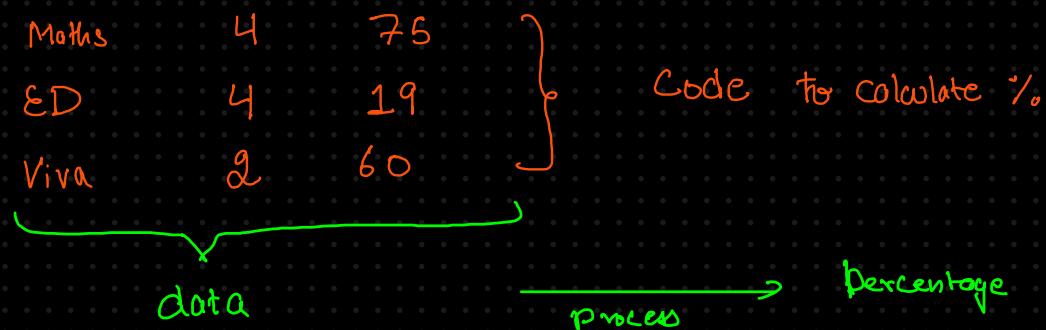




## Map Reduce

NSIT → marks ⇒ percentage



Mayork  
II



1



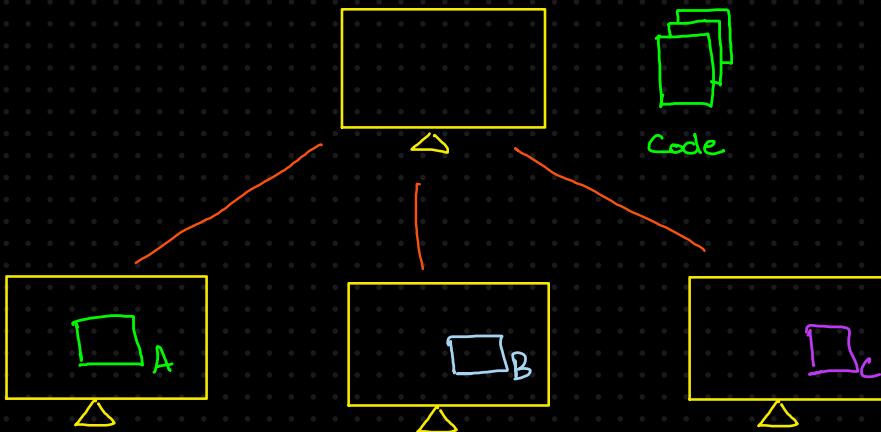
F2



F3

Google chrome  
Voting

Delhi



data → code

\* data locality  
+ parallel processing

Divide & Conquer  
Pseudo code

# Map Reduce

GMR  $\longrightarrow$  MR

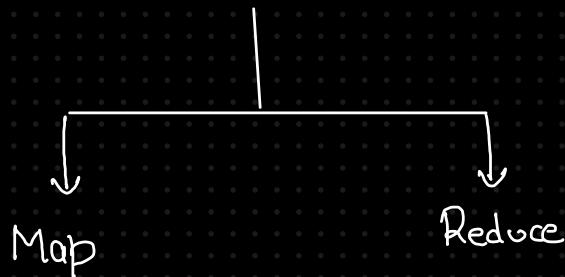
1. it is difficult to write
2. Theory is a lot imp  $\longrightarrow$  distributed processing  
Practical is not that imp.

Map Reduce is a programming model designed for processing large dataset in a distributed computing manner

- Scalability
- Fault tolerance
- parallelism

Smaller task

1. We will have data distributed Parallelly via hdfs which is required



input data is split into chunks, & each chunk processed by a mapper

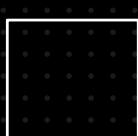
$\downarrow$   
Mapper produces intermediate output



Reducer combines it



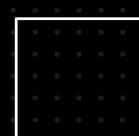
A



B



C



D



5 min

5 min

5 min

2 min

5 min

5 min

5 min

2 min

5 min

5 min

5 min

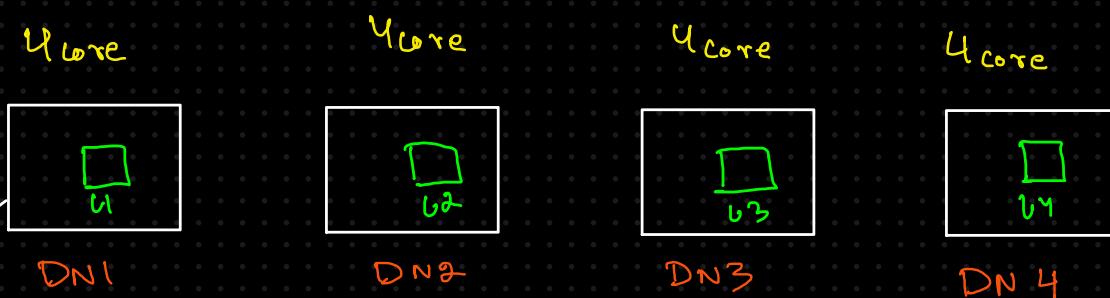
2 min

= 20 mins

= 7 min

Each and every Input or Output of MR job  
is a Key-value pair.

Say we have a 400 mb file in a hadoop cluster



1 core can process 1 mapper job on 1 block / input  
of data

4 cores

it can parallelly process 4 blocks / IS of data

total 16 cores

1 file → 32 blocks  
128mb block size

so in this case as blocks >> cores, we will process  
the job in batches.

## Word Count example

1. Input → text with sentences

2. Mapper → Value (`'hello'`, `'world'`)  
↓

(`('hello', 1)` (`'world'`, 1))

3. Combiner (optional)

4. Shuffle and sort

5. Reducers

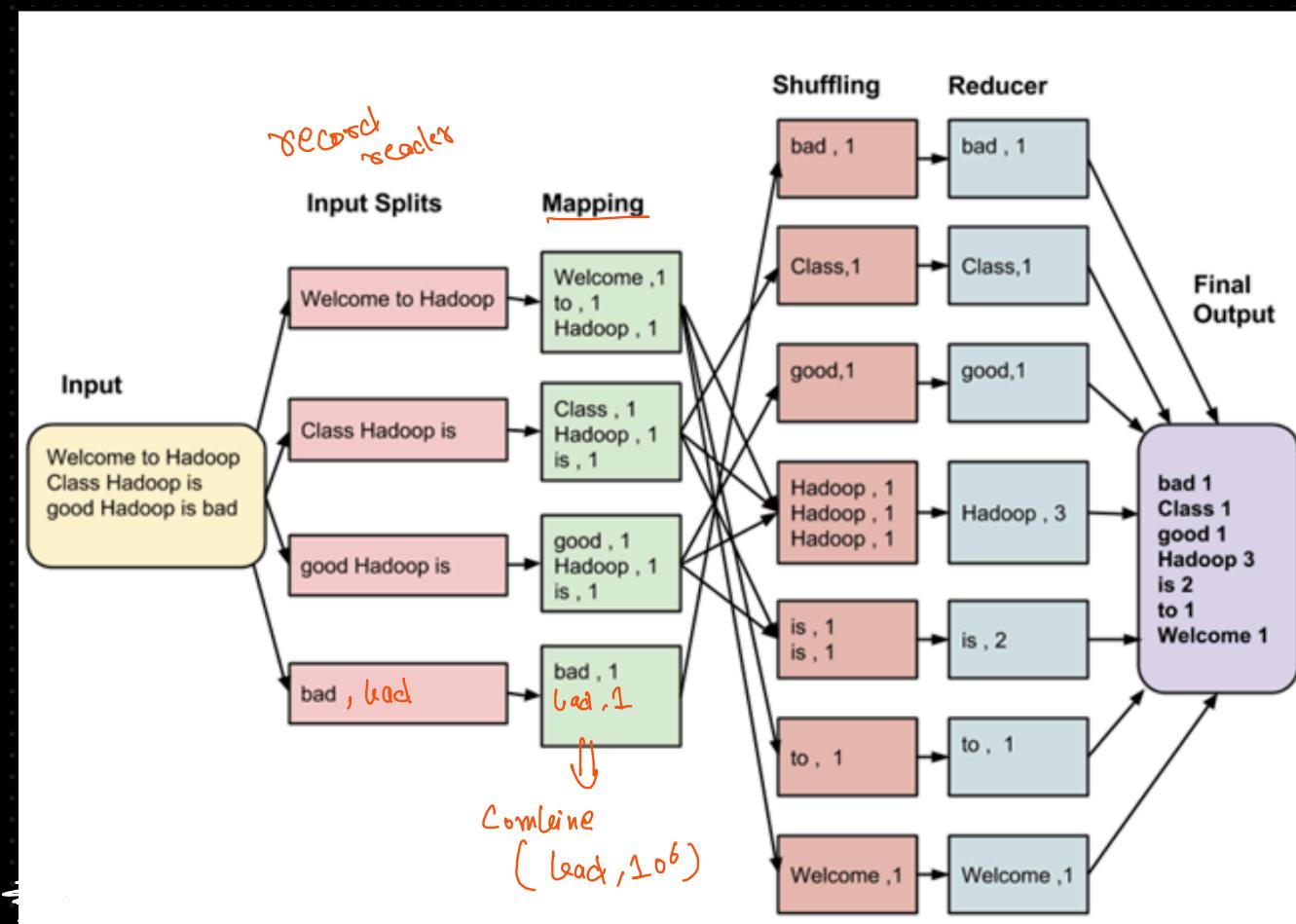
Aggregate the values

Goku, {1, 1, 1, 1, 1}

↓

(Goku, {43})

6. Final Output



Gohu vegeta Gohou → (Gohu,1) (vegeta,1) (Gohou,1)

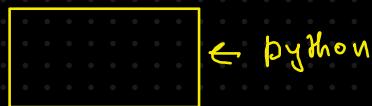
Gohou Frieza Gohu Gohu → (Gohou,1) (Frieza,1) (Gohou,1)  
(Gohu,2)

↓ Combiner

(Gohu,2)

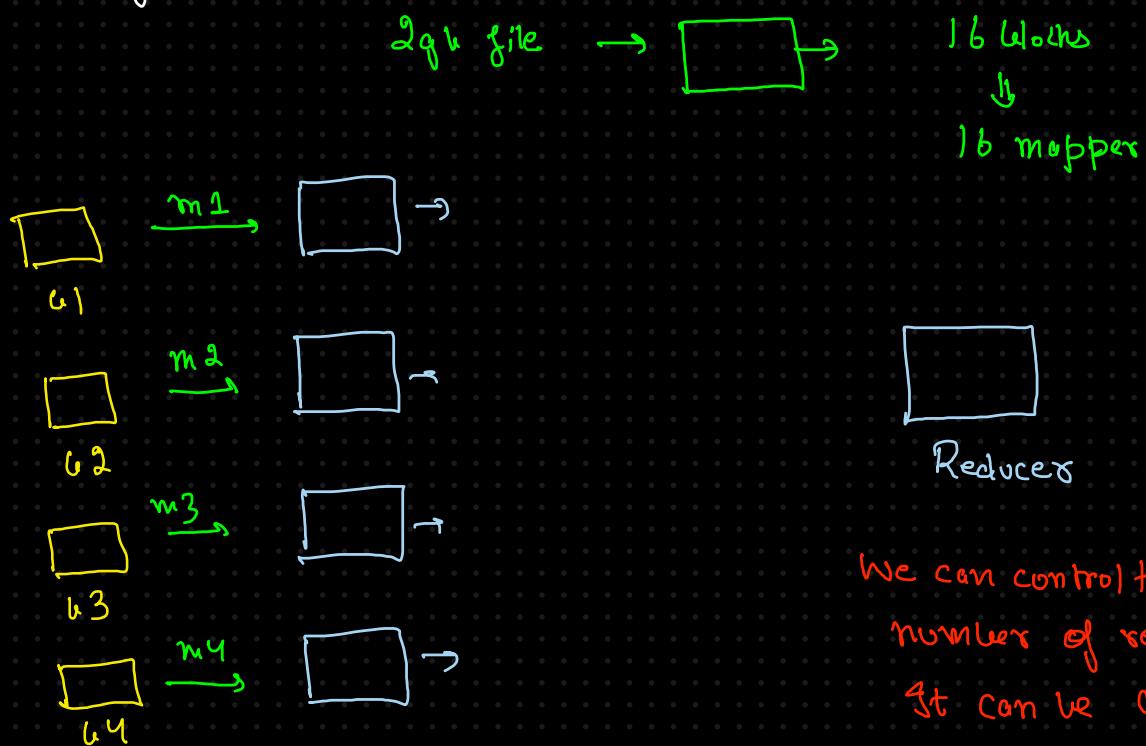
Step	Process	Output Example
Input Splitting	Splitting input file into HDFS chunks.	Split 1: Goku Vegeta Gohan , Split 2: Frieza Goku , etc.
Mapper Phase	Generating key-value pairs ( word , 1 ) from each split.	( "Goku" , 1 ) , ( "Vegeta" , 1 ) , ( "Gohan" , 1 )
Combiner Phase	Optional local aggregation at Mapper to reduce shuffle data.	( "Goku" , 2 ) , ( "Frieza" , 1 )
Shuffle and Sort	Grouping and sorting intermediate key-value pairs by key.	( "Goku" , [ 1 , 2 , 1 , 2 ] ) , ( "Frieza" , [ 1 , 1 , 1 ] )
Reducer Phase	Aggregating grouped data to produce final results.	( "Goku" , 6 ) , ( "Frieza" , 3 )
Final Output	Writing final aggregated results to HDFS.	Frieza 3 , Gohan 3 , Goku 6 , Vegeta 2

$(\text{word}, 1)$   
 $(\text{word}, 1) \Rightarrow 10^6$       ↗  
 $\vdots$   
 $(\text{word}, 1)$



MR doesn't make sense for smaller data.

No. of Mappers ? = number of blocks



We can control the  
number of reducers  
It can be 0, 1, ... n



2. Map reducer with more than one reducer

