Data Analytics

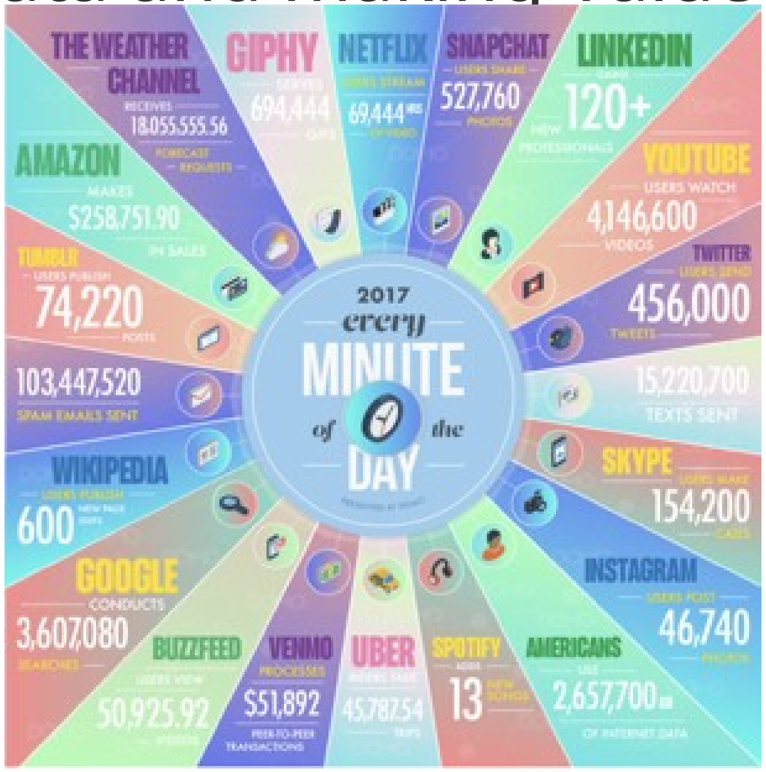
Dr. Bibhas Ghoshal
Assistant Professor
Department of Information Technology
Indian Institute of Information Technology
Allahabad

Big Data

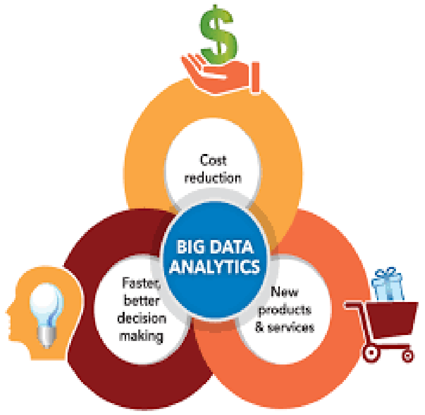
- Data that exceeds the capacity of conventional Database systems
- Too Big, Moves Fast and does not for into the structures of the Database
- Thus, we need an alternate way of processing

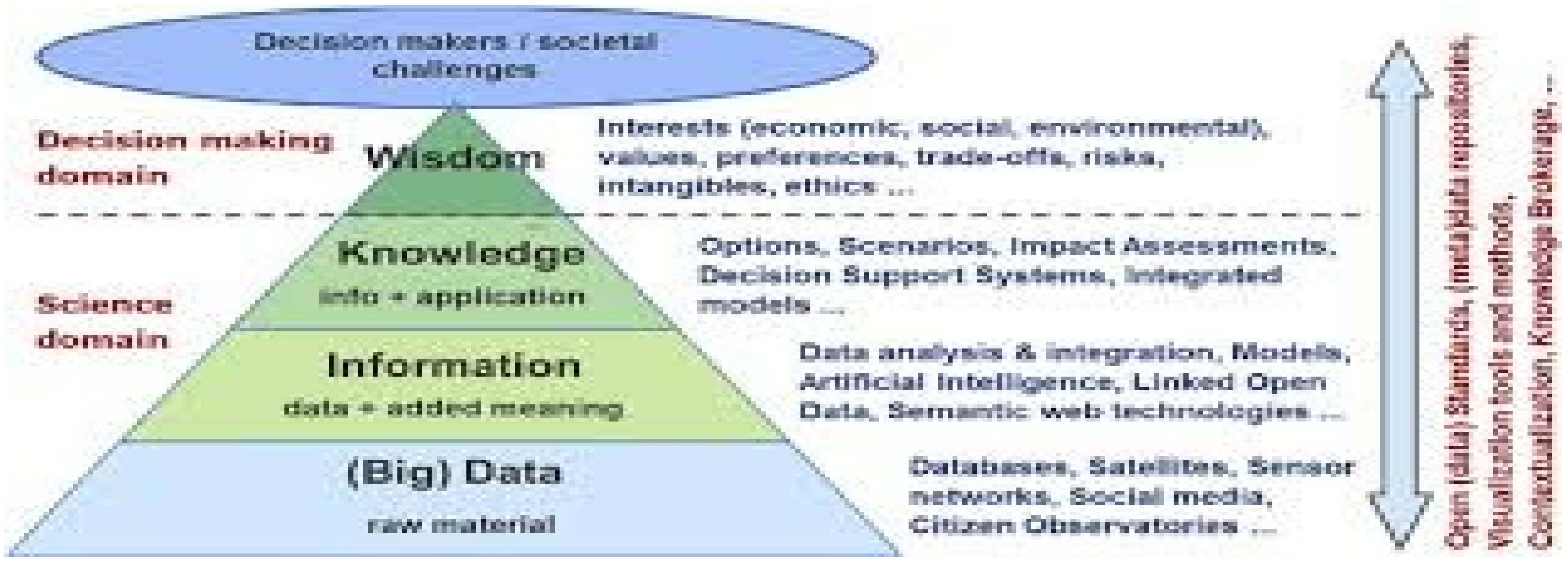
5 V's

- Volume, Velocity and Variety of Data Generated
- Data Generated by Humans, Machines, Sensors
- Veracity of data and making Value out of it



Benefits


Cost reduction
BIG DATA ANALYTICS
Faster, better decision making
New products & services


Decision makers / societal challenges
Decision making domain
Wisdom
Interests (economic, social, environmental), values, preferences, trade-offs, risks, intangibles, ethics ...
Knowledge
info + application
Options, Scenarios, Impact Assessments, Decision Support Systems, Integrated models ...
Science domain
Information
data + added meaning
Data analysis & integration, Models, Artificial Intelligence, Linked Open Data, Semantic web technologies ...
(Big) Data
raw material
Databases, Satellites, Sensor networks, Social media, Citizen Observatories ...
Open (data) Standards, (meta)data repositories,
Visualization tools and methods,
Contextualization, Knowledge Brokerage, ...

Challenges

analysis

capture

data curation

Search

transfer

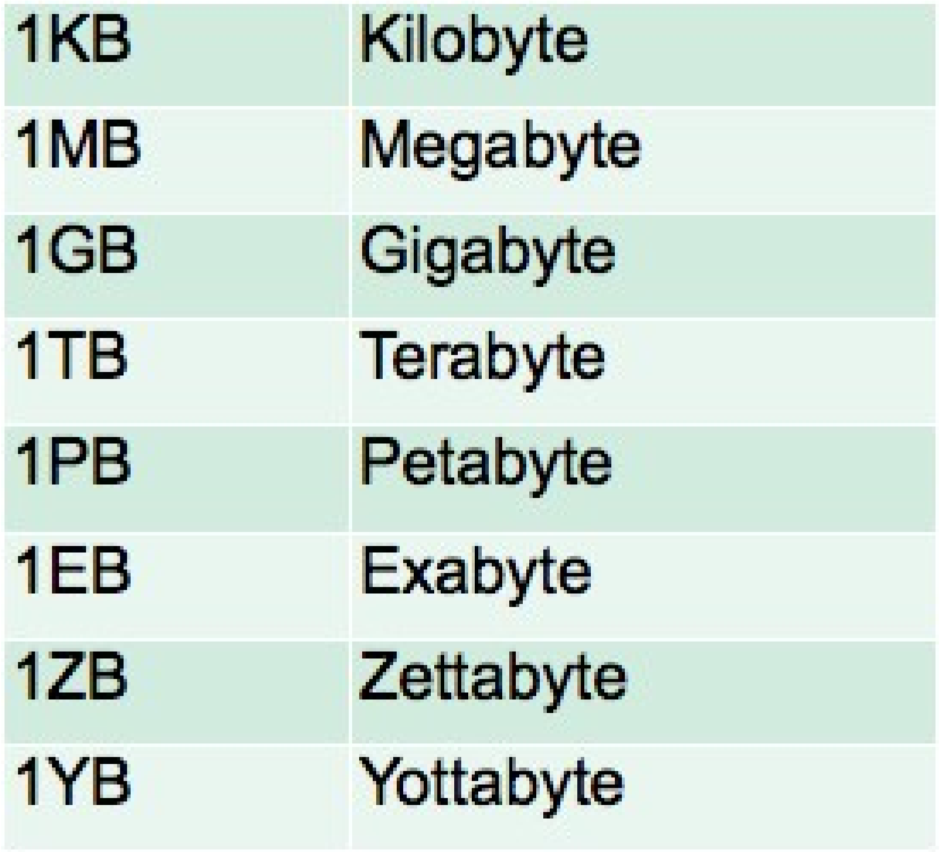
visualization

querying

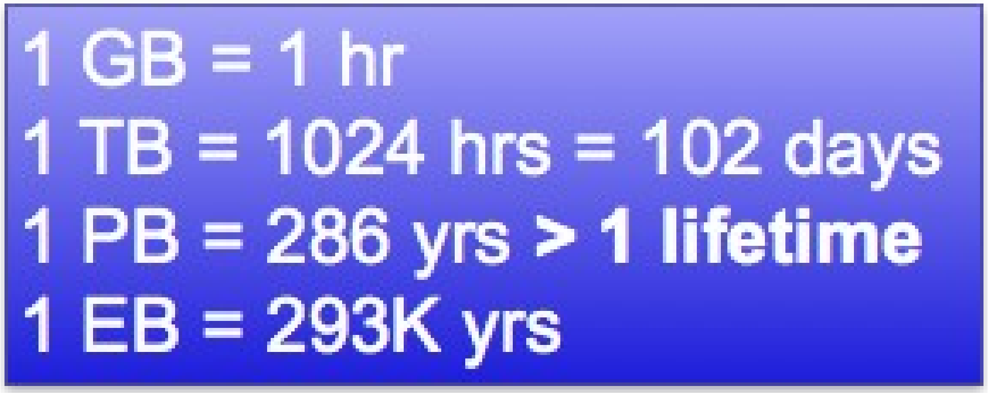
updating

information privacy.

Challenges

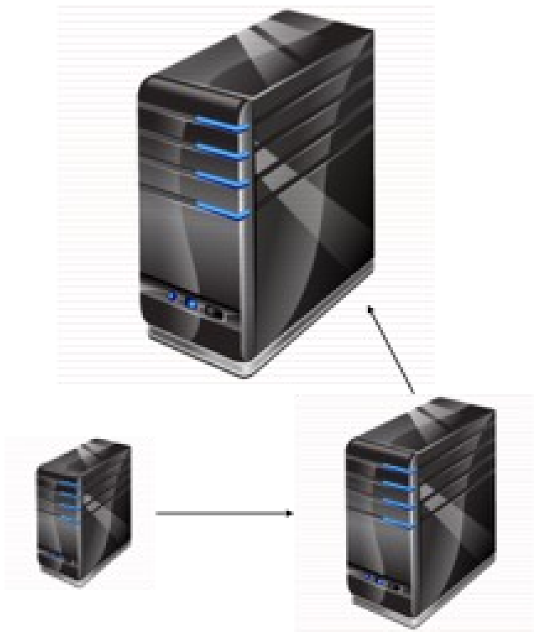


1KB	Kilobyte
1MB	Megabyte
1GB	Gigabyte
1TB	Terabyte
1PB	Petabyte
1EB	Exabyte
1ZB	Zettabyte
1YB	Yottabyte



1 GB = 1 hr
1 TB = 1024 hrs = 102 days
1 PB = 286 yrs **> 1 lifetime**
1 EB = 293K yrs

Big Data Challenges



Vertical Scaling

Horizontal Scaling

Big Data Challenges

Scale of Infrastructure

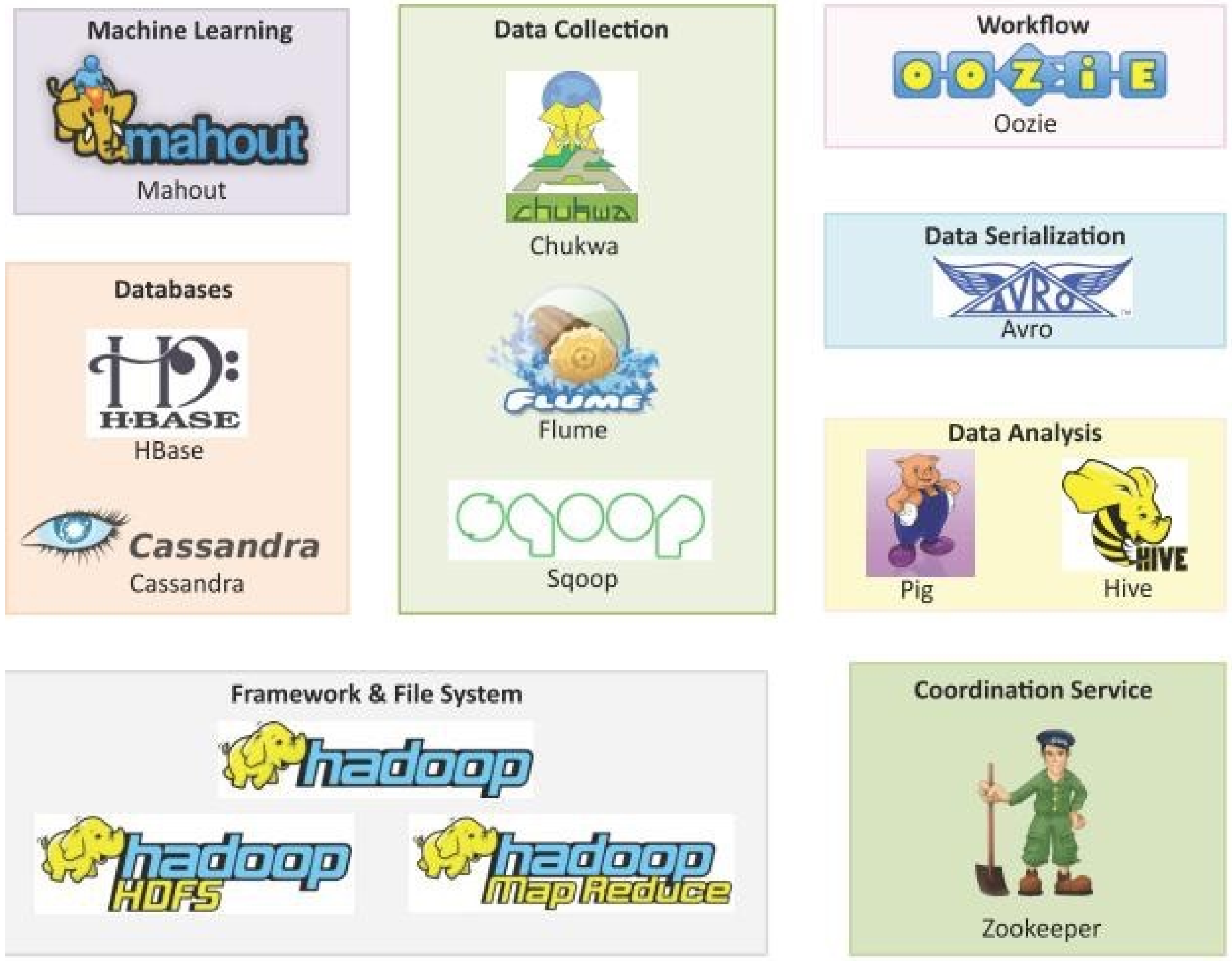
- How to Store such Big Data ?

Requirements

- Efficient Access
- Effective Utilization of Space
- Redundancy

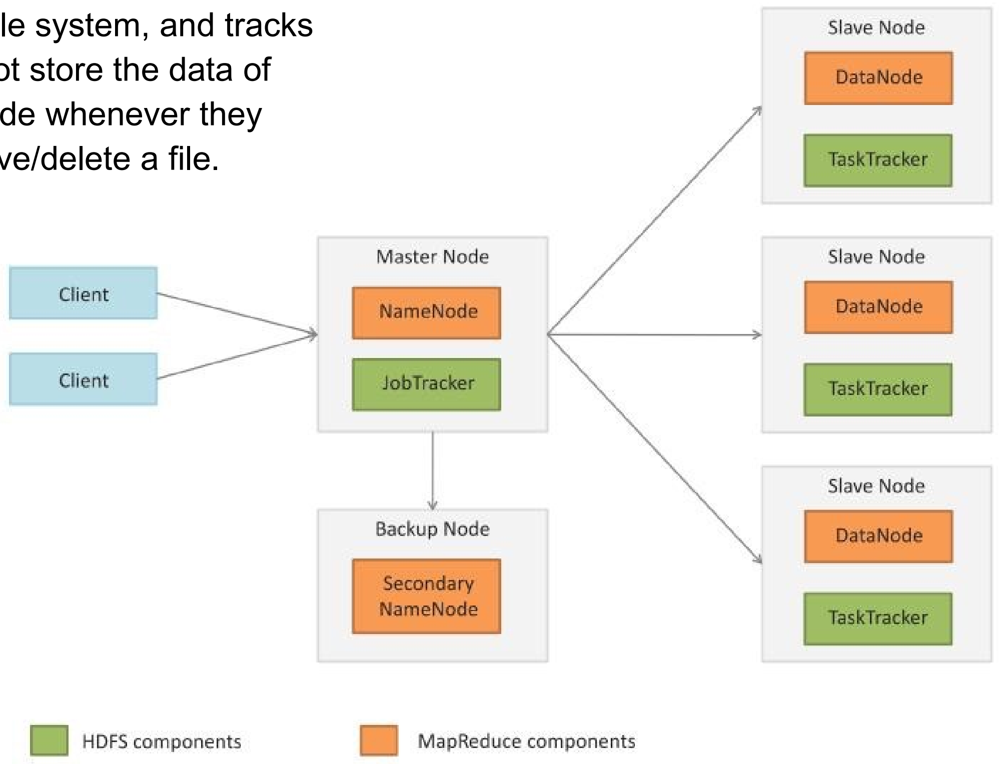
Hadoop EcoSystem

• Apache Hadoop is an open source framework for distributed batch processing of big data.



Hadoop Distributed File System

- A Hadoop cluster comprises of a Master node, backup node and a number of slave nodes.
- The master node runs the NameNode and JobTracker processes and the slave nodes run the DataNode and TaskTracker components of Hadoop.
- The backup node runs the Secondary NameNode process.
- NameNode keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. Client applications talk to the NameNode whenever they wish to locate a file, or when they want to add/copy/move/delete a file.
- NameNode is a Single Point of Failure for the HDFSCluster. An optional Secondary NameNode which is hosted on a separate machine creates checkpoints of the namespace.
- The JobTracker is the service within Hadoop that distributes MapReduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack.



• TaskTracker

• TaskTracker is a node in a Hadoop cluster that accepts Map, Reduce and Shuffie tasks from the JobTracker. Each TaskTracker has a defined number of slots which indicate the number of tasks that it can accept.

• DataNode

• A DataNode stores data in an HDFS file system.

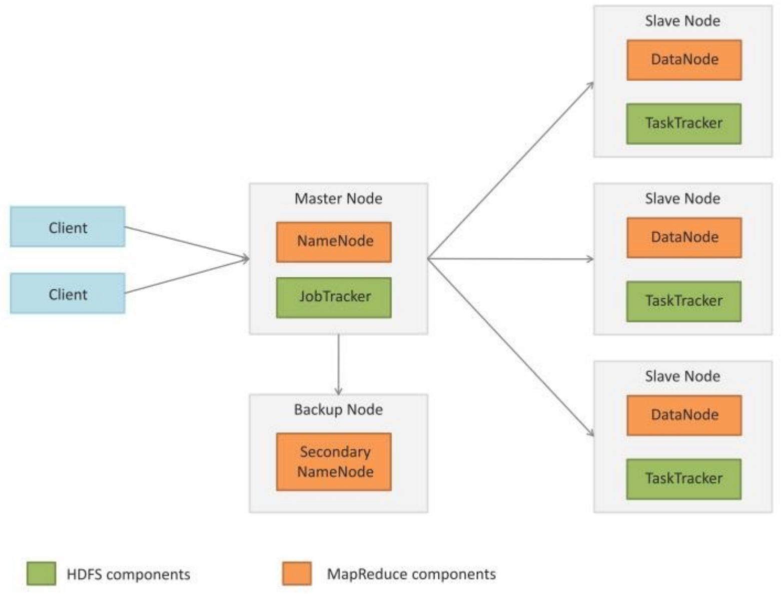
• A functional HDFS filesystem has more than one DataNode, with data replicated across them.

• DataNodes respond to requests from the NameNode for filesystem operations.

• Client applications can talk directly to a DataNode, once the NameNode has provided the location of the data.

• Similarly, MapReduce operations assigned to TaskTracker instances near a DataNode, talk directly to the DataNode to access the files.

• TaskTracker instances can be deployed on the same servers that host DataNode instances, so that MapReduce operations are performed close to the data.



Storing Files in HDFS

Motivation:

Reliability, Availability , Network Bandwidth

The input file (say 1 TB) is split into smaller chunks/blocks of 128 MB

The chunks are stored on multiple nodes as independent files on data nodes

To ensure that data is not lost, data can typically be replicated on:

local rack

remote rack (in case local rack fails)

remote node (in case local node fails)

randomly

Default replication factor is 3

Storing Files in HDFS

Default replication factor is 3

first replica of a block will be stored on a local rack

the next replica will be stored on a remote rack

the third replica will be stored on the same remote rack but on a different Datanode

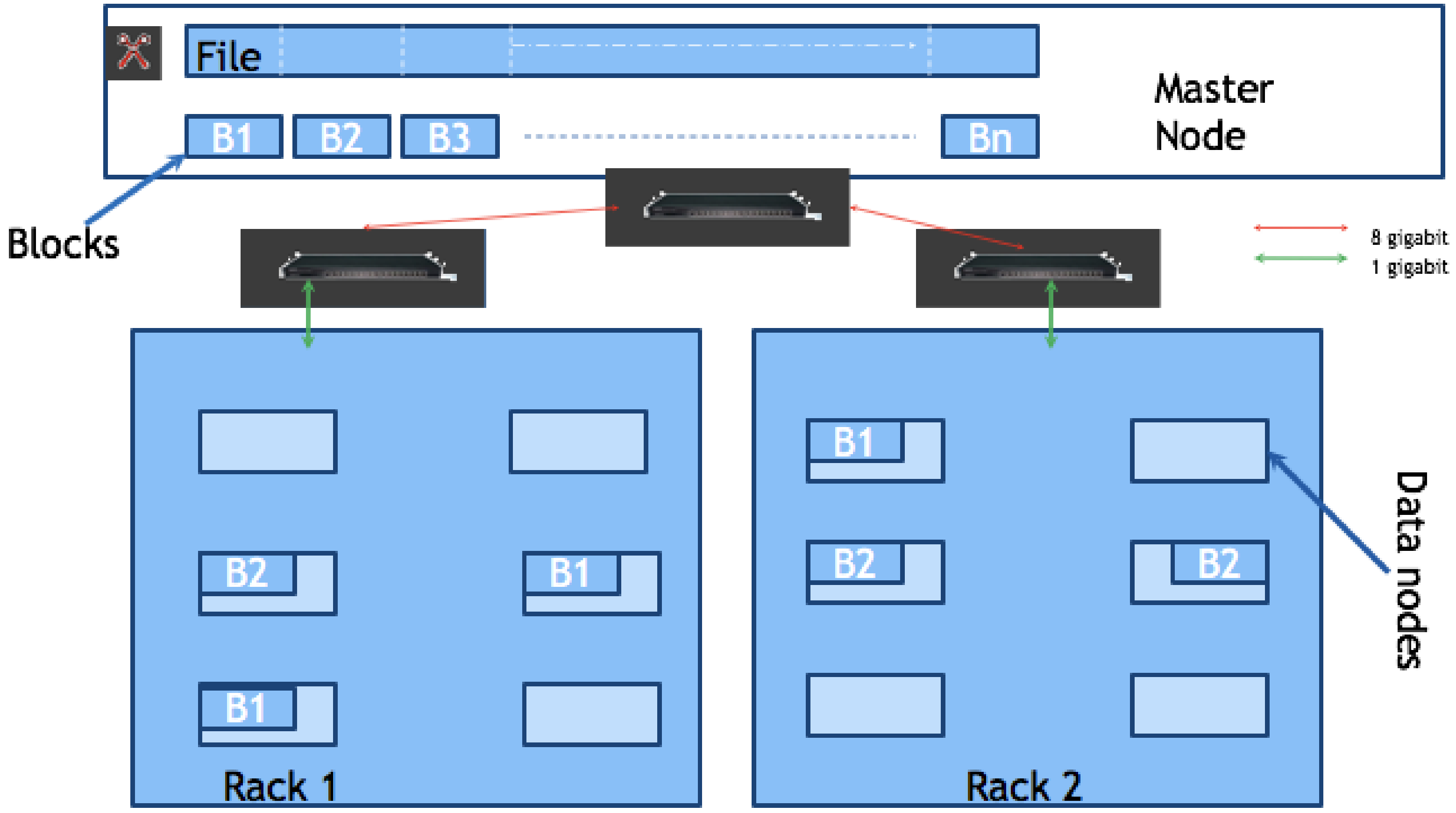
Why?

More replicas?

the rest will be placed on random Datanodes

As far as possible, no more than two replicas are kept on the same rack

• Master Node and Data Node



Tasks of NameNode

- Manages File System
- mapping files to blocks and blocks to data nodes
- Maintaining status of data nodes
- Heartbeat
  - Data node sends heartbeat at regular intervals
  - If heartbeat is not received, Data node is declared dead
- Blockreport
  - DataNode sends list of blocks on it
  - Used to check health of HDFS

NameNode Functions

- Replication
  - On Datanode failure
  - On Disk failure
  - On Block corruption
  - Data integrity
  - Checksum for each block
  - Stored in hidden file
- Rebalancing - balancer tool
  - Addition of new nodes
  - Decommissioning
  - Deletion of some files

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™)

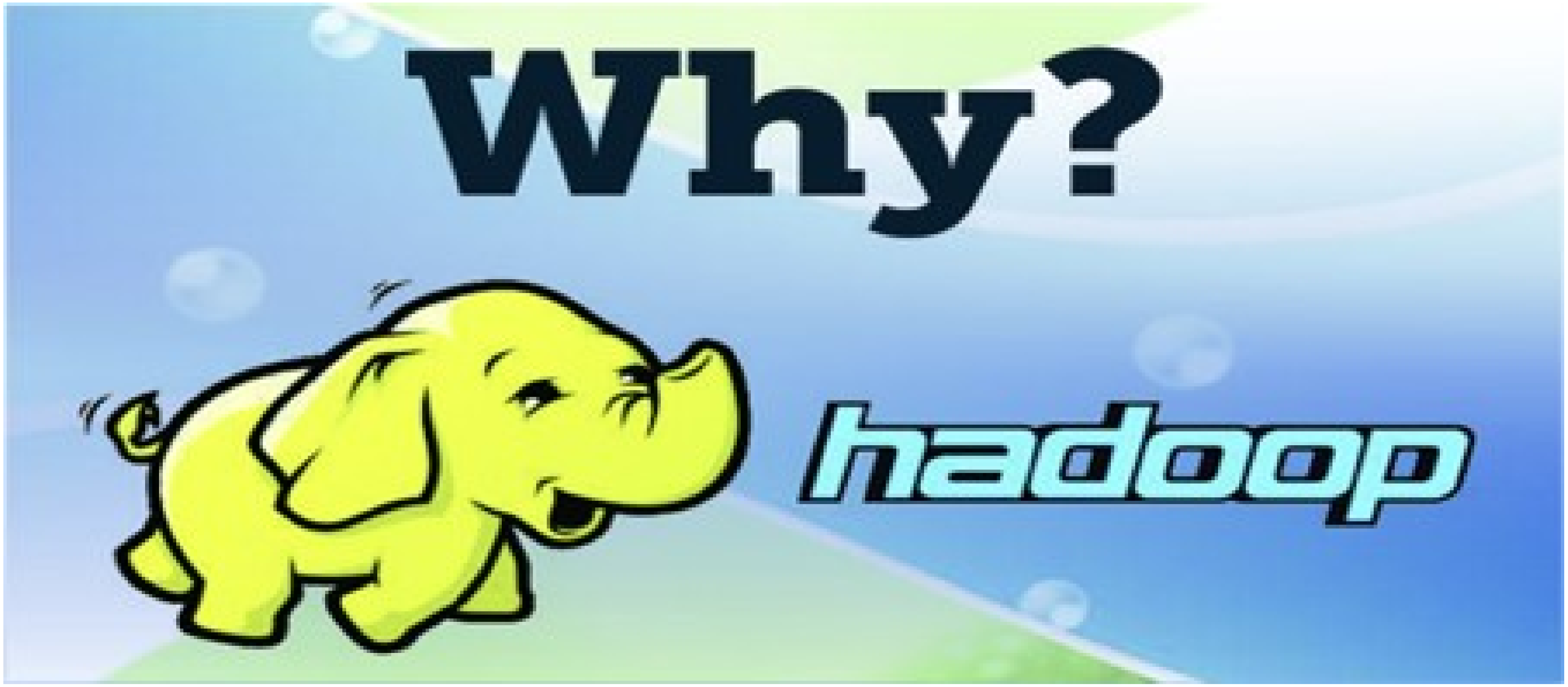
  - A distributed file system that provides high-throughput access to application data.

- 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

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. HDFS and MapReduce being the key components
Hadoop is used only for unstructured data, web analytics	Enables many types of analytics

Users of Hadoop

Map Reduce

It is a powerful paradigm for parallel computation

Hadoop uses MapReduce to execute jobs on files in HDFS

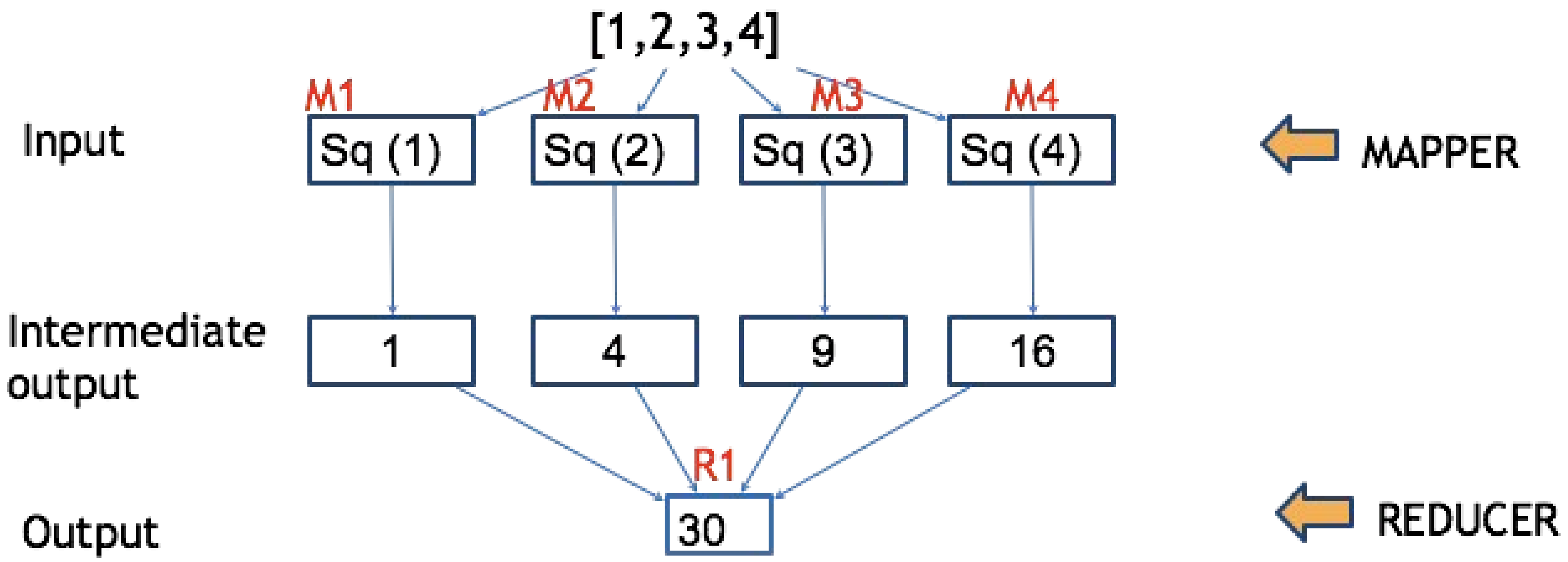
Hadoop will intelligently distribute computation over cluster

Take computation to data

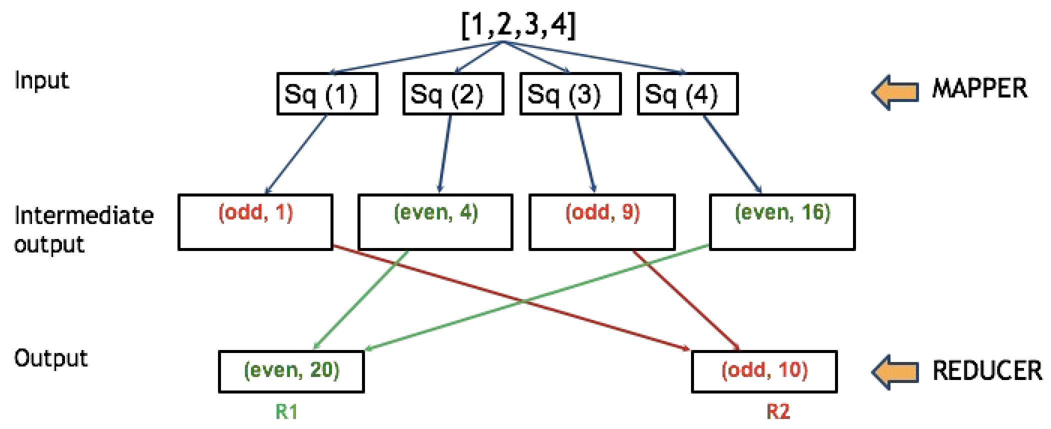
Functional Programming

- 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



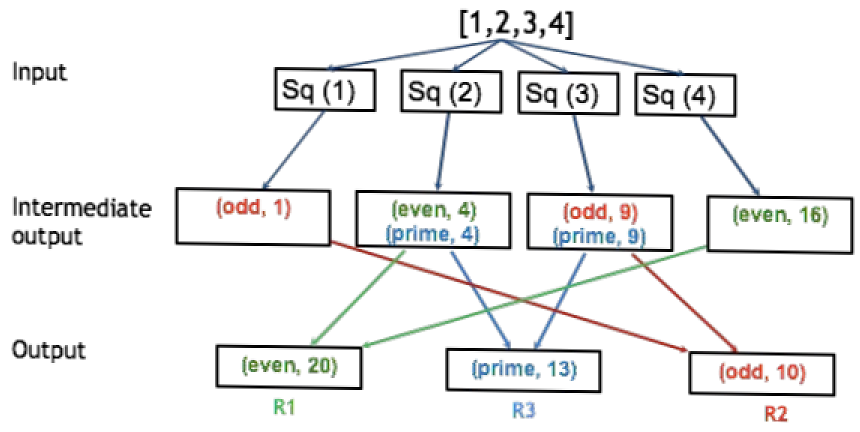
Programming model- key, value pairs

Format of input
The output (key, value)

Map: (k1, v1) → list (k2, v2)

Reduce: (k2, list v2) → list (k3, v3)

Sum of squares of even and odd and prime



Many keys, many values

Format of input
output: (key, value)

Map: (k1, v1) → list (k2, v2)
Reduce: (k2, list v2) → list (k3, v3)

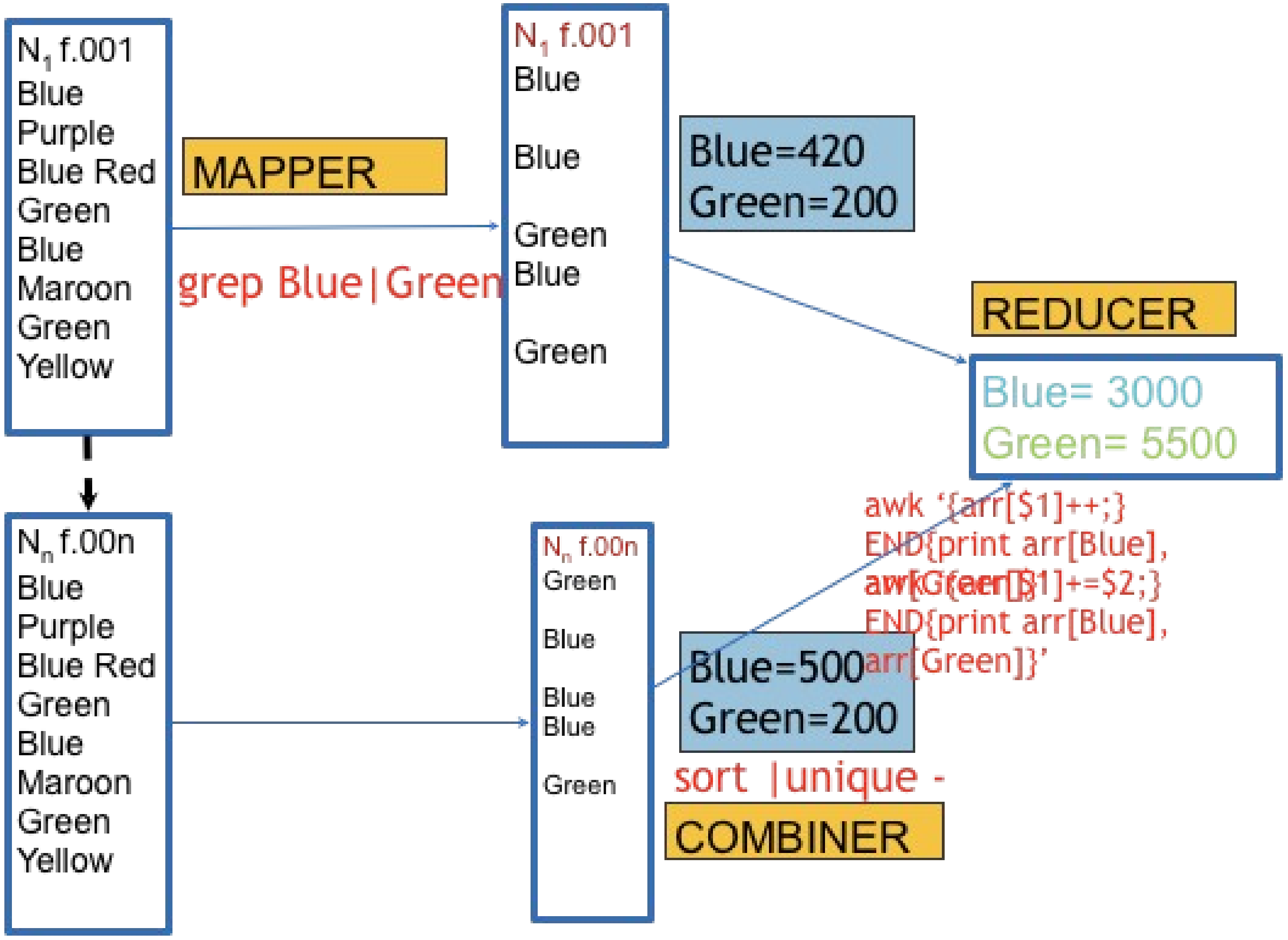
Selecting Colors

Input :

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

Output :

Occurrence of colours Blue and Green



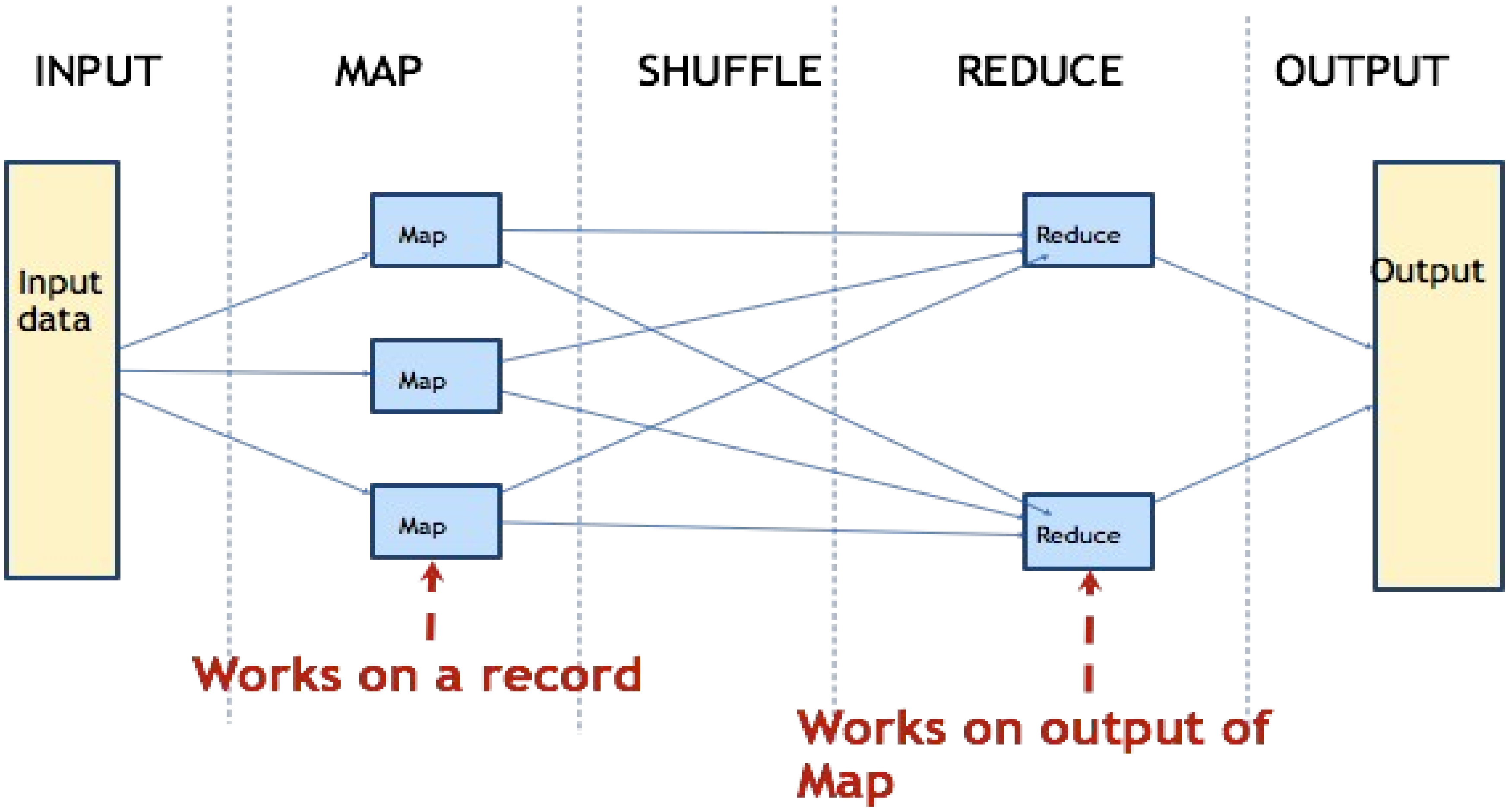
```
grep Blue|Green
```

```
awk '{arr[$1]++;}
END{print arr[Blue],
arr[Green]}'
```

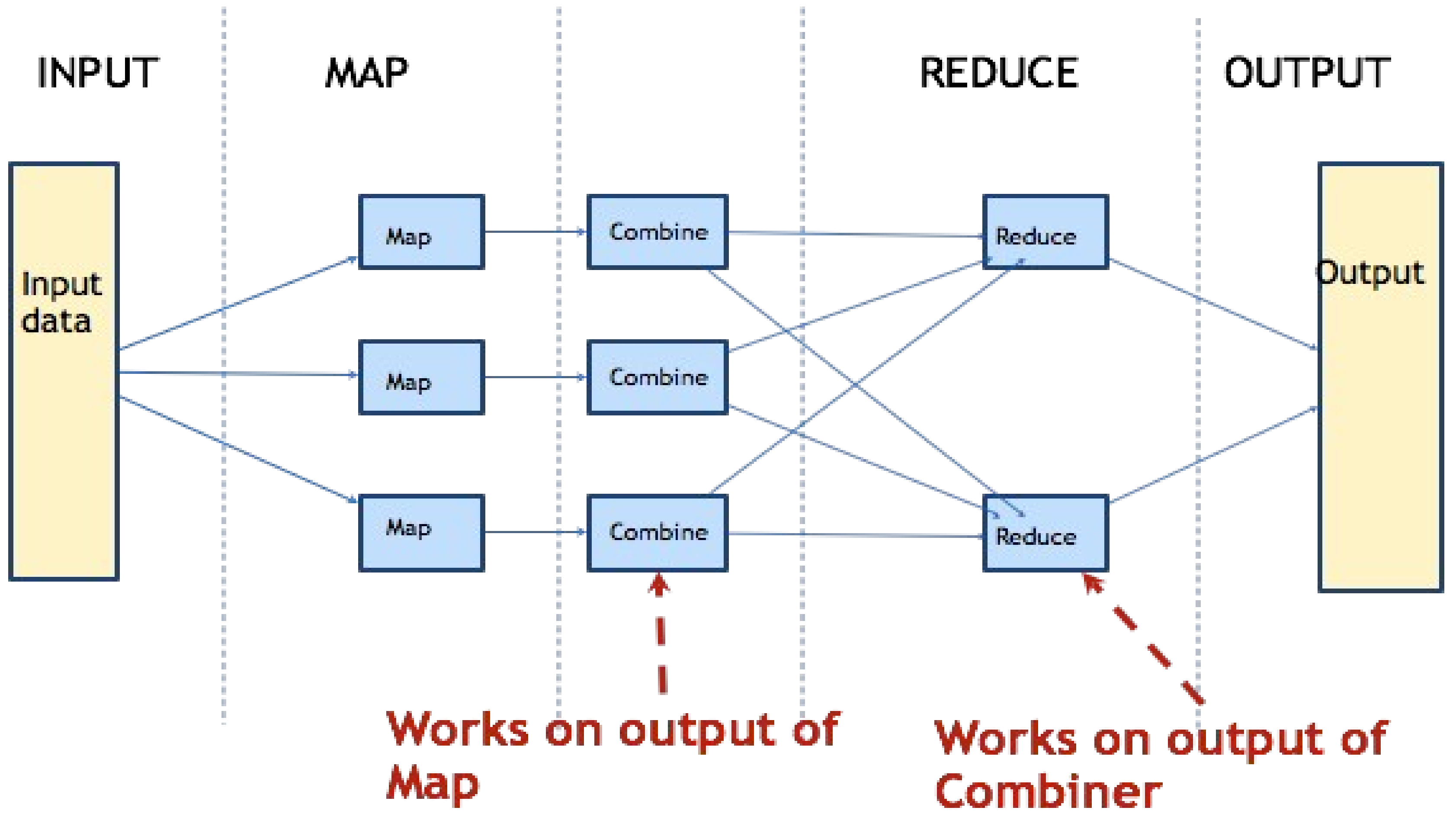
```
awk '{arr[$1]+=$2;}
END{print arr[Blue],
arr[Green]}'
```

```
sort |unique -
```

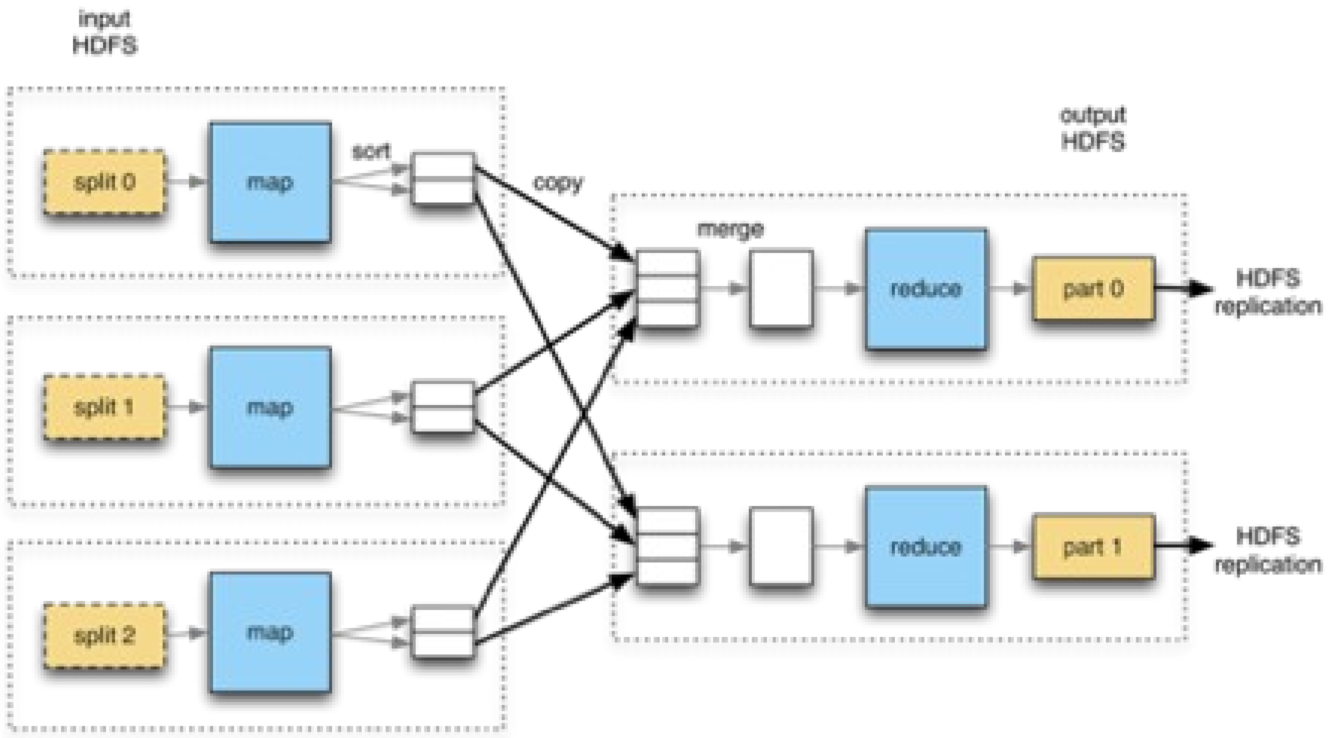
MapReduce Overview



Map Reduce Overview



Map reduce Overview



Map Reduce 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

MapReduce Job Execution Workflow

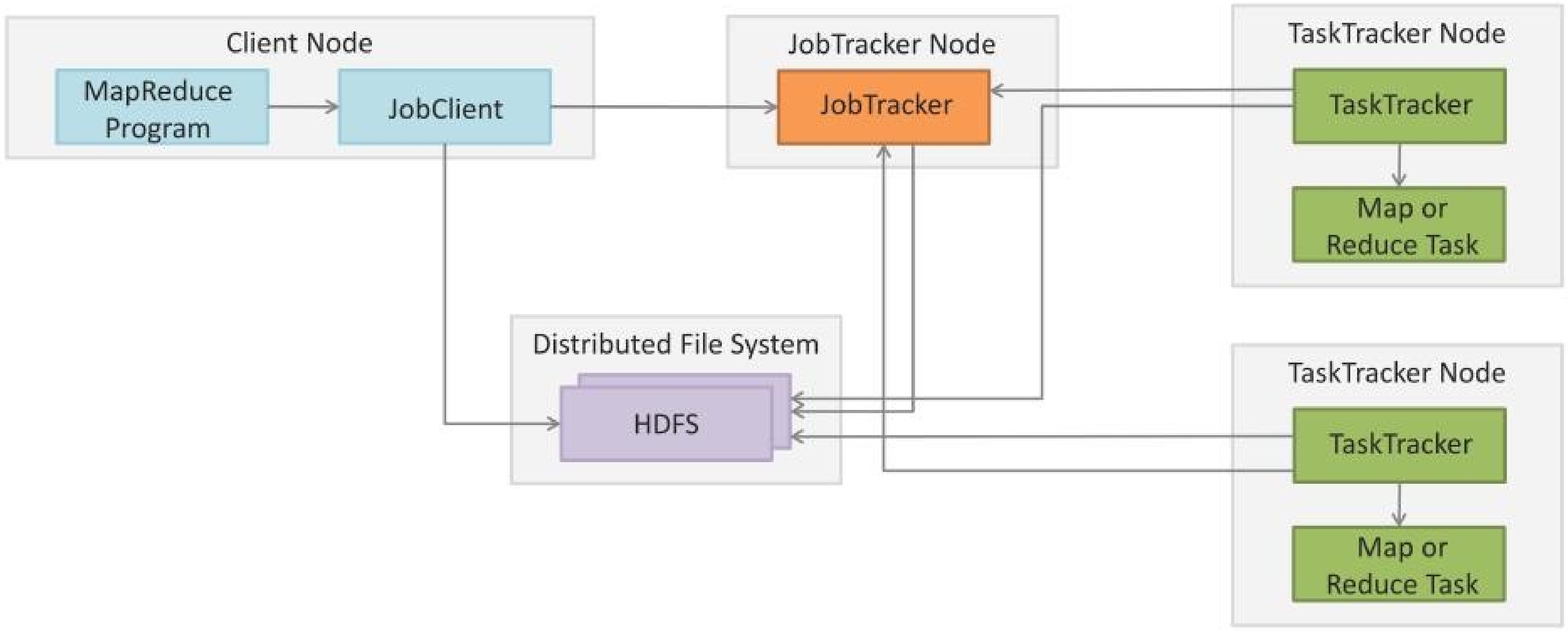
• MapReduce job execution starts when the client applications submit jobs to the Job tracker.

• The JobTracker returns a JobID to the client application. The JobTracker talks to the NameNode to determine
the location of the data.

• The JobTracker locates TaskTracker nodes with available slots at/or near the data.

• The TaskTrackers send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the
JobTracker that they are still alive. These messages also inform the JobTracker of the number of available
slots, so the JobTracker can stay up to date with where in the cluster, new work can be delegated.

Map Reduce Execution Flow



MapReduce Job Execution Workflow

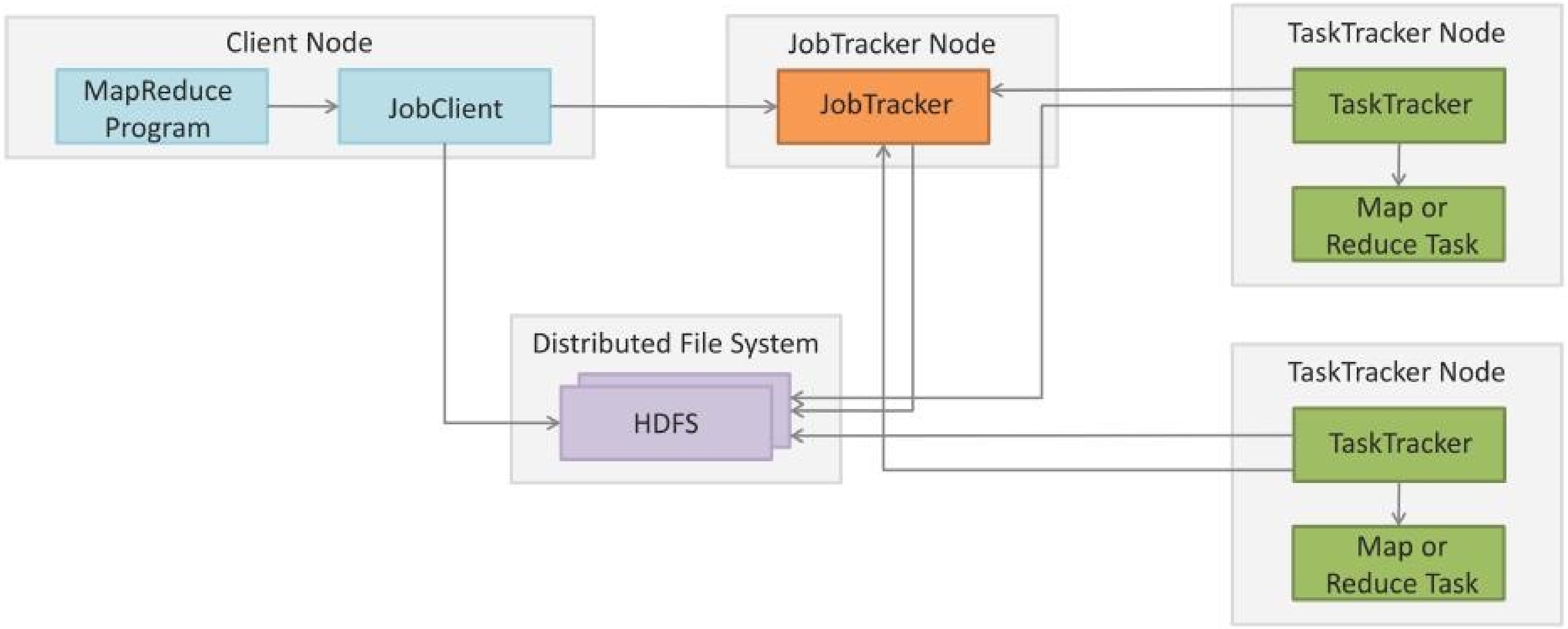
• The JobTracker submits the work to the TaskTracker nodes when they poll for tasks. To choose a task for a

TaskTracker, the JobTracker uses various scheduling algorithms (default is FIFO).

• The TaskTracker nodes are monitored using the heartbeat signals that are sent by the TaskTrackers to
JobTracker.

• The TaskTracker spawns a separate JVM process for each task so that any task failure does not bring down
the TaskTracker.

• The TaskTracker monitors these spawned processes while capturing the output and exit codes. When the
process finishes, successfully or not, the TaskTracker notifies the JobTracker. When the job is completed, the
JobTracker updates its status.


Client Node
MapReduce Program
JobClient
JobTracker Node
JobTracker
TaskTracker Node
TaskTracker
Map or Reduce Task
Distributed File System
HDFS
TaskTracker Node
TaskTracker
Map or Reduce Task

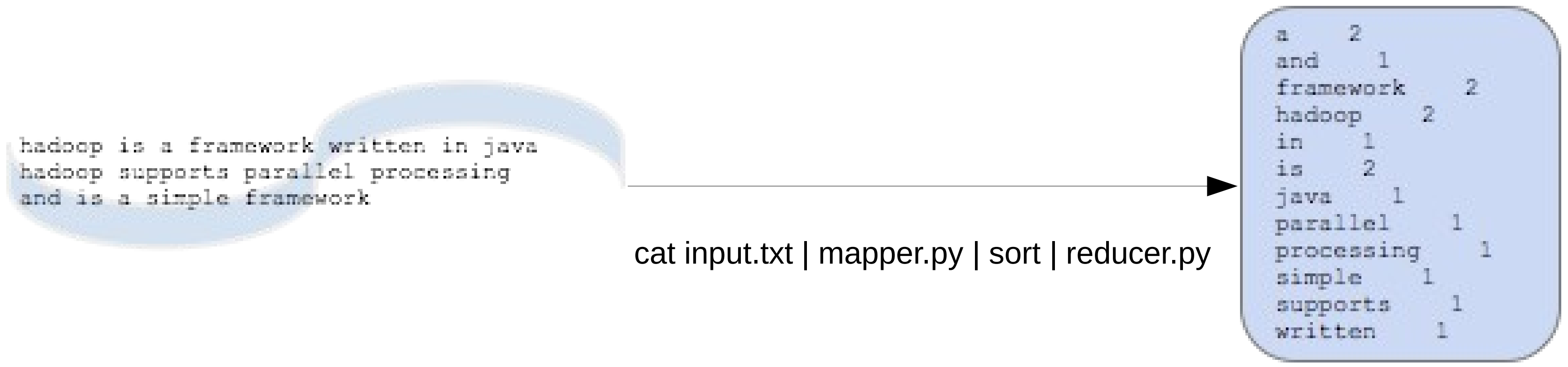
Map Reduce Demo

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> pairs for all words in the file

- 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
```

```
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/

Installation (From the above page)

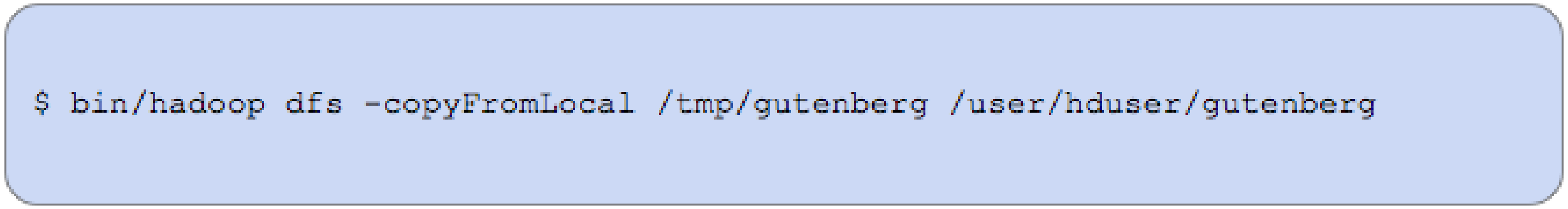
Running Hadoop On Ubuntu Linux (Single-Node Cluster) – How to set up a *pseudo-distributed*, *single-node* Hadoop cluster backed by the Hadoop Distributed File System (HDFS)

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

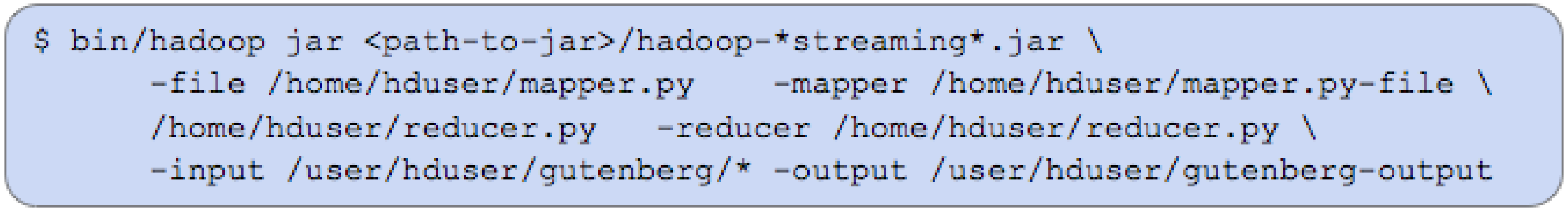
Actual Hadoop Flow : Snippets from http://www.michael-noll.com/tutorials/writing-an-hadoop-mapreduce-program-in-python/

- Copy input to HDFS
-



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

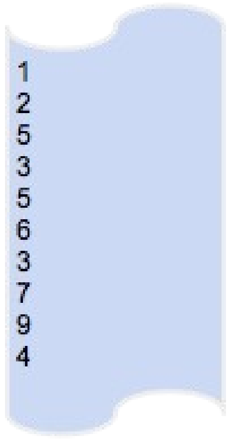
Run the mapper and reducer



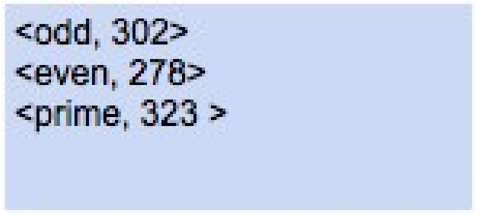
```
$ bin/hadoop jar <path-to-jar>/hadoop-*streaming*.jar \
      -file /home/hduser/mapper.py    -mapper /home/hduser/mapper.py-file \
      /home/hduser/reducer.py    -reducer /home/hduser/reducer.py \
      -input /user/hduser/gutenberg/* -output /user/hduser/gutenberg-output
```

Another program in hadoop

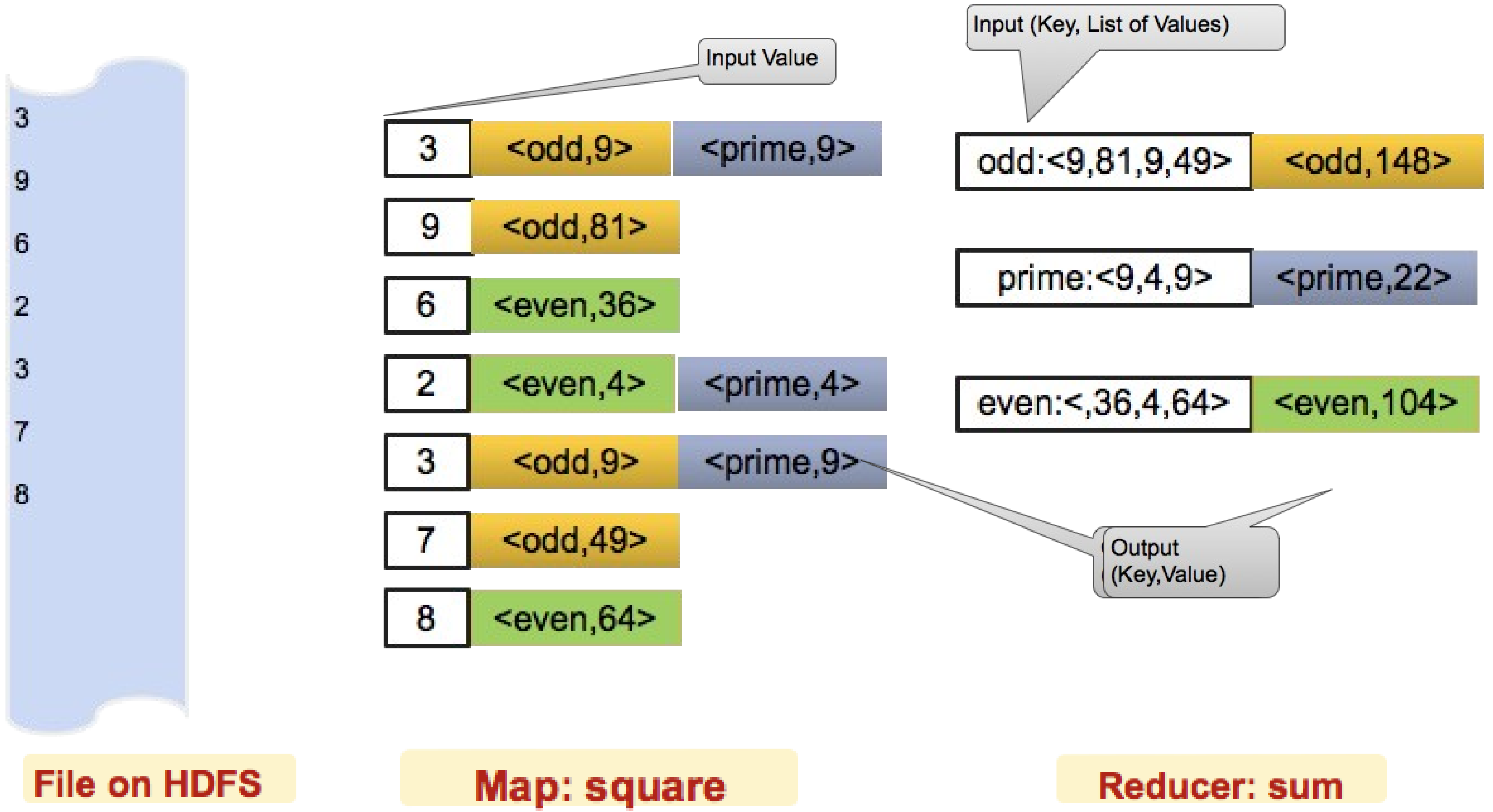
- Task:
- Given a text file containing numbers, one per line, count sum of squares of odd, even and prime
- Input:
- File containing integers, one per line
- Expected Output:
- <type, sum of squares> for odd, even, prime



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



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


Input Value
Input (Key, List of Values)
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>
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