Introduction to MAP-REDUCE

Agenda

- What is MapReduce
- Use cases of MapReduce
- MapReduce Logical dataflow – Input split, record reader, Mapper, Sort, Shuffle, Reducer, Output
- MapReduce Design and Execution

Apache Hadoop Core Components

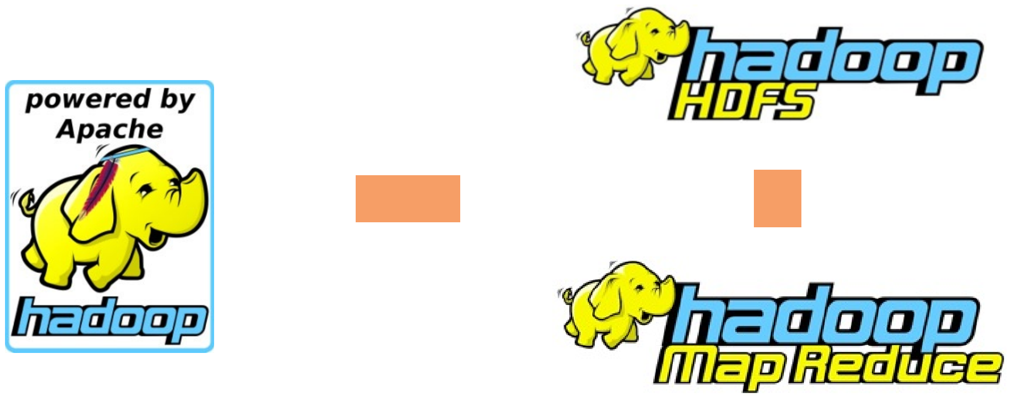
HDFS (storage) - stores data in chunks by splitting it into 64MB each block, is from the "Infrastructural" point of view

MapReduce (processing) - is from the "Programming" aspect

–A Java framework for processing parallelizable problems across huge datasets, using commodity hardware, in a distributed environment

–Google has used it to process its "big-data" sets (~ 20,000 PB/day)

–Can be implemented in many languages: Java, C++, Ruby, Python etc.



Problem statement: To count the frequency of words in a file

Input file name : **secret.txt** and has just 2 lines of data. Contents of the file:

this is not a secret if you read it

it is a secret if you do not read it

THE EXPECTED OUTPUT AS BELOW

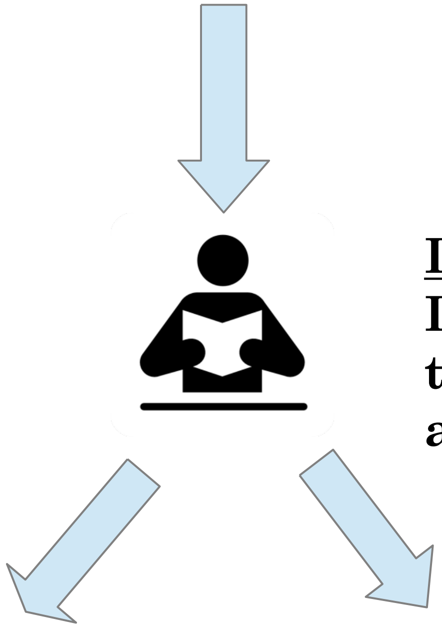
This	**1**
is	**2**
not	**2**
a	**2**
secret	**2**
if	**2**
you	**2**
do	**1**
read	**2**
it	**2**

2 stage approach using MAP & REDUCE

THE MAP STAGE

this is not a secret if you read it

(The above is the first line in the input file)



<u>I am RECORD READER</u>
I process one line at a time from the input file and produce 2 outputs

Output 1 (a number) **Output 2 (entire line)**

Output of the record reader

output 1 (A number)   output 2 (The entire line)

Output 1 is always called as KEY

Output 2 is always called as VALUE

(The same naming conventions would be used throughout the discussion hereafter)

KEY : 0 (file offset)

VALUE : this is not a secret if you read it (first line)

WHAT IS FILE OFFSET ?

Consider this file having 2 lines

It’s a new file

Which is almost used for nothing !

Each character in the file occupies one byte of data

First character of line one starts at location 0

Number of characters in the first line

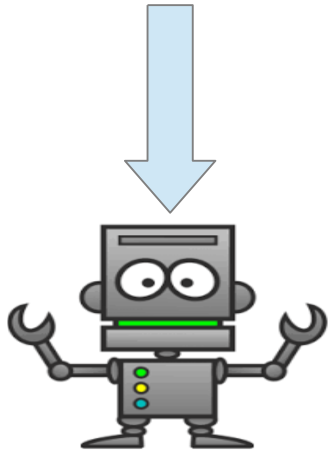
It’s : 4	space : 1	(total 5)
a : 1	space : 1	(total 2)
new : 3	space : 1	(total 4)
file : 4	new line : 1	(total 5)

Total of 16 characters or 16 bytes. The next line would begin at location **17.** File offset for next line is **17**

Record reader's o/p to mapper

Output of the record reader is fed to the MAPPER

0 ***(KEY)*** ***this is not a secret if you read it*** ***(VALUE)***



I'm the MAPPER
I can be PROGRAMMED !
I accept only one key value pairs as input and produce key-value-pairs as output

- Mapper can process only one key & value at a time

- Produce output in key, value pairs based on what its programmed to perform

Programming the mapper

- Mapper can be programmed based on the problem statement

- The input is a key value pair (file offset, one line from file)

  0,this is not a secret if you read it

- In the word count problem we shall program the mapper to do the following

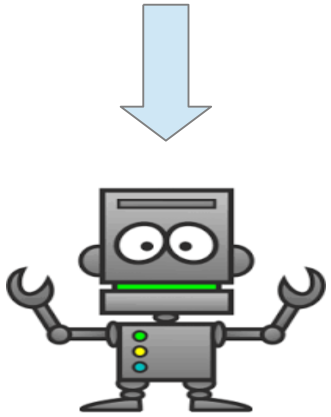
  Step1 : Ignore the key (file offset)

  Step 2: Extract each word from the line

  Step 3: Produce the output in key value pairs where key is each word of the line and value as 1 (integer/a number)

Output of the mapper

0,this is not a secret if you read it



this 1 ✓

is 1 ✓

not 1

a 1

secret 1

if 1

you 1

read 1

it 1

The sort operation

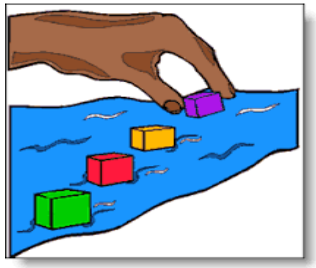
(This happens in the memory of the mapper machine)

Output of the mapper is fed into the **sorter** which sorts the mapper output in ascending order of the KEYS ! (lexicographic ordering or dictionary ordering since the keys are of string type)

Consider the simple example

K	V
Cathy	**34**
Alice	**10**
Bob	**89**

SORTER



K	V
Alice	**10**
Bob	**89**
Cathy	**34**

Note : Sorter can be reprogrammed (overridden) to sort based on values if required. Its called the sort comparator.

Input to the sort phase

this 1

is 1

not 1

a 1

secret 1

if 1

you 1

read 1

it 1

it 1

is 1

a 1

(keys not sorted)
(partial output of the mapper is shown here)

Output of the sort phase

a 1

a 1

do 1

if 1

if 1

is 1

it 1

it 1

it 1

not 1

not 1

read 1

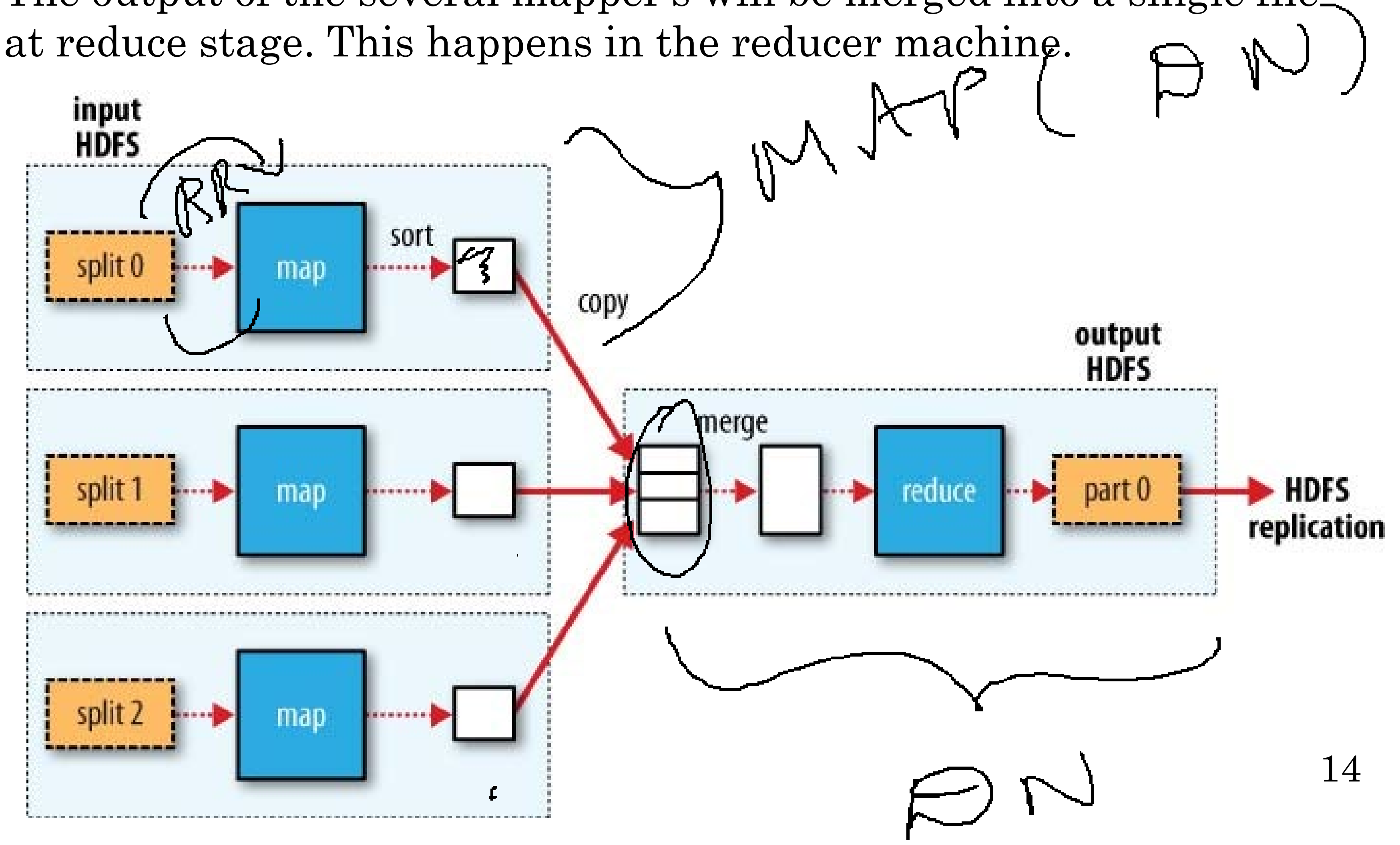
read 1

secret 1

REDUCE stage

It has 3 sub-stages - merge, shuffle and reducer operation

The output of the several mapper's will be merged into a single file at reduce stage. This happens in the reducer machine.



Shuffle/aggregate phase in REDUCE stage

- Shuffling is a phase where duplicate keys from the input are aggregated.

Consider the simple example

- Input is key value pairs (**contains duplicate keys**)

- Output is a set of key value pairs **without duplicates**

Key	value
Apple	2
Apple	4
Mango	1
Orange	11
Orange	06

SHUFFLER

Key	value
Apple	2, 4
Mango	1
Orange	11, 06

Shuffle operation at the reduce stage

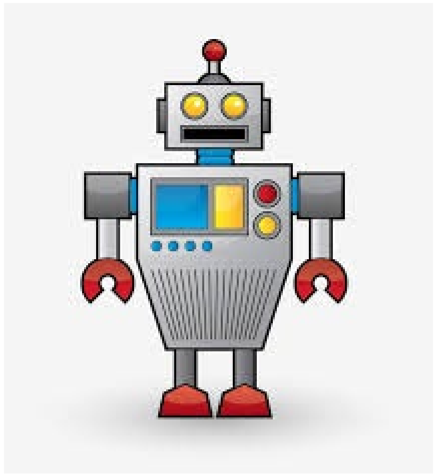
a	1
a	1
do	1
if	1
if	1
is	1
is	1
it	1
it	1

SHUFFLER

a	1, 1
do	1
if	1,1
is	1,1
it	1,1

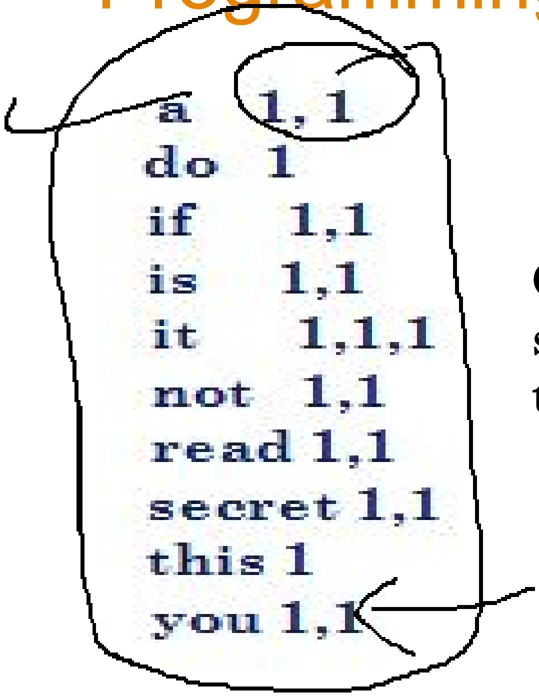
The REDUCER operation in REDUCE stage

Reducer accepts the input from the shuffle stage



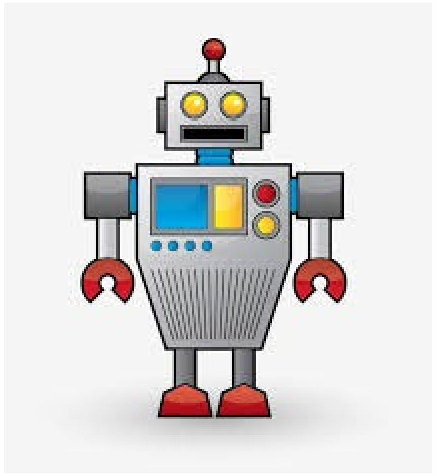
Reducer produces output in key value pairs based on what it is programmed to do as per the problem statement

Programming the reducer



a	1, 1
do	1
if	1,1
is	1,1
it	1,1,1
not	1,1
read	1,1
secret	1,1
this	1
you	1,1

Output of the shuffle is the input to the REDUCER



Reducer can handle only one key value pair at a time

Step1: Input to the reducer is **a 1, 1**

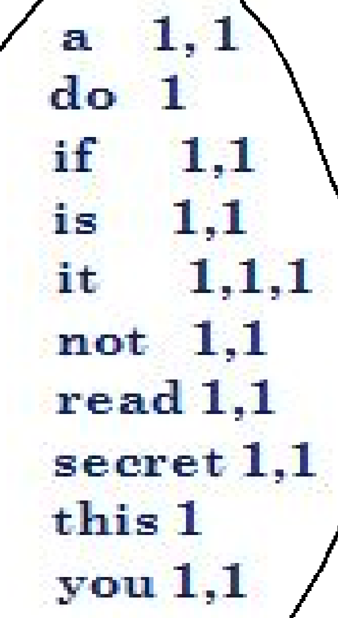
Step2: The reducer must add the list of values from the input i.e
sum=1+1 = 2

Step3: Output the key and sum as output key, value pairs to an output file. The o/ would look like **a 2**

Step 4: Repeat the above operations (1,2&3) for entire input

Final output of the reducer

Reducer input



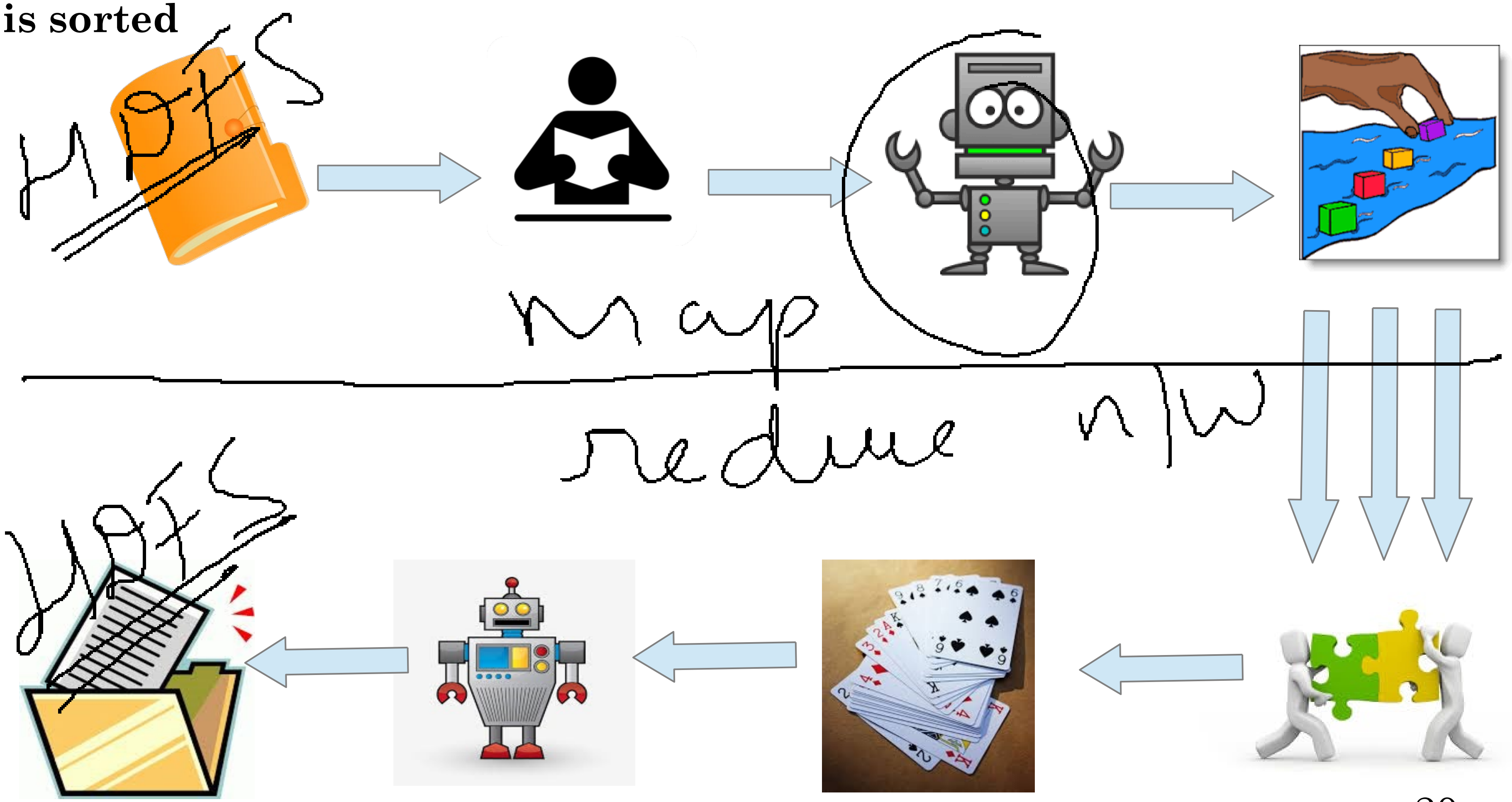
a	1, 1
do	1
if	1,1
is	1,1
it	1,1,1
not	1,1
read	1,1
secret	1,1
this	1
you	1,1

Reducer output (final o/p)

a	2
do	1
if	2
is	2
it	3
not	2
read	2
secret	2
this	1
you	2

Everything in a nutshell

Input file processed by RECORD READER o/p goes **to MAPPER & its o/p is sorted**



Final o/p file **REDUCER** **SHUFFLE** **MERGE**

ANAGRAM PROBLEM USING MAP REDUCE

What is anagram ?

MARY is a word and **ARMY** is another word which is formed by re arranging the letters in the original word MARY

Hence **MARY** and **ARMY** are ***anagrams***

POOL and **LOOP** are ***anagrams***. There could a lot of such examples.

Problem statement

To identify and list all the anagrams found in a document. Eg A book (a novel)

Input file name : **sample.txt** (a file in text format) and has 2 lines in the file.

File contents: mary worked in army

the loop fell into the pool

Expected output : (must contain all the anagrams)

mary army

loop pool

Programming the mapper

Mapper is programmed do the following

Step 1: Ignore the key from the record reader

Step 2: Split the words in the value (the full line)

mary works in army

[mary] [works] [in] [army] (the line is split)

Step 3: sort each word in dictionary order (lexicographic ordering)

mary after sorting would become ***amry***

Step 4: Output the sorted word as key and original word as value . The sample output of mapper would look like

KEY	VALUE
amry	mary

Step 5: Repeat the above steps for all the words in the line

Output of the record reader

This is going to the be output of the record reader after reading the first line of the file

Contents of the file : mary worked in army
loop fell into the pool

KEY	VALUE
file offset	**entire line of the file**
0	mary worked in army

The above (key-value pair) is now going to be fed into the mapper as an input

Output of the mapper after processing the entire file

KEY	VALUE
amry	mary
dekorw	worked
in	in
eht	the
amry	army
loop	loop
efll	fell
inot	into
eht	the
loop	pool

Output after sorting the keys

KEY	VALUE
amry	mary
amry	army
dekorw	worked
eht	the
eht	the
in	in
inot	into
loop	loop
loop	pool
efll	fell

Output after shuffling the keys (aggregation of duplicate keys)

KEY	VALUE
amry	mary, army
dekorw	worked
eht	the,the
in	in
inot	into
loop	loop, pool
efll	fell

Observation and inference

KEY	VALUE
amry	mary, army
dekorw	worked
eht	the,the
in	in
inot	into
loop	loop, pool
efll	fell

ANY BULBS LIGHTING UP ?

LOOK CLOSER !

There are some keys with more than one value. We need to only look at such key, value pairs

amry	**mary ,army**
eht	**the , the**
loop	**loop , pool**

Logic to list the anagrams

amry	mary ,army
eht	the,the
loop	loop,pool

Problem : We need to only print the values belonging to keys “**amry**” and “**loop**” since only their values qualify for anagrams.

We need to ignore the values belonging to the keys “**eht**” since its corresponding values do not qualify for being anagrams.

How to ignore the non anagram values ?

amry	mary ,army
eht	the ,the
loop	loop, pool

We need to program the following into the reducer

Step 1: Check if the number of values are > 1 for each key

Step 2: Compare the first and second value in the values list for every key, if they match, ignore them.

key	val1	val2
eht	the	the

Step 3: If the values don't match in step 2. Compose a single string comprising of all the values in the list and print it to the output file. This final string is the KEY and value can be NULL (do not print anything for value)

KEY = "mary army"   VALUE =" "

Step 4: Repeat the above steps for all the key value pairs input to the reducer.

Final output of the reducer

mary army

loop pool

The above output is for the case of a file with just 2 lines of data.

What if the file is 640MB in size ?

How does map reduce help in speeding up the job completion ?

HOW DOES MAP REDUCE SPEED UP THE PROCESSING ?

What is the input file used in this example is 640MB instead of just containing 2 lines ?

- The HADOOP framework would first split the entire file into 10 blocks each of 64MB
- Each 64MB block would be treated as a single file
- There would be one record-reader and one mapper assigned to each such block
- Output of all the mappers would finally reach the reducer (one reducer is used by default) however we can have multiple reducers depending on degree of optimization required
-

Why MapReduce?

- Scale *out* not scale *up:* MR is designed to work with commodity hardware
- *Move* code where the data is: cluster have limited bandwidth
- *Hide* system-level details from developers: no more race condition, dead locks etc
- Separating the *what* from *how:* developer specifies the computation, framework handles actual execution
- *Failures* are common and handled automatically
- Batch processing: access data sequentially instead of random to avoid locking up
- Linear Scalability: once the MR algorithm is designed, it can work on any size cluster
- *Divide* & *Conquer*: MR follows Partition and Combine in Map/Reduce phase
- High-level *system details*: monitoring of the status of data and processing
- Everything happens on top-of a *HDFS*

Use case of MapReduce?

-
- Mainly used for searching keywords in massive amount of data
- Google uses it for wordcount, adwords, pagerank, indexing data for Google Search, article clustering for Google News
- Yahoo: “web map” powering Search, spam detection for Mail
- Simple algorithms such as grep, text-indexing, reverse indexing
- Data mining domain
- Facebook uses it for data mining, ad optimization, spam detection
- Financial services use it for analytics
- Astronomy: Gaussian analysis for locating extra-terrestrial objects
- Most batch oriented non-interactive jobs analysis tasks

Summary

Flow of map reduce using an example of word count

Anagram problem using map reduce

Why map reduce

Use cases of map reduce