# Hadoop and MapReduce

Original Slides by Dr Sandeep Deshmukh, SadePach Labs

Modifications by Dr Amey Karkare, IIT Kanpur

http://hadoop.apache.org/

# Hadoop

- Framework that allows for the distributed processing of large data sets
  - across clusters of computers
  - using simple programming models.
- Designed to scale up from single servers to thousands of machines, each offering local computation and storage.
- Designed to detect and handle failures at the application layer
  - delivering a highly-available service on top of a cluster of computers, each of which may be prone to failures.

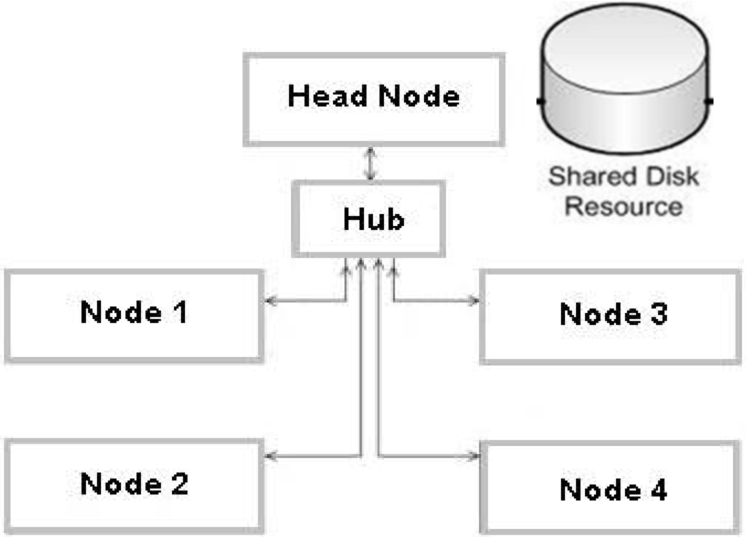
# Hadoop Modules

- Hadoop Common
  - The common utilities that support the other Hadoop modules.
- Hadoop Distributed File System (HDFS™)
  - A distributed file system that provides high-throughput access to application data.
- Hadoop YARN
  - A framework for job scheduling and cluster resource management.
- Hadoop MapReduce
  - A YARN-based system for parallel processing of large data sets.

Why?
hadoop

# Motivation - Traditional Distributed systems

- ❑ Processor Bound
- ❑ Using multiple machines
- ❑ Developer is burdened with managing too many things
  - ➢ Synchronization
  - ➢ Failures
- ❑ Data moves from shared disk to compute node
- ❑ Cost of maintaining clusters
- ❑ Scalability as and when required not present



# What we need

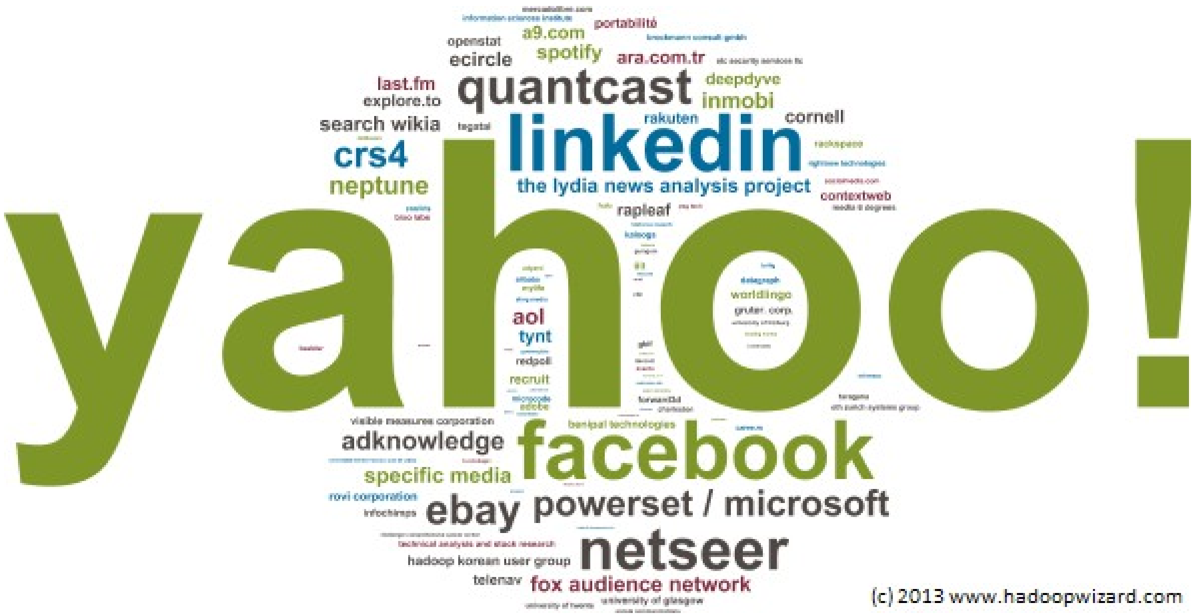
- Handling failure
  - One computer = fails once in 1000 days
  - 1000 computers = 1 per day
- Petabytes of data to be processed in parallel
  - 1 HDD= 100 MB/sec
  - 1000 HDD= 100 GB/sec
- Easy scalability
  - Relative increase/decrease of performance depending on increase/decrease of nodes

# Hadoop: Myth Vs Truth

| Myth | Truth |
|---|---|
| HDFS is a database | HDFS is a Distributed File System |
| Hadoop is a replacement of database warehouse | Compliments it, not a substitute |
| Hadoop is a complete, single product | **Ecosystem**, not just a product.<br>HDFS and MapReduce being the key components |
| Hadoop is used only for unstructured data, web analytics | Enables many types of analytics |

http://saphanatutorial.com/top-10-myths-about-hadoop/

# Who is using Hadoop



(c) 2013 www.hadoopwizard.com

Also see
https://www.dezyre.com/article/top-10-industries-using-big-data-and-121-companies-who-hire-hadoop-developers/
69

# MapReduce

# What is MapReduce?

- It is a powerful paradigm for parallel computation
- Hadoop uses MapReduce to execute jobs on files in HDFS
- Hadoop will intelligently distribute computation over cluster
- **Take computation to data**

# Origin: Functional Programming

map f [a, b, c] = [f(a), f(b), f(c)]

Returns a list constructed by applying a function (the first argument) to all items in a list passed as the second argument

Example:

```
map sq [1, 2, 3]  = [sq(1),  sq(2),  sq(3)]
                  = [1,4,9]
```

# Origin: Functional Programming

reduce f  [a, b, c]        = f(a, b, c)

OR  f(a, f(b, c))

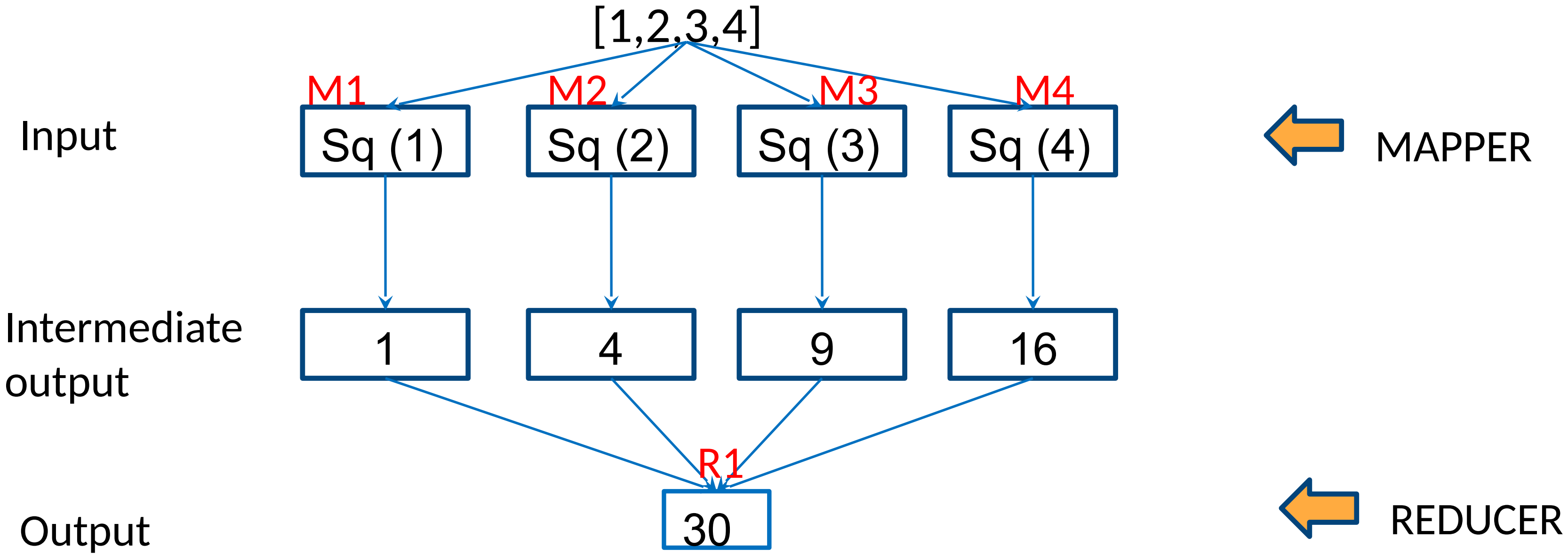
Returns a list constructed by applying a function (the first argument) on the list passed as the second argument

Example:

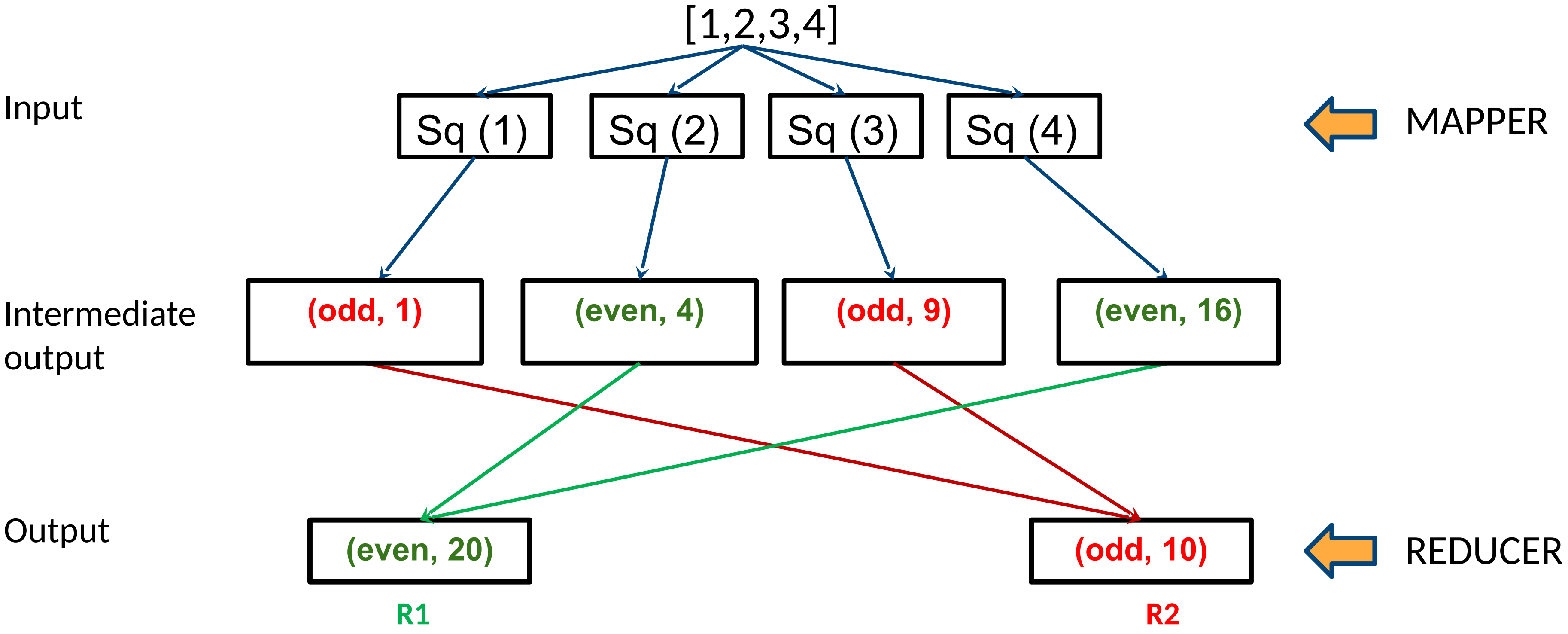
reduce sum [1, 4, 9]  = sum(1, 4, 9)

= 14

# Example: Sum of squares



# Example: Sum of squares of even and odd



# Programming model- key, value pairs

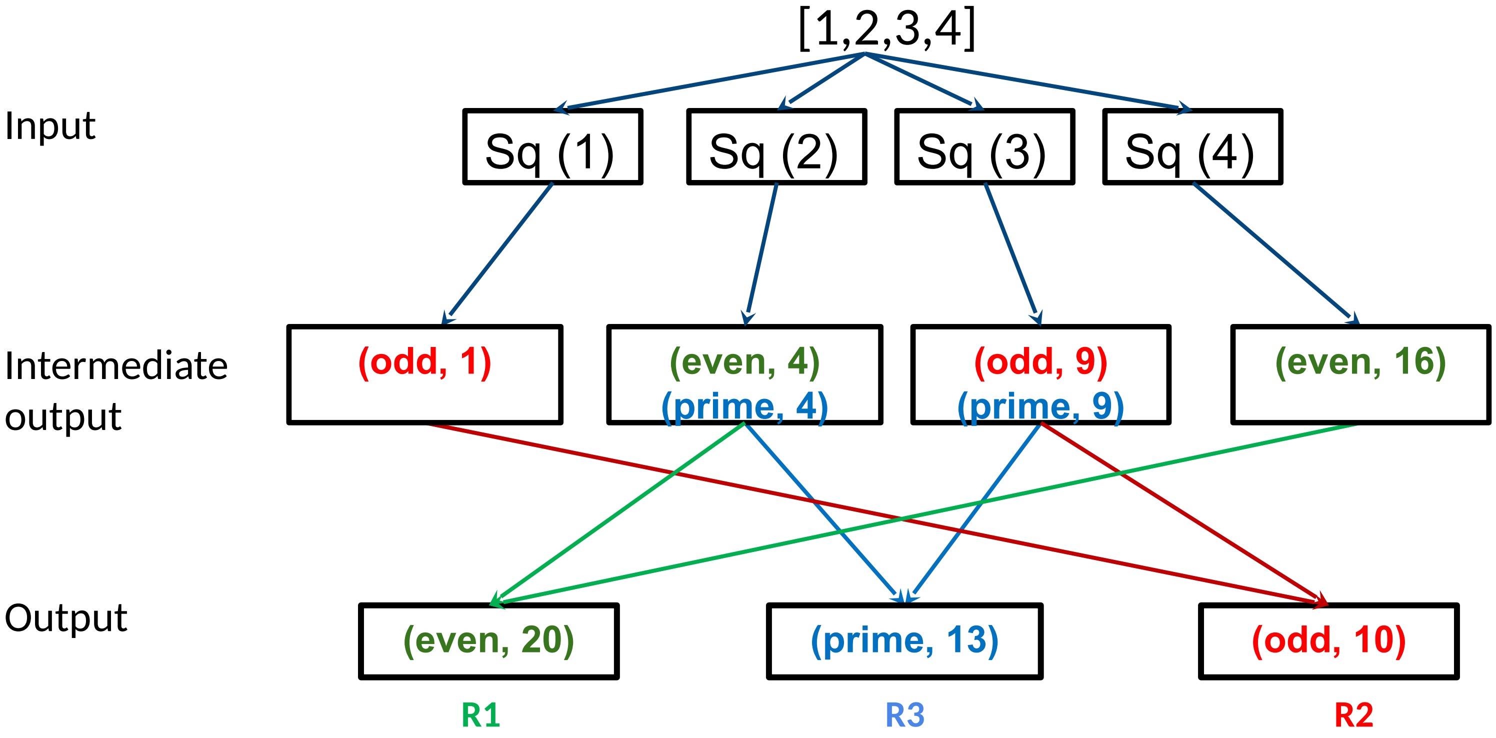
Format of input- output

(key, value)

Map: $(k_1, v_1) \rightarrow$ list $(k_2, v_2)$

Reduce: $(k_2,$ list $v_2) \rightarrow$ list $(k_3, v_3)$

# Sum of squares of even and odd and prime



# Many keys, many values

Format of input- output: (key, value)

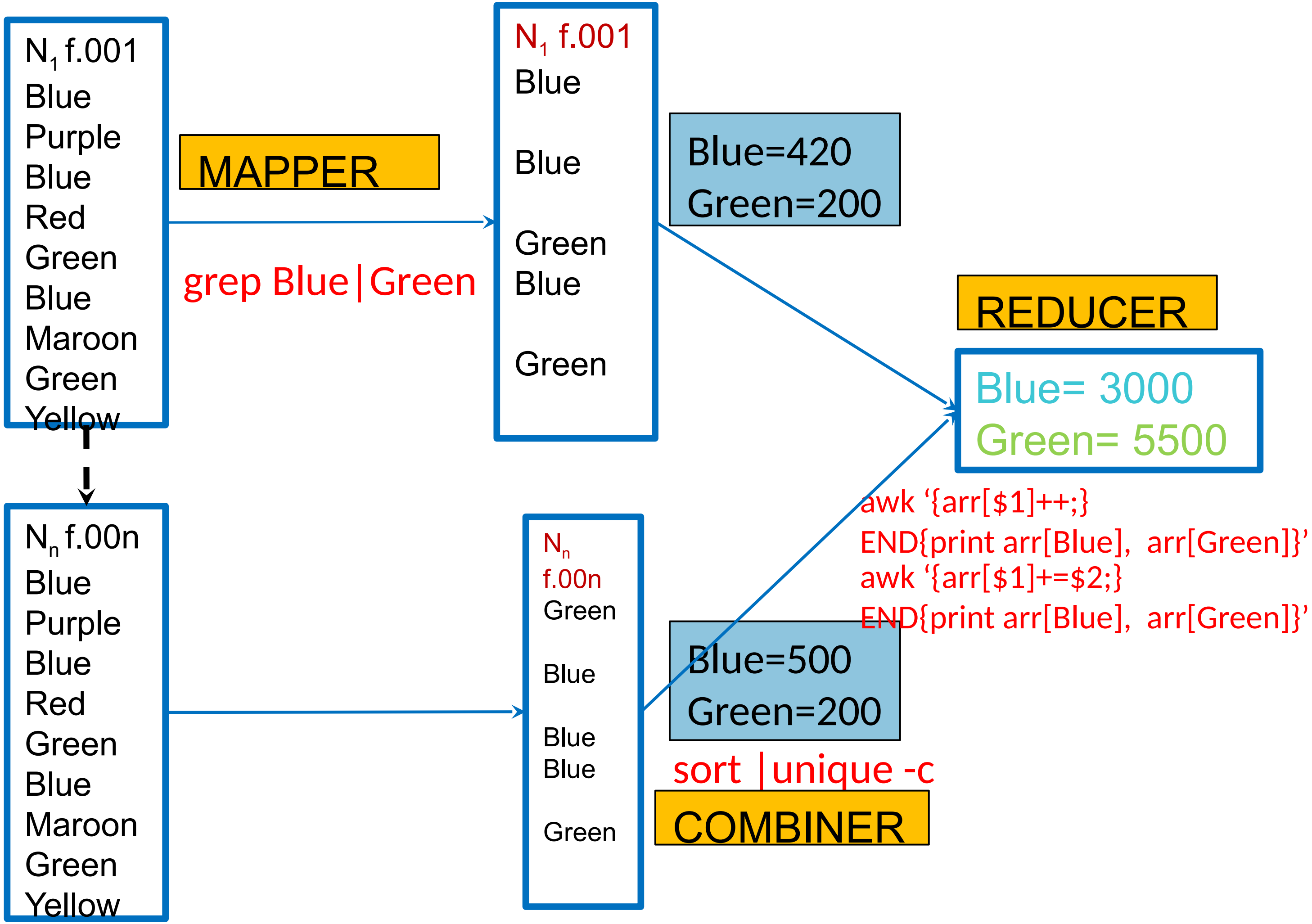
Map: $(k_1, v_1) \rightarrow$ list $(k_2, v_2)$

Reduce: $(k_2,$ list $v_2) \rightarrow$ list $(k_3, v_3)$

# Selecting Colors

Input: 1TB text file containing color names- Blue, Green, Yellow, Purple, Pink, Red, Maroon, Grey

Desired output: Occurrence of colors Blue and Green



$N_1$ f.001
Blue
Purple
Blue
Red
Green
Blue
Maroon
Green
Yellow

MAPPER

```
grep Blue|Green
```

$N_1$ f.001
Blue

Blue

Green
Blue

Green

Blue=420
Green=200

REDUCER

Blue= 3000
Green= 5500

```
awk ‘{arr[$1]++;}
END{print arr[Blue],  arr[Green]}’
awk ‘{arr[$1]+=$2;}
END{print arr[Blue],  arr[Green]}’
```

$N_n$ f.00n
Blue
Purple
Blue
Red
Green
Blue
Maroon
Green
Yellow

$N_n$
f.00n
Green

Blue

Blue
Blue

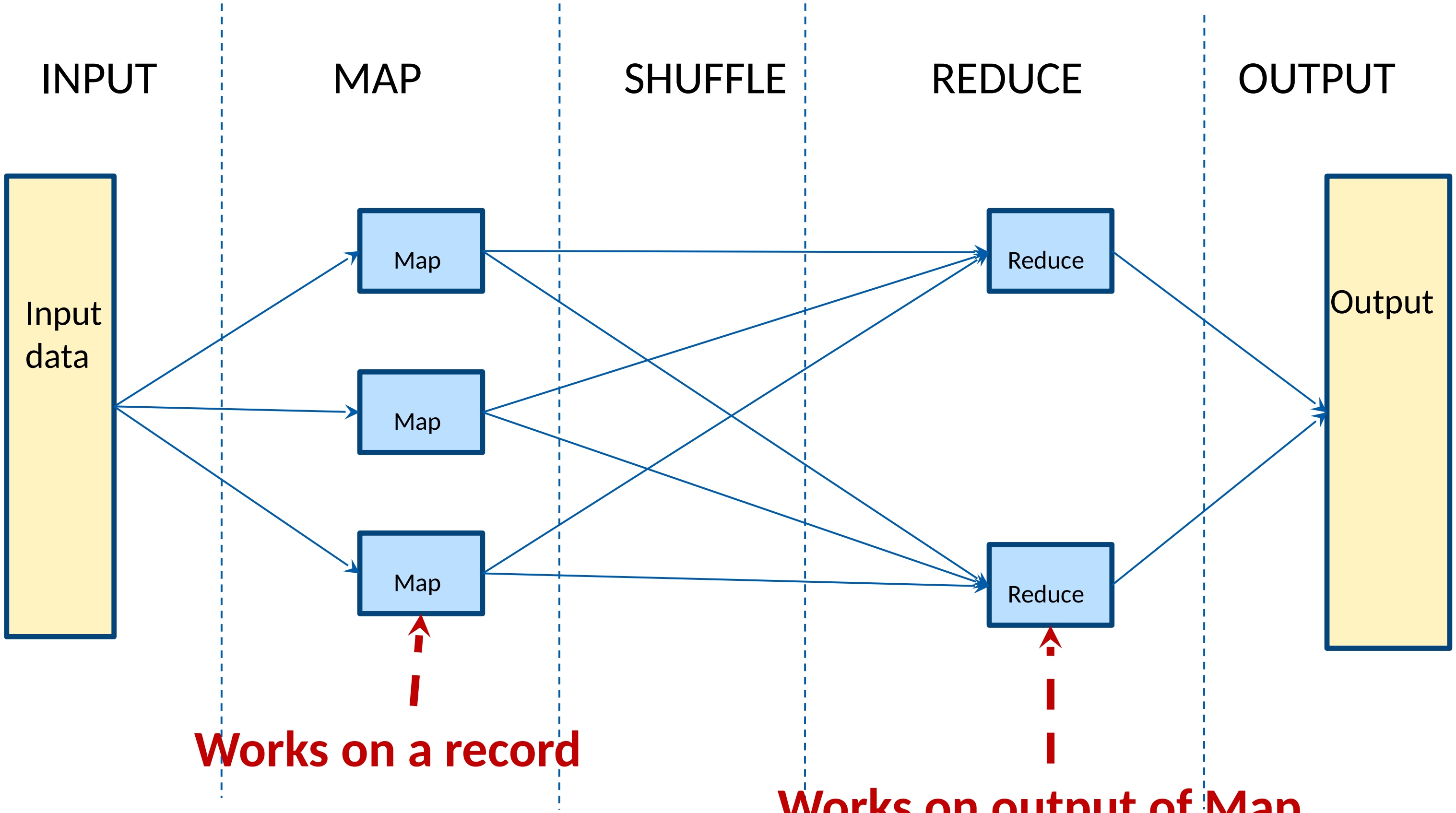
Green

Blue=500
Green=200

```
sort |unique -c
```

COMBINER

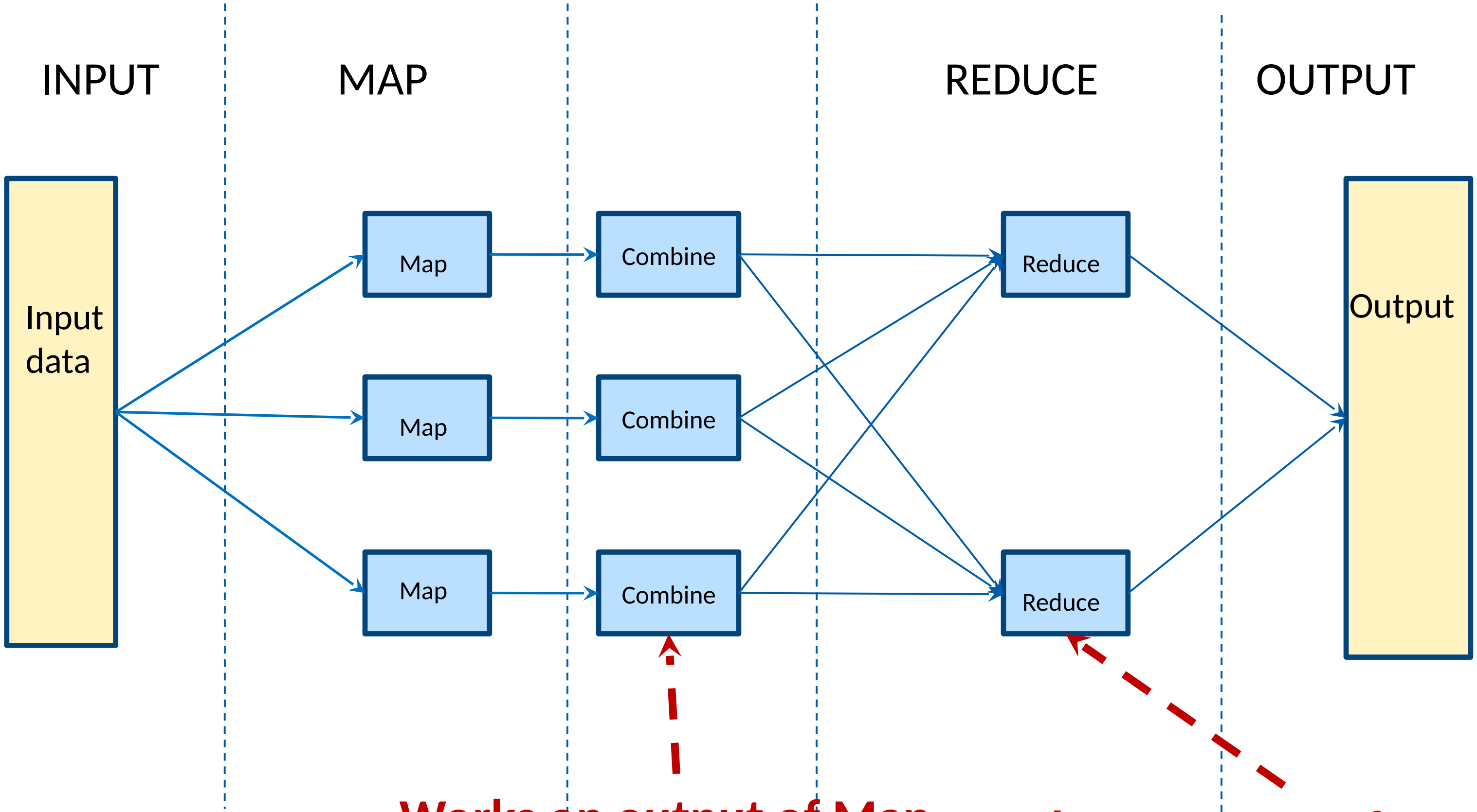
# MapReduce Overview



Works on a record

Works on output of Map

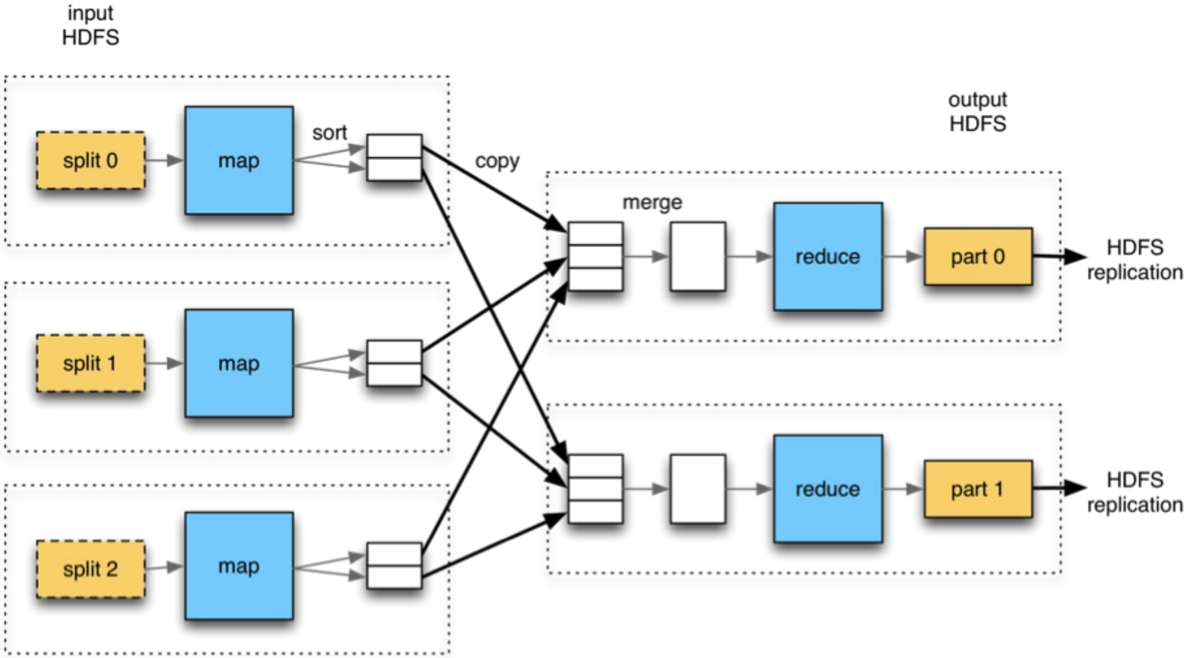
# MapReduce Overview



Works on output of Map

Works on output of Combiner

# MapReduce Overview



## Takes computation to data

# MapReduce Summary

- Mapper, Reducer and Combiner act on <key, value> pairs
- Map function gets one record at a time as an input
- Combiner (if present) works on output of map
- Reducer works on output of map (or combiner, if present)
- Combiner can be thought of local-reducer
    - Reduces output of maps that are executed on same node

# What Hadoop is not..

- Not for interactive file accessing
- Not meant for a large number of *small* files - but for a small number of *large* files
- MapReduce cannot be used for any and all applications

# Hadoop: Take Home

- **Takes computation to data**
- Suitable for large data centric operations
- Scalable on demand
- Fault tolerant and highly transparent

# **Questions**?

- ❑ Coming up next …
  - ❑ First hadoop program
  - ❑ Second hadoop program

# Your first program in hadoop (DEMO)

Open up any tutorial on hadoop and first program you see will be of wordcount

**Task**: Given a text file, generate a list of words with the number of times each of them appear in the file

**Input**: Plain text file

**Expected Output:**

- <*word, frequency*> p

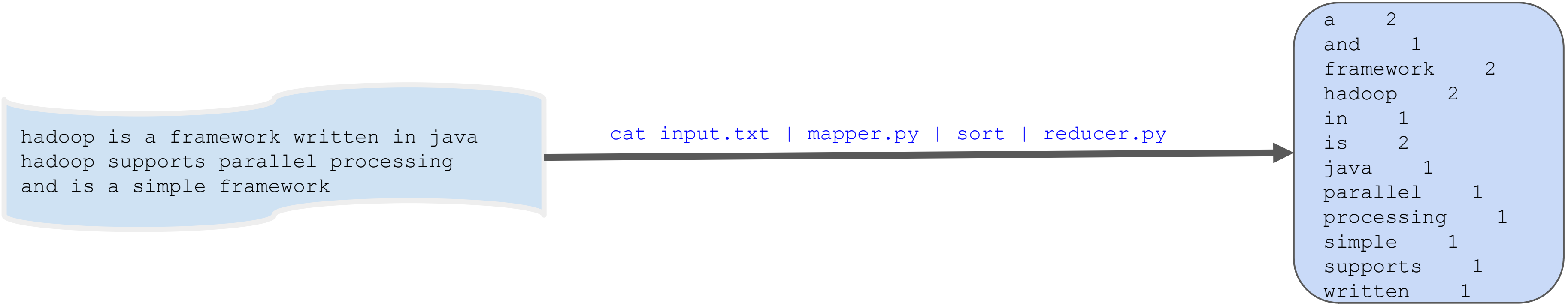
hadoop is a framework written in java
hadoop supports parallel processing
and is a simple framework

<hadoop, 2>
<is, 2>
<a , 2>
<java , 1>
<framework , 2>
<written , 1>
<in , 1>
<and,1>
<supports , 1>
<parallel , 1>
<processing. , 1>
<simple,1>

# Mimicking the Hadoop Flow

- ❑ Create files “mapper.py” for Map and “reducer.py” for Reduce
- ❑ Mimic Hadoop using the Linux pipe (|)

```
cat input.txt | mapper.py | sort | reducer.py
```



```
hadoop is a framework written in java
hadoop supports parallel processing
and is a simple framework
```

```
cat input.txt | mapper.py | sort | reducer.py
```

```
a     2
and     1
framework     2
hadoop     2
in     1
is     2
java     1
parallel     1
processing     1
simple     1
supports     1
written     1
```

# Actual Hadoop Flow

- [http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/](http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/)
  - Installation (From the above page)
  - Running Hadoop On Ubuntu Linux (Single-Node Cluster) – How to set up a *pseudo-distributed*, *single-node* Hadoop cluster backed by the Hadoop Distributed File System (HDFS)
  - Running Hadoop On Ubuntu Linux (Multi-Node Cluster) – How to set up a *distributed*, *multi-node* Hadoop cluster backed by the Hadoop Distributed File System (HDFS)
  - Minor changes needed due to changes in recent hadoop distribution directory

# Actual Hadoop Flow

: Snippets from [http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/](http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/)

- Copy input to HDFS

```
$ bin/hadoop dfs -copyFromLocal /tmp/gutenberg /user/hduser/gutenberg
```

- Run the mapper and reducer

```
$ bin/hadoop jar <path-to-jar>/hadoop-*streaming*.jar \
      -file /home/hduser/mapper.py    -mapper /home/hduser/mapper.py-file \

      /home/hduser/reducer.py   -reducer /home/hduser/reducer.py \
      -input /user/hduser/gutenberg/* -output /user/hduser/gutenberg-output
```

# Actual Hadoop Flow

: Snippets from http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

## Check the output

```
$ bin/hadoop dfs -cat /user/hduser/gutenberg-output/part-00000
"(Lo)cra"    1
"1490   1
"1498," 1
"35"1
"40,"   1
"A  2
"AS-IS".     2
"A_ 1
"Absoluti    1
[...]
hduser@ubuntu:/usr/local/hadoop$
```

# Your second program in hadoop

Task:

- Given a text file containing numbers, one per line, count sum of squares of odd, even and prime

Input:

- File containing integers, one per line

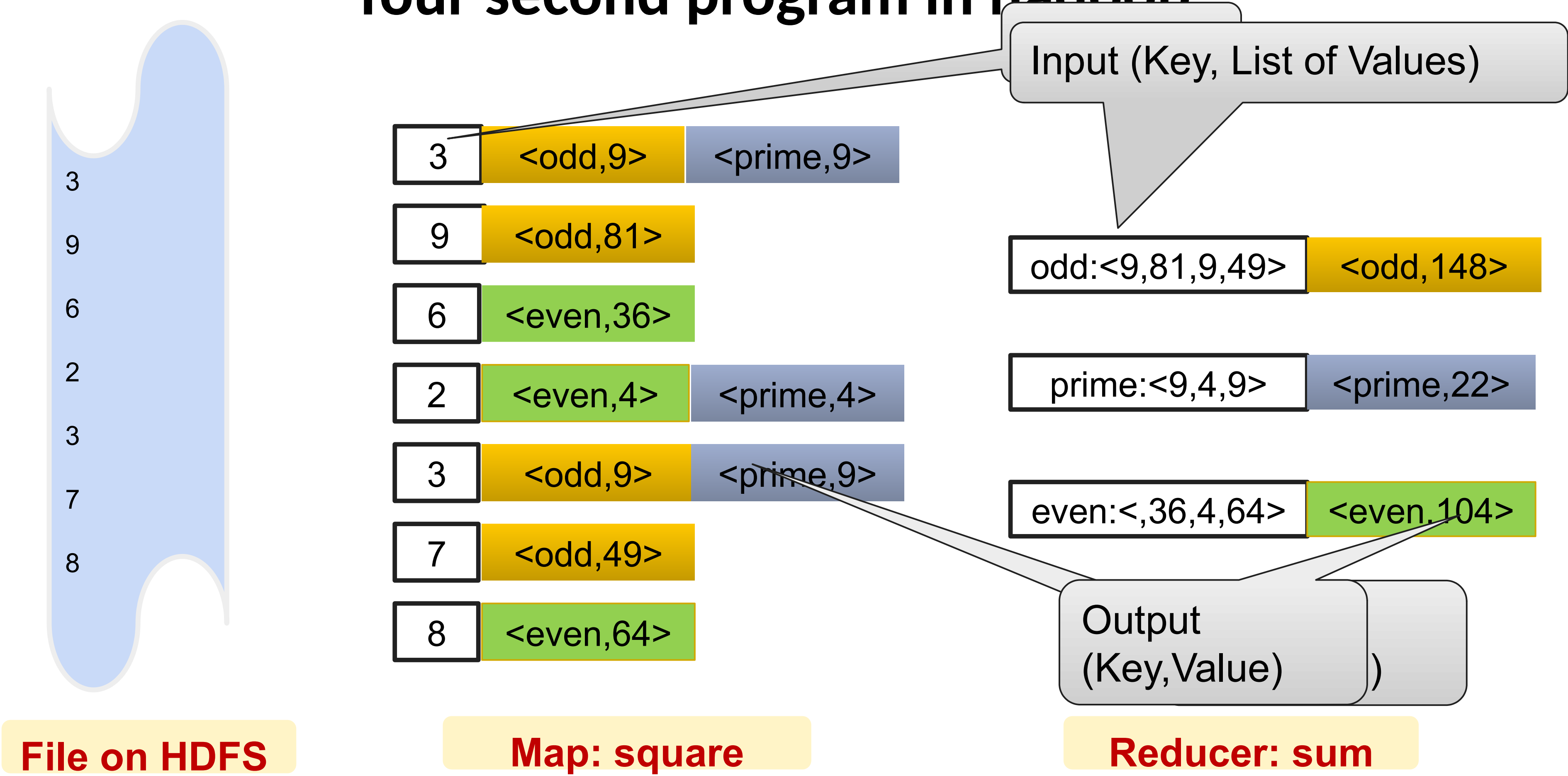
Expected Output:

- <*type, sum of squares*> for odd, even, prime

```
1
2
5
3
5
6
3
7
9
4
```

```
<odd, 302>
<even, 278>
<prime, 323 >
```

# Your second program in hadoop



3

9

6

2

3

7

8

| | | |
|---|---|---|
| 3 | <odd,9> | <prime,9> |
| 9 | <odd,81> | |
| 6 | <even,36> | |
| 2 | <even,4> | <prime,4> |
| 3 | <odd,9> | <prime,9> |
| 7 | <odd,49> | |
| 8 | <even,64> | |

Input (Key, List of Values)

| | |
|---|---|
| odd:<9,81,9,49> | <odd,148> |
| prime:<9,4,9> | <prime,22> |
| even:<,36,4,64> | <even,104> |

Output (Key,Value)

)

**File on HDFS**

**Map: square**

**Reducer: sum**

# Your second program in hadoop : Exercises

1. Mimic Hadoop Flow by writing appropriate mapper and reducer python scripts
2. Follow the tutorial to setup and run Single Node Hadoop cluster
3. Collaborate with others to setup and run Multi Node Hadoop cluster
   - ➢ Post on canvas any deviations from the steps given in the tutorial

# Hadoop Distributions

MAPR™
TECHNOLOGIES

Hortonworks

# References

- ❑ Official Hadoop website- http://hadoop.apache.org/
- ❑ Hadoop presentation wiki- http://wiki.apache.org/hadoop/HadoopPresentations?action=AttachFile
- ❑ http://developer.yahoo.com/hadoop/
- ❑ http://wiki.apache.org/hadoop/
- ❑ http://www.cloudera.com/hadoop-training/
- ❑ http://developer.yahoo.com/hadoop/tutorial/module2.html#basics

# References

# Further Reading

- ❑ Hadoop: The Definitive Guide: Tom White
- ❑ http://developer.yahoo.com/hadoop/tutorial/
- ❑ http://www.cloudera.com/content/cloudera-content/cloudera-docs/HadoopTutorial/CDH4/Hadoop-Tutorial.html

**Questions**?