CSCE 5300 Introduction to Big Data and Data Science

Zeenat Tariq, Ph.D.

Hadoop MapReduce and Hadoop Distributed File System (HDFS)

Overview

- Hadoop
- Hadoop MapReduce
- Hadoop Distributed File System (HDFS)
- Use case – Wordcount
- ICE – Counting frequency of words in the given input using MapReduce paradigm
  - Counting frequency of words starting with ‘a’

Hadoop

- Apache Hadoop is a software framework that supports data-intensive distributed applications under a free license
- It enables applications to work with thousands of computational independent computers and petabytes of data
- Open Source Implementation of MapReduce by Apache Software Foundation.
- Created by Doug Cutting.
- Derived from Google's MapReduce and Google File System (GFS) papers.

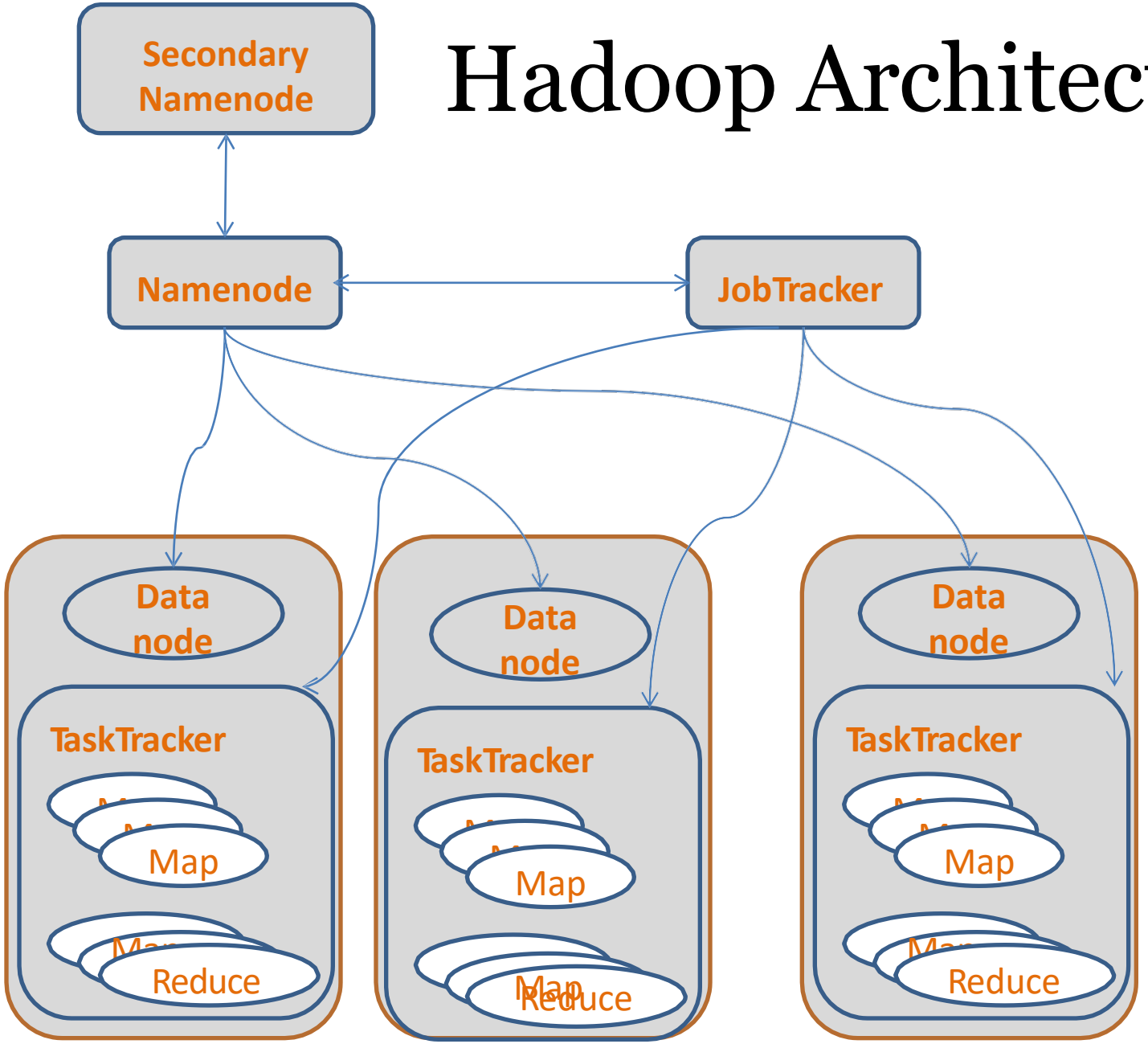
Source: http://hadoop.apache.org/ and https://www.slideshare.net/

Hadoop Architecture

- Hadoop MapReduce
  - Single master node, many worker nodes
  - Client submits a *job* to master node
  - Master splits each job into *tasks* (MapReduce), and assigns tasks to worker nodes
- Hadoop Distributed File System (HDFS)
  - Single name node, many data nodes
  - Files stored as large, fixed-size (e.g. 64MB) blocks
  - HDFS typically holds map input and reduce output

Source: https://www.slideshare.net/

Hadoop Architecture



Source: https://www.slideshare.net/

Job Scheduling in Hadoop

- One map task for each block of the input file
  - Applies user-defined map function to each record in the block
  - Record = <key, value>
- User-defined number of reduce tasks
  - Each reduce task is assigned a set of record groups
  - For each group, apply user-defined reduce function to the record values in that group
- Reduce tasks read from *every* map task
  - Each read returns the record groups for that reduce task

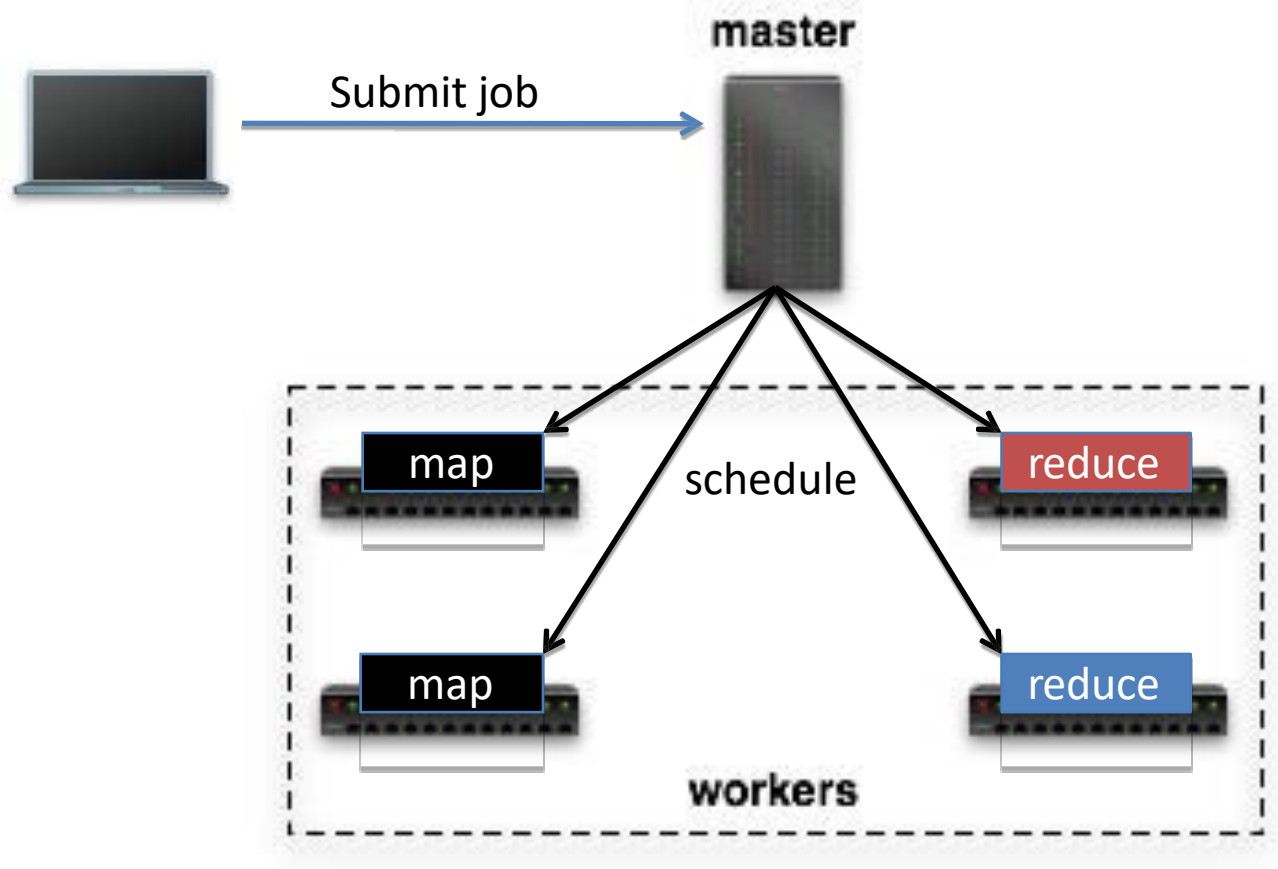
Source: https://www.slideshare.net/

Dataflow in Hadoop

- Map tasks write their output to local disk
  - Output available after map task has completed
- Reduce tasks write their output to HDFS
  - Once job is finished, next job's map tasks can be scheduled, and will read input from HDFS
- Therefore, fault tolerance is simple: simply re- run tasks on failure
  - No consumers see partial operator output

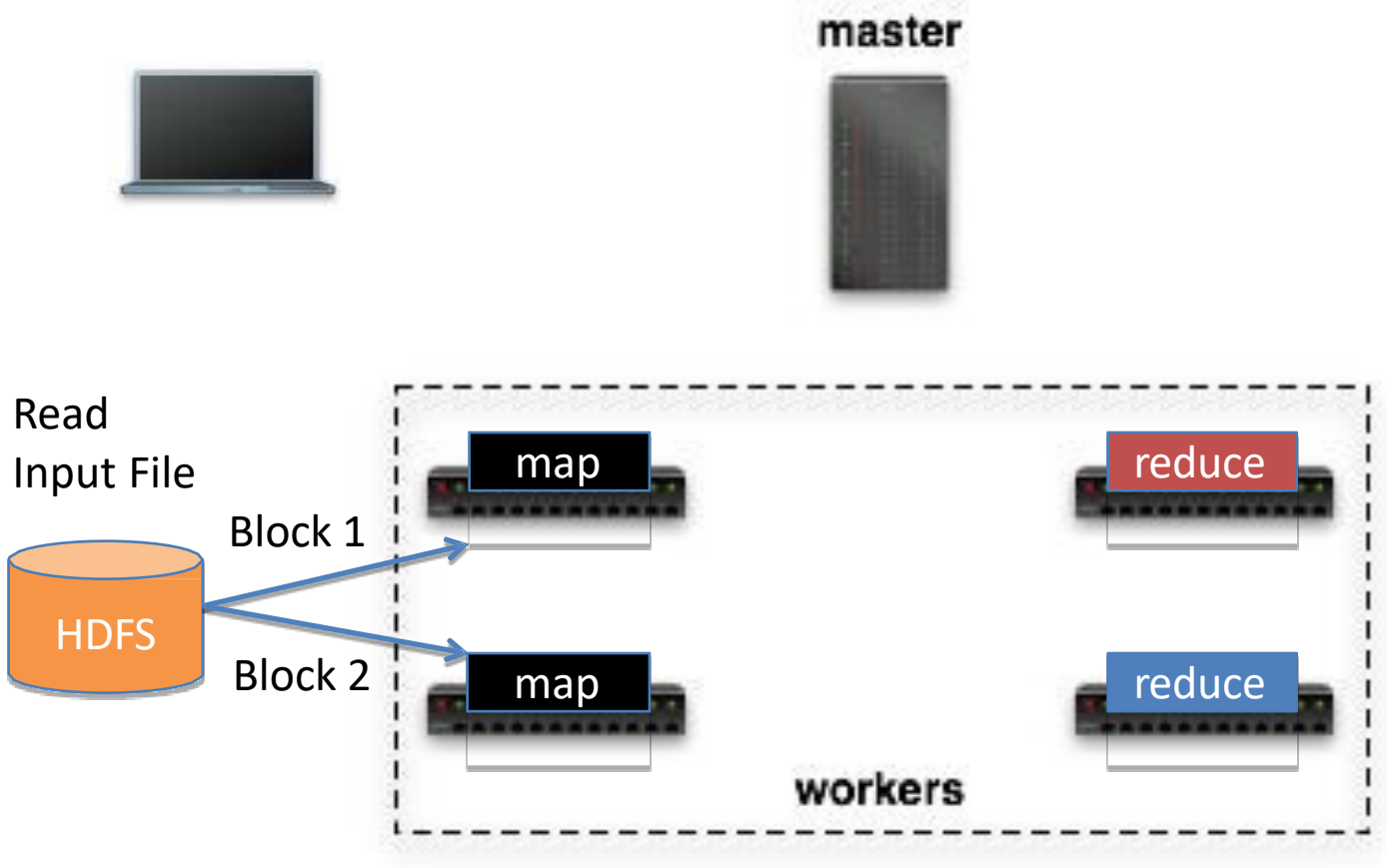
Source: https://www.slideshare.net/

Dataflow in Hadoop



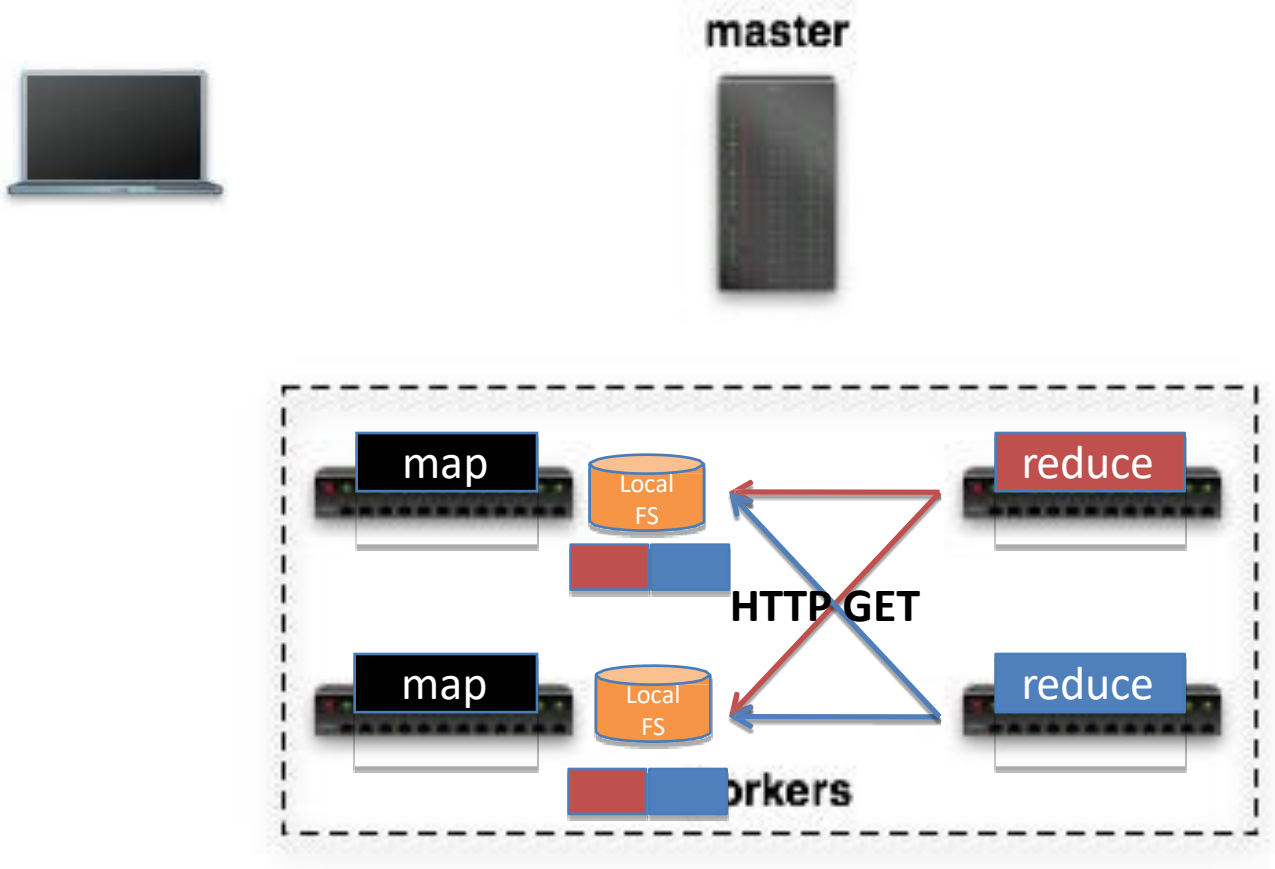
Source: https://www.slideshare.net/

Dataflow in Hadoop



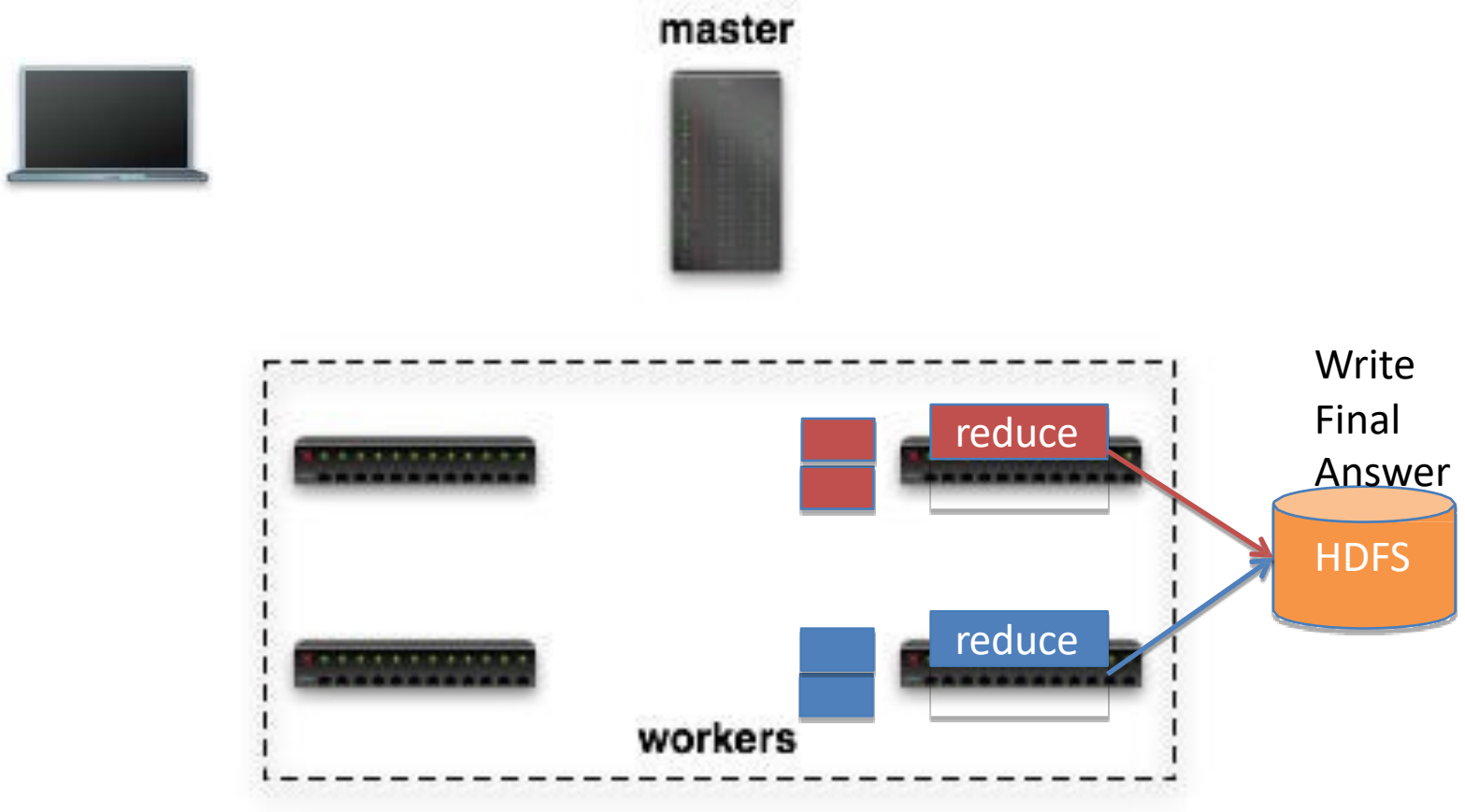
Source: https://www.slideshare.net/

Dataflow in Hadoop



Source: https://www.slideshare.net/

Dataflow in Hadoop



Source: https://www.slideshare.net/

Hadoop MapReduce

- Splits input files into blocks (typically of 64MB each)
- Operates on key/value pairs
- Mappers filter & transform input data
- Reducers aggregate mappers output
- Efficient way to process the cluster:
  - Move code to data
  - Run code on all machines

Source: https://www.slideshare.net/

Mapper

- Mapper maps input key/value pairs to a set of intermediate key/value pairs
- Maps are the individual tasks that transform input records into intermediate records
- The transformed intermediate records do not need to be of the same type as the input records
- A given input pair may map to zero or many output pairs

Source: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

Reducer

- Reducer reduces a set of intermediate values which share a key to a smaller set of values

- Reducer has 3 primary phases:
  - shuffle
  - sort
  - reduce

Source: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

Reducer

Shuffle

- Input to the Reducer is the sorted output of the mappers
- In this phase the framework fetches the relevant partition of the output of all the mappers, via HTTP

Sort

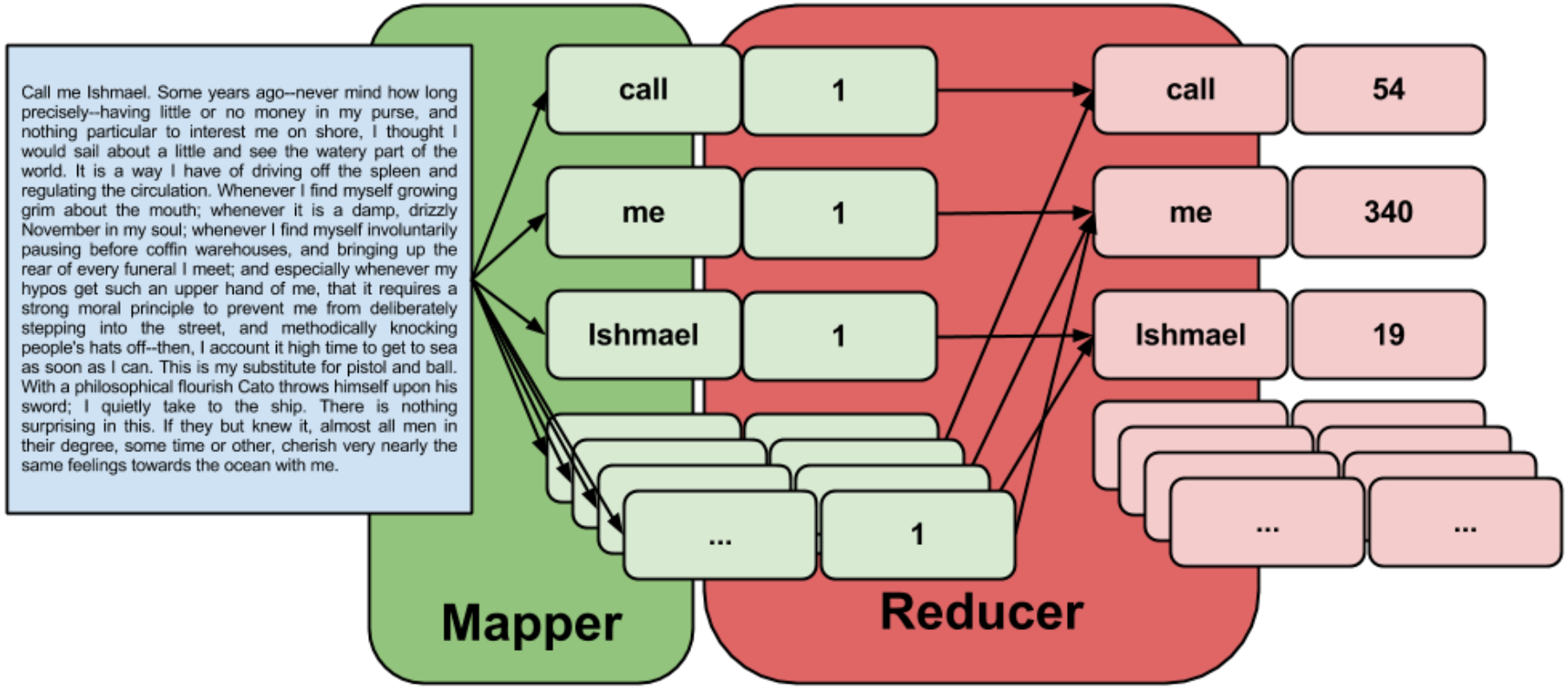
- The framework groups Reducer inputs by keys (since different mappers may have output the same key) in this stage
- The shuffle and sort phases occur simultaneously; while map-outputs are being fetched they are merged

Reduce

- In this phase the reduce method is called for each <key, (list of values)> pair in the grouped inputs
- The output of the reduce task is typically written to the FileSystem
- The output of the Reducer is not sorted

Source: https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

Hadoop MapReduce



Source: http://www.glennklockwood.com/data-intensive/hadoop/streaming.html

Hadoop Distributed File System (HDFS)

- Data is distributed and replicated over multiple machines.
- Files are not stored in contiguously on servers broken up into blocks.
- Designed for large files (large means GB or TB)
- Block Oriented
- Linux Style commands (eg. ls, cp, mkdir, mv)

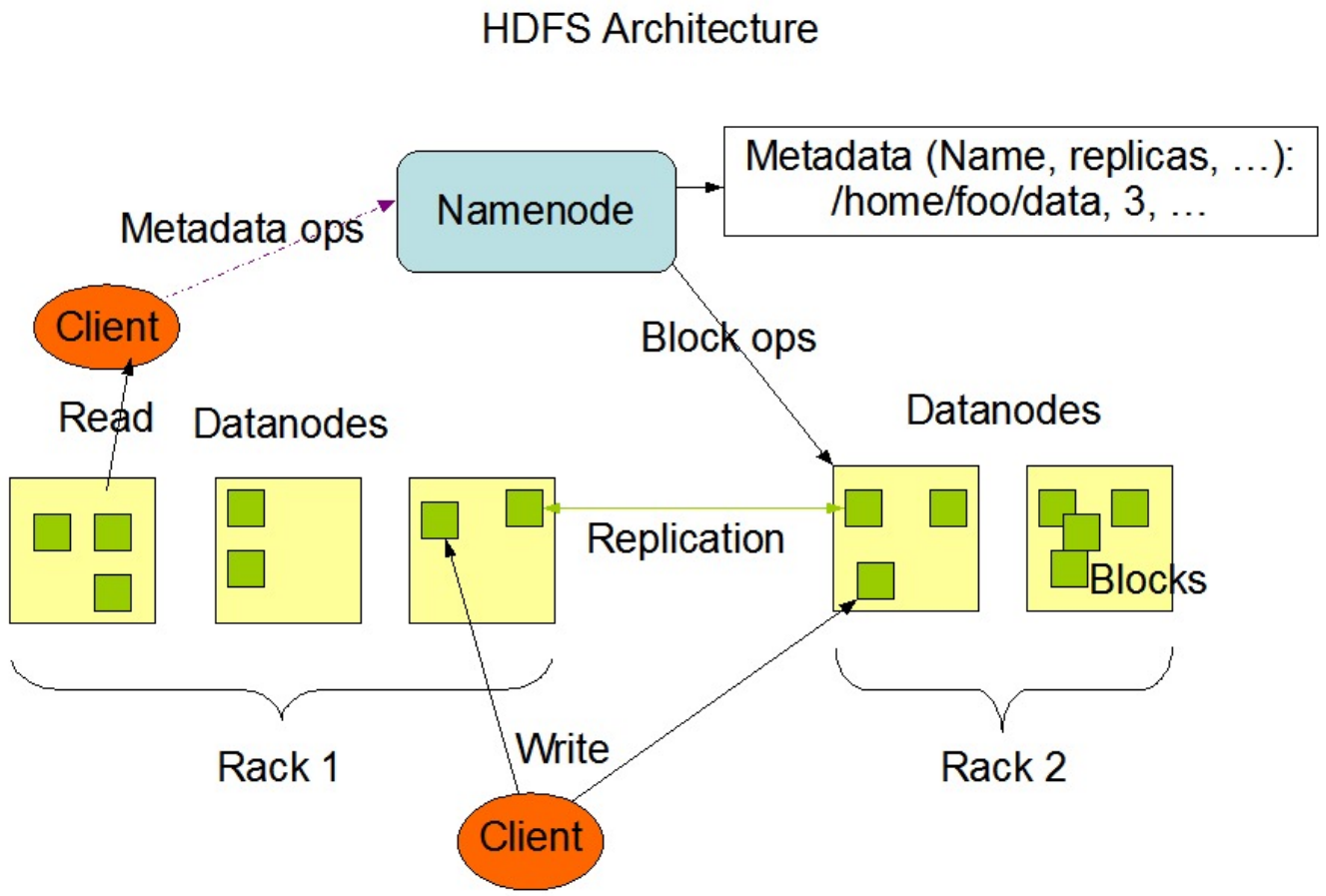
Source: https://www.slideshare.net/

Hadoop Distributed File System (HDFS)

- The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware
- HDFS is highly fault-tolerant and is designed to be deployed on low-cost hardware
- HDFS provides high throughput access to application data and is suitable for applications that have large data sets
- HDFS relaxes a few POSIX requirements to enable streaming access to file system data
- HDFS was originally built as infrastructure for the Apache Nutch web search engine project
- HDFS is part of the Apache Hadoop Core project

Source: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Hadoop Distributed File System (HDFS)



Source: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Hadoop Distributed File System (HDFS)

NameNode and DataNodes

- HDFS has a master/slave architecture
- An HDFS cluster consists of a single NameNode, a master server that manages the file system namespace and regulates access to files by clients
- In addition, there are a number of DataNodes, usually one per node in the cluster, which manage storage attached to the nodes that they run on
- HDFS exposes a file system namespace and allows user data to be stored in files
- Internally, a file is split into one or more blocks and these blocks are stored in a set of DataNodes

Source: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Hadoop Distributed File System (HDFS)

NameNode and DataNodes

- The NameNode executes file system namespace operations like opening, closing, and renaming files and directories
- It also determines the mapping of blocks to DataNodes
- The DataNodes are responsible for serving read and write requests from the file system's clients
- The DataNodes also perform block creation, deletion, and replication upon instruction from the NameNode
- The NameNode and DataNode are pieces of software designed to run on commodity machines

Source: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Hadoop Distributed File System (HDFS)

NameNode and DataNodes

- These machines typically run a GNU/Linux operating system (OS)
- HDFS is built using the Java language; any machine that supports Java can run the NameNode or the DataNode software
- Usage of the highly portable Java language means that HDFS can be deployed on a wide range of machines
- A typical deployment has a dedicated machine that runs only the NameNode software
- Each of the other machines in the cluster runs one instance of the DataNode software

Source: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Hadoop Distributed File System (HDFS)

NameNode and DataNodes

- The architecture does not preclude running multiple DataNodes on the same machine but in a real deployment that is rarely the case
- The existence of a single NameNode in a cluster greatly simplifies the architecture of the system
- The NameNode is the arbitrator and repository for all HDFS metadata
- The system is designed in such a way that user data never flows through the NameNode

Source: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html

Different Workflows

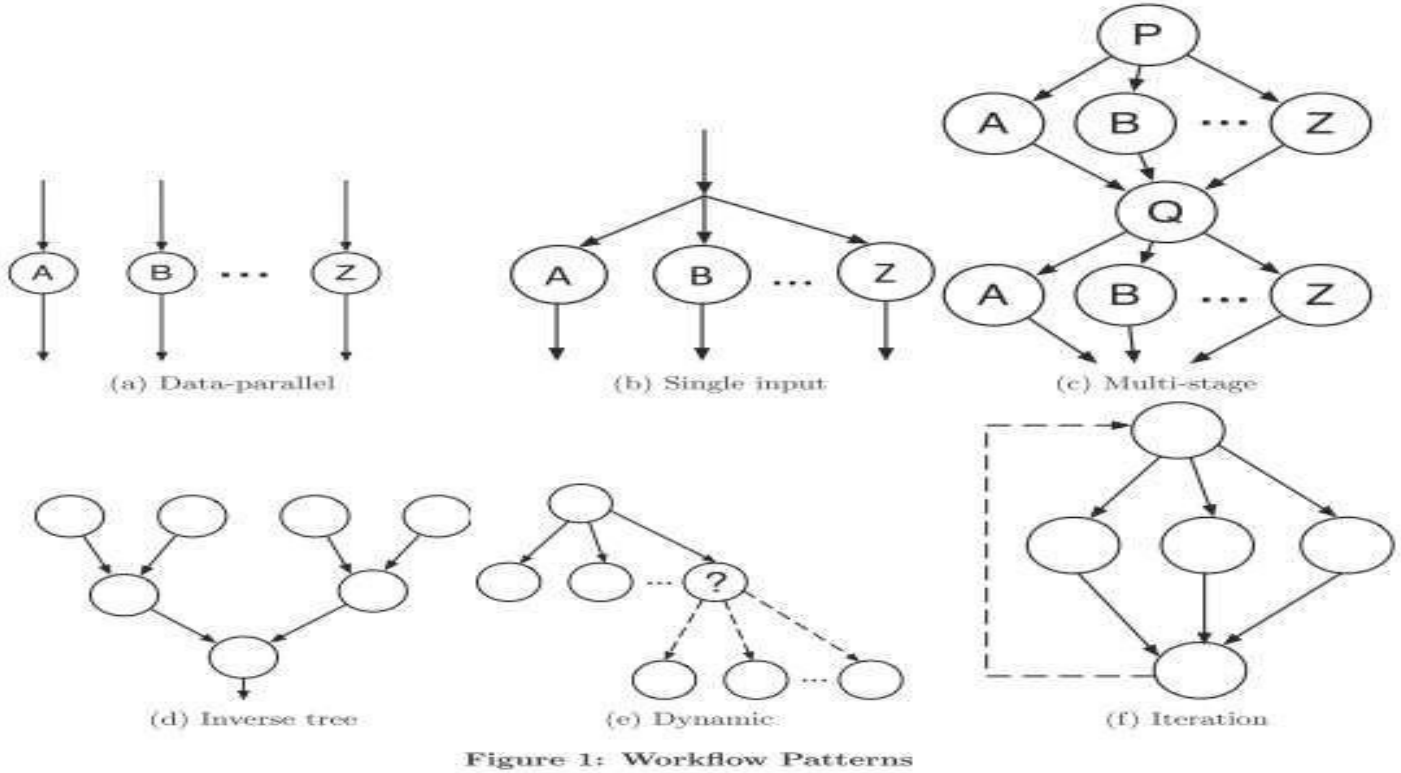


Figure 1: Workflow Patterns

Source: https://www.slideshare.net/

Hadoop Applicability by Workflow

Workflow Type	Hadoop Impl.	Data Management	MR Realization	Perf. & Reliability	Total Score
Data Parallel	○	○	○	○	0
Single Input	○	◒	◒	◒	1.5
Scatter-Gather	◒	◒	◒	◒	2
Inverse Tree	◒	●	●	◒	3
Dynamic	●	○	●	●	3.5
Iteration	●	○	●	●	3.5

Key: ○= Easy/Minimal ◒= Moderate ●= Difficult/Significant

Score Meaning:

Score Zero implies Easily adaptable to the workflow
Score 0.5 implies Moderately adaptable to the workflow
Score 1 indicates one of the potential workflow areas where Hadoop needs improvement

Source: https://www.slideshare.net/

Relative Merits and Demerits of Hadoop Over DBMS

Pros

- Fault tolerance
- Self Healing rebalances files across cluster
- Highly Scalable
- Highly Flexible as it does not have any dependency on data model and schema

Cons

- No high level language like SQL in DBMS
- No schema and no index
- Low efficiency
- Very young (since 2004) compared to over 40years of DBMS

Hadoop	Relational
Scale out (add more machines)	Scaling is difficult
Key/Value pairs	Tables
Say how to process the data	Say what you want (SQL)
Offline/ batch	Online/ realtime

Source: https://www.slideshare.net/

Conclusions and Future Work

- MapReduce is easy to program
- Hadoop=HDFS+MapReduce
- Distributed, Parallel processing
- Designed for fault tolerance and high scalability
- MapReduce is unlikely to substitute DBMS in data warehousing instead we expect them to complement each other and help in data analysis of scientific data patterns
- Finally, Efficiency and especially I/O costs needs to be addressed for successful implications

Source: https://www.slideshare.net/

Hadoop Commands

- https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html

References for papers

- [**LLCCM12**] Kyong-Ha Lee, Yoon-Joon Lee, Hyunsik Choi, Yon Dohn Chung, and Bongki Moon, Parallel data processing with MapReduce: a surveLJ, *SIGMOD*, January 2012, pp. 11-20.
- [**MTAGS11**] Elif Dede, Madhusudhan Govindaraju, Daniel Gunter, and Lavanya Ramakrishnan, RidiŶg the Elephant: Managing Ensembles with Hadoop, *Proceedings of the 2011 ACM international workshop on Many task computing on grids and supercomputers*, ACM, New York, NY, USA, pp. 49-58.
- [**DG08**]Jeffrey Dean and Sanjay Ghemawat, MapReduĐe: simplified data processing on large Đlusters, January 2008, pp. 107-113. ACM.
- [**CAHER10**]Tyson Condie, Neil Conway, Peter Alvaro, Joseph M. Hellerstein, Khaled Elmeleegy, aŶd Russell Sears, MapReduĐe oŶliŶe, *Proceedings of the 7th USENIX conference on Networked systems design and implementation* (NSDI'10), USENIX Association, Berkeley, CA, USA, 2010, pp. 21-37.

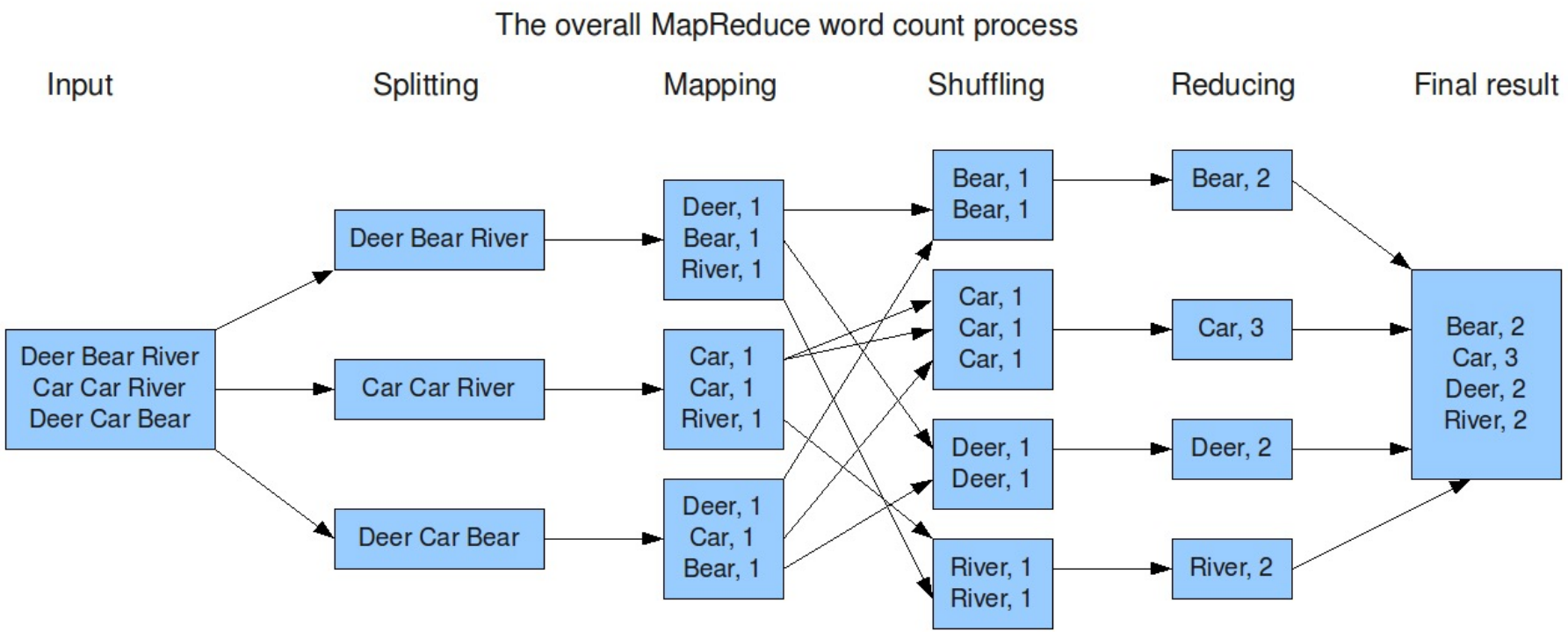
Source: https://www.slideshare.net/

References

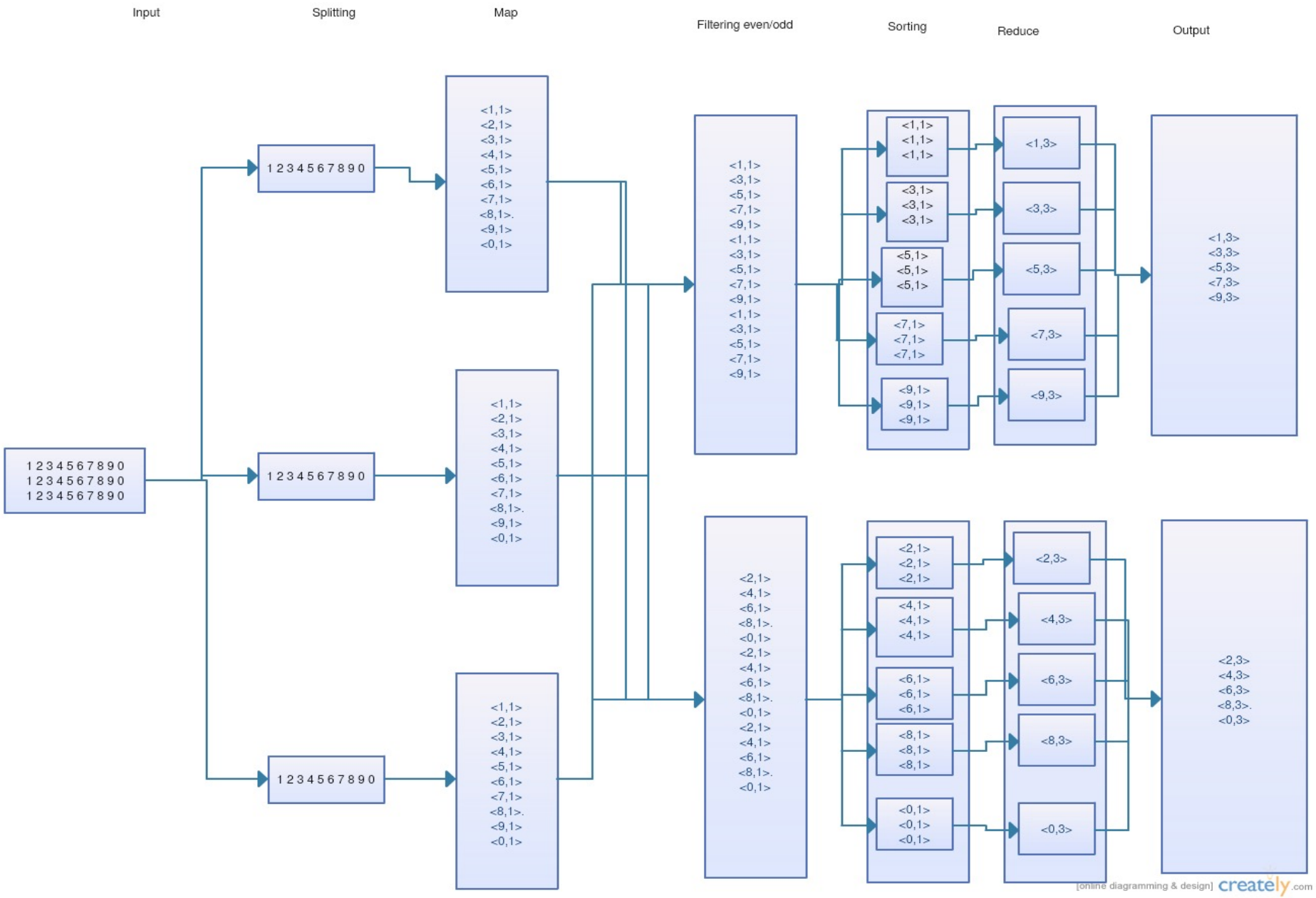
- http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/CommandsManual.html
- https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html
- https://hadoopi.wordpress.com/2013/05/25/setup-maven-project-for-hadoop-in-5mn/
- http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/

Source: https://www.slideshare.net/

Use case


The overall MapReduce word count process
Input
Splitting
Mapping
Shuffling
Reducing
Final result
Deer Bear River
Car Car River
Deer Car Bear
Deer Bear River
Car Car River
Deer Car Bear
Deer, 1
Bear, 1
River, 1
Car, 1
Car, 1
River, 1
Deer, 1
Car, 1
Bear, 1
Bear, 1
Bear, 1
Car, 1
Car, 1
Car, 1
Deer, 1
Deer, 1
River, 1
River, 1
Bear, 2
Car, 3
Deer, 2
River, 2
Bear, 2
Car, 3
Deer, 2
River, 2

ICE-2


Input
Splitting
Map
Filtering even/odd
Sorting
Reduce
Output
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
1 2 3 4 5 6 7 8 9 0
<1,1> <2,1> <3,1> <4,1> <5,1> <6,1> <7,1> <8,1>. <9,1> <0,1>
<1,1> <2,1> <3,1> <4,1> <5,1> <6,1> <7,1> <8,1>. <9,1> <0,1>
<1,1> <2,1> <3,1> <4,1> <5,1> <6,1> <7,1> <8,1>. <9,1> <0,1>
<1,1> <3,1> <5,1> <7,1> <9,1> <1,1> <3,1> <5,1> <7,1> <9,1> <1,1> <3,1> <5,1> <7,1> <9,1>
<2,1> <4,1> <6,1> <8,1>. <0,1> <2,1> <4,1> <6,1> <8,1>. <0,1> <2,1> <4,1> <6,1> <8,1>. <0,1>
<1,1> <1,1> <1,1>
<3,1> <3,1> <3,1>
<5,1> <5,1> <5,1>
<7,1> <7,1> <7,1>
<9,1> <9,1> <9,1>
<2,1> <2,1> <2,1>
<4,1> <4,1> <4,1>
<6,1> <6,1> <6,1>
<8,1> <8,1> <8,1>
<0,1> <0,1> <0,1>
<1,3>
<3,3>
<5,3>
<7,3>
<9,3>
<2,3>
<4,3>
<6,3>
<8,3>
<0,3>
<1,3> <3,3> <5,3> <7,3> <9,3>
<2,3> <4,3> <6,3> <8,3>. <0,3>
[online diagramming & design] creately.com