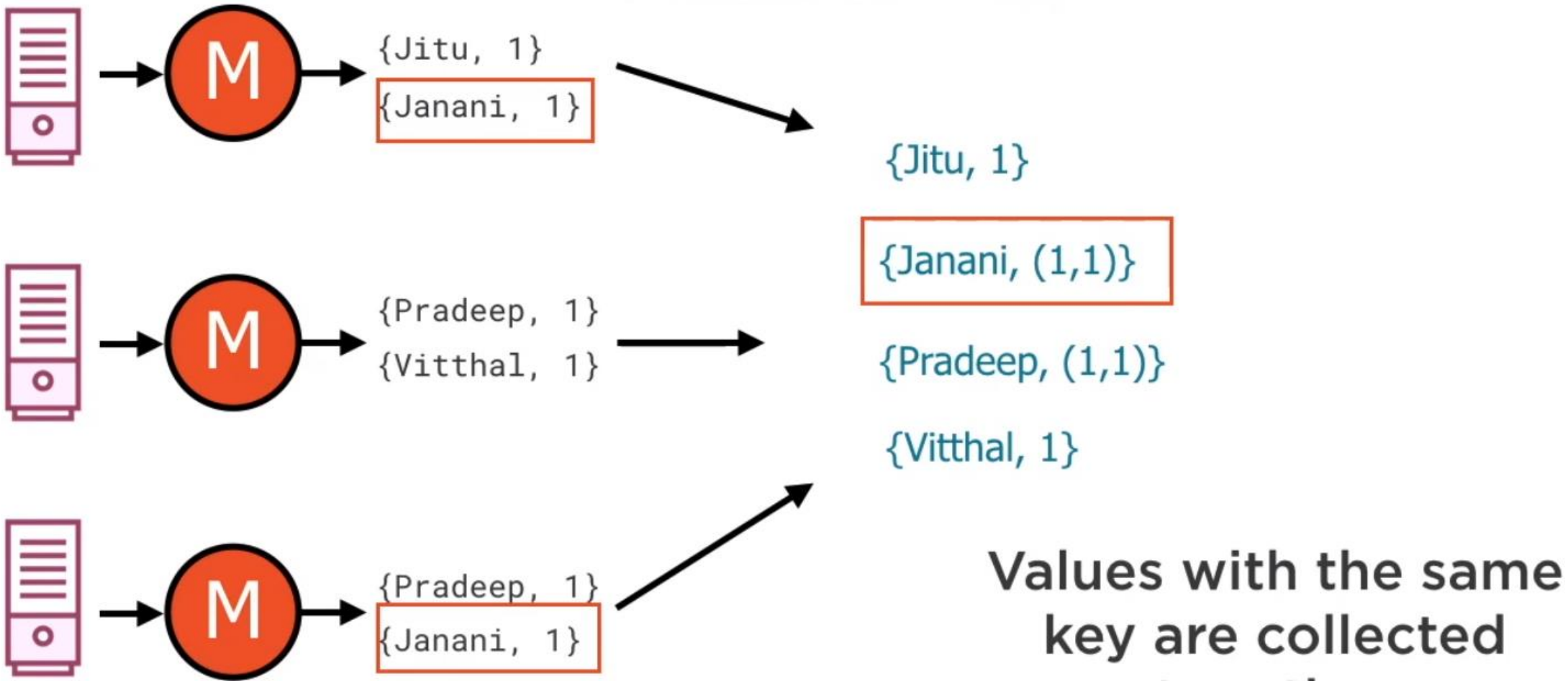
{Pradeep, 1}
{Vitthal, 1}

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



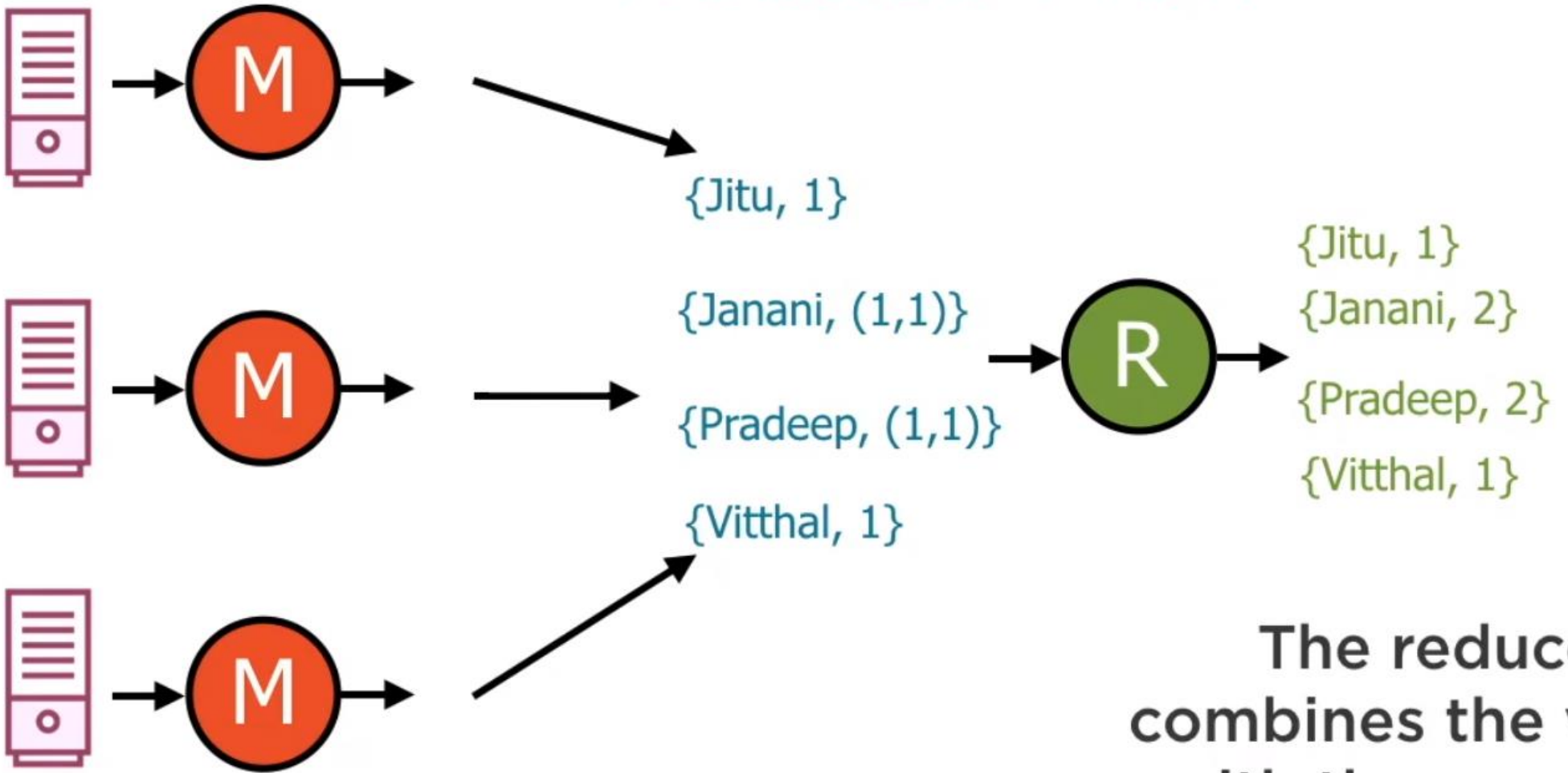
{Pradeep, 1}
{Janani, 1}

Reduce Flow



Values with the same key are collected together

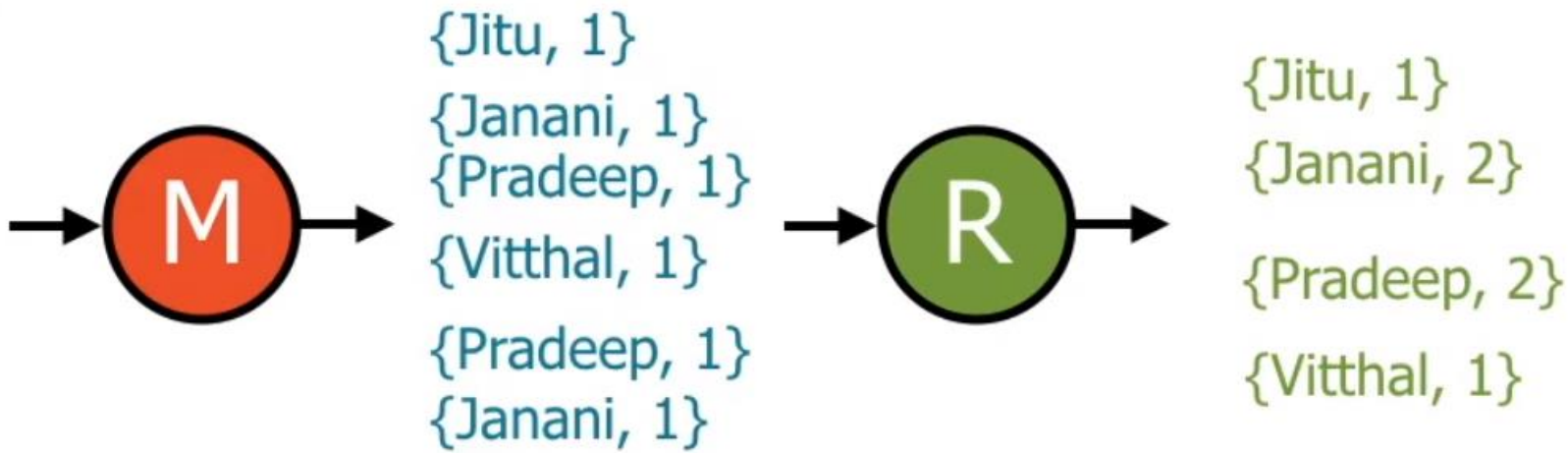
Reduce Flow



The reducer combines the values with the same key

MapReduce Flow

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



The basic form of EVERY MapReduce task

MapReduce pattern

How to MapReduce

Answer 2 questions:

- What are the {key; value} pair you need to setup for each step
- How the values should be combined

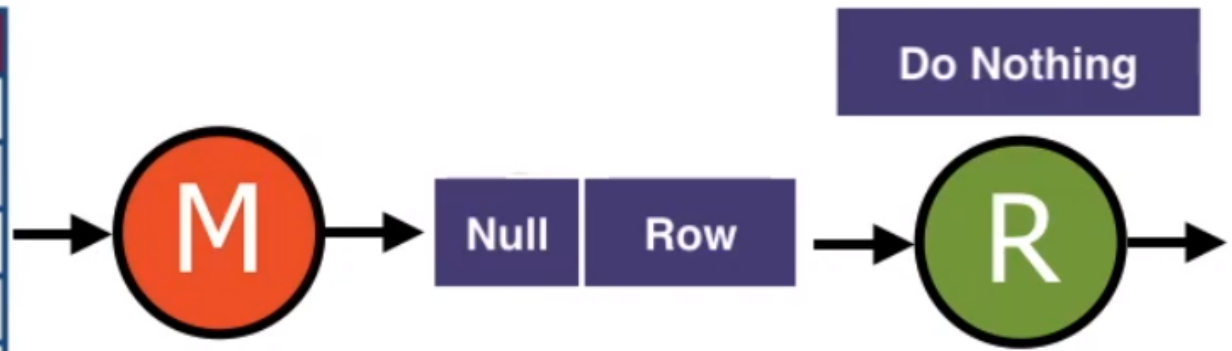
Building a User-ViewCount Map

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

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

Filter Views for a User

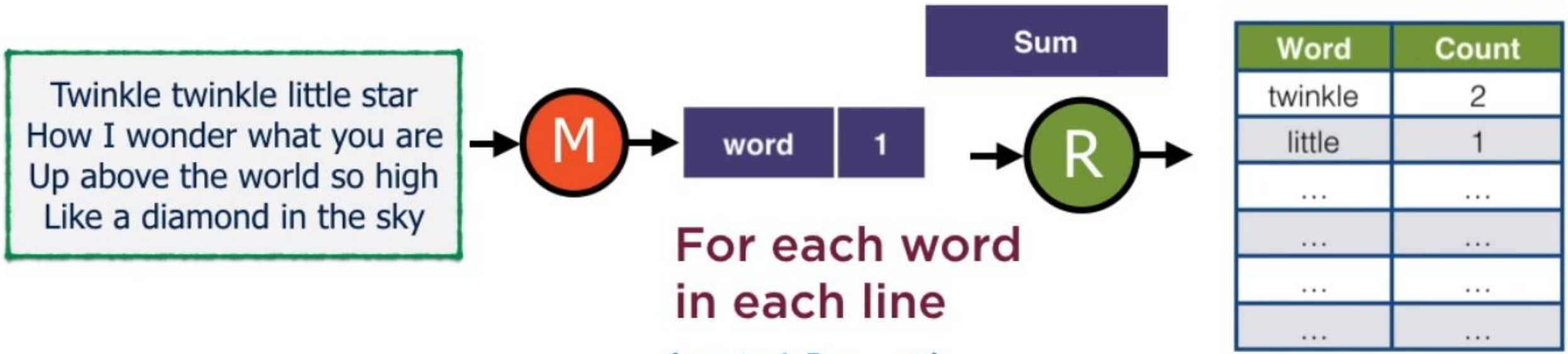
View	From	To
1	Janani	Jitu
2	Swetha	Janani
3	Shreya	Pradeep
4	Jitu	Vitthal
5	Shreya	Janani
6	Jitu	Pradeep



View ID	From	To
3	Shreya	Pradeep
6	Jitu	Pradeep

If the user matches the given user

Word Counts in a Document



Summary

It is complicated to always think about the parallel data processing and manually define the rules of how it should be done, so there is plenty of frameworks that add a level of abstraction so you would only need to think about what work should be done and those frameworks are built using MapReduce (example Hadoop) concept.

To solve your problem in a MapReduce way you need to:

- Define the {key; value;} on each step
- Choose the Reduce operation