**Drexel University**

**College of Computing and Informatics**

**INFO 323 – Cloud Computing and Big Data**

**Assignment 2**

**Due Date: Sunday, January 30, 2022**

**A. Requirements**

1. Provide full and detailed answer to each question.
2. Work individually. The Drexel University Academic Honesty Rules and Procedures (as stated in the student handbook) will be adhered to strictly.
3. There are 5 questions. The total marks are 100.

**B. Questions**

**Question 1 [15 marks].** Explain the following terms and discuss the roles they play in cloud computing:

1. Distributed file systems
2. Commodity clusters
3. MapReduce

-Distributed file systems

A distributed file system for cloud is **a file system that allows many clients to have access to data and supports operations (create, delete, modify, read, write)** on that data. ... Typically, data is stored in files in a hierarchical tree, where the nodes represent directories.

-Commodity clusters

A commodity cluster is **a distributed computing system** consisting of an integrated set of fully and independently operational and marketed computer subsystems (node) used together to perform a single application program or workload.

-MapReduce

MapReduce is **a programming model or pattern within the Hadoop framework** that is used to access big data stored in the Hadoop File System (HDFS). ... MapReduce facilitates concurrent processing by splitting petabytes of data into smaller chunks, and processing them in parallel on Hadoop commodity servers

**MapReduce:** As we have large data volumes now a days to be processed in cloud applications and are continuously increasing day by day by the exponential growth of data with social media, internet, business organizations etc. over the world so it is very important for managing these data and their analysis. Hadoop MapReduce had become a popular in one of the big data tools for data computation and analysis in data science because of its flexibility, low-cost, large-scale volumes and works very fast for the vast amount of data with the help of parallel computation power on larger clusters of nodes.

**Question 2 [15 marks]**. Describe the key components and workflow of MapReduce’s parallel computing process. Describe how MapReduce coordinates communication and handles machine failures.

At a high level, MapReduce breaks input data into fragments and distributes them across different machines.

The input fragments consist of key-value pairs. Parallel map tasks process the chunked data on machines in a cluster. The mapping output then serves as input for the reduce stage. The reduce task combines the result into a particular key-value pair output and writes the data to HDFS.

The Hadoop Distributed File System usually runs on the same set of machines as the MapReduce software. When the framework executes a job on the nodes that also store the data, the time to complete the tasks is reduced significantly.

As the name suggests, MapReduce works by processing input data in two stages – **Map** and **Reduce**. To demonstrate this, we will use a simple example with counting the number of occurrences of words in each document.

The final output we are looking for is: *How many times the words Apache, Hadoop, Class, and Track appear in total in all documents*.

For illustration purposes, the example environment consists of three nodes. The input contains six documents distributed across the cluster. We will keep it simple here, but in real circumstances, there is no limit. You can have thousands of servers and billions of documents.

**Distributed Data Processing**

Distributed systems are often used to collect, access, and manipulate large data sets. For example, the database systems described earlier in the chapter can operate over datasets that are stored across multiple machines. No single machine may contain the data necessary to respond to a query, and so communication is required to service requests.

This section investigates a typical big data processing scenario in which a data set too large to be processed by a single machine is instead distributed among many machines, each of which process a portion of the dataset. The result of processing must often be aggregated across machines, so that results from one machine's computation can be combined with others. To coordinate this distributed data processing, we will discuss a programming framework called MapReduce.

Creating a distributed data processing application with MapReduce combines many of the ideas presented throughout this text. An application is expressed in terms of pure functions that are used to *map* over a large dataset and then to *reduce* the mapped sequences of values into a final result.

Familiar concepts from functional programming are used to maximal advantage in a MapReduce program. MapReduce requires that the functions used to map and reduce the data be pure functions. In general, a program expressed only in terms of pure functions has considerable flexibility in how it is executed. Sub-expressions can be computed in arbitrary order and in parallel without affecting the final result. A MapReduce application evaluates many pure functions in parallel, reordering computations to be executed efficiently in a distributed system.

The principal advantage of MapReduce is that it enforces a separation of concerns between two parts of a distributed data processing application:

1. The map and reduce functions that process data and combine results.
2. The communication and coordination between machines.

The coordination mechanism handles many issues that arise in distributed computing, such as machine failures, network failures, and progress monitoring. While managing these issues introduces some complexity in a MapReduce application, none of that complexity is exposed to the application developer. Instead, building a MapReduce application only requires specifying the map and reduce functions in (1) above; the challenges of distributed computation are hidden via abstraction.

**MapReduce**

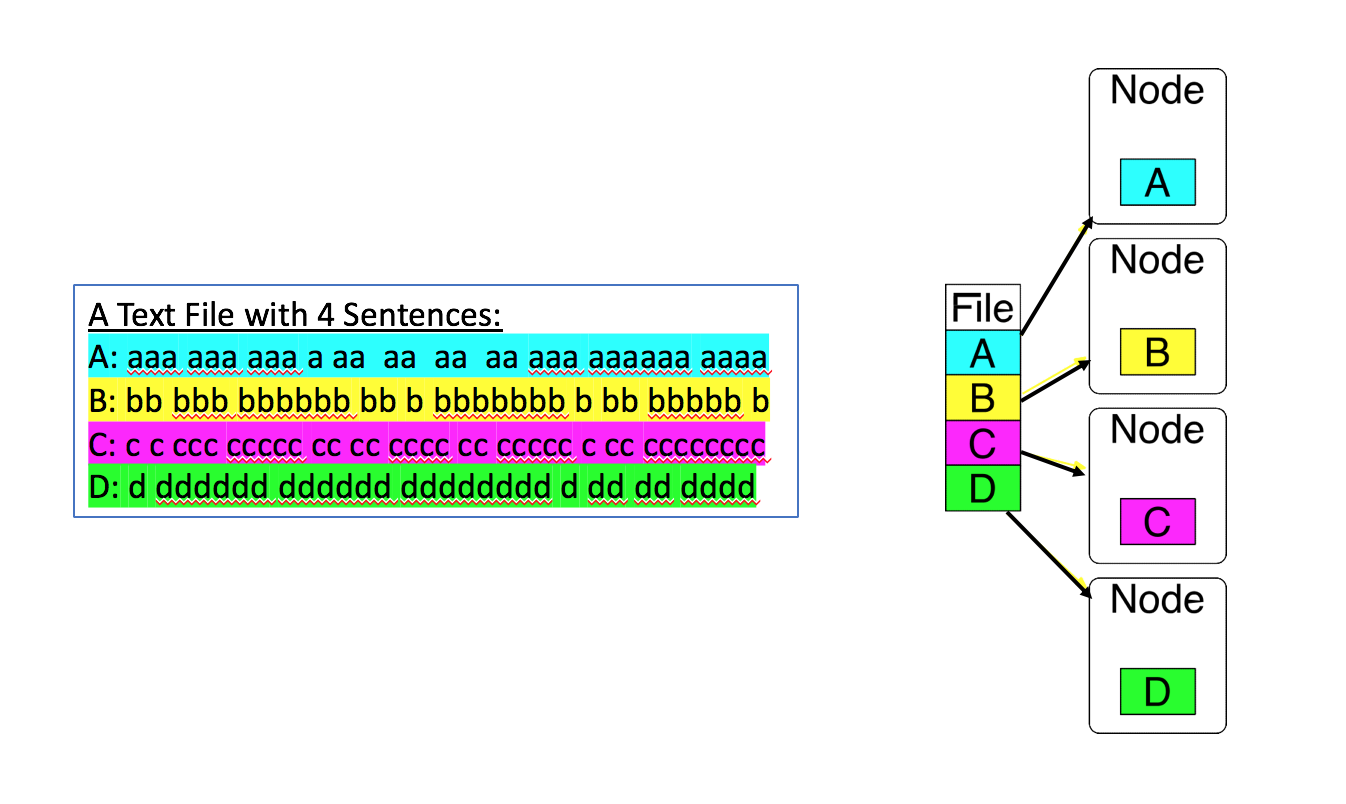
The MapReduce framework assumes as input a large, unordered stream of input values of an arbitrary type. For instance, each input may be a line of text in some vast corpus. Computation proceeds in three steps.

1. A map function is applied to each input, which outputs zero or more intermediate key-value pairs of an arbitrary type.
2. All intermediate key-value pairs are grouped by key, so that pairs with the same key can be reduced together.
3. A reduce function combines the values for a given key k; it outputs zero or more values, which are each associated with k in the final output.

To perform this computation, the MapReduce framework creates tasks (perhaps on different machines) that perform various roles in the computation. A *map task* applies the map function to some subset of the input data and outputs intermediate key-value pairs. A *reduce* task sorts and groups key-value pairs by key, then applies the reduce function to the values for each key. All communication between map and reduce tasks is handled by the framework, as is the task of grouping intermediate key-value pairs by key.

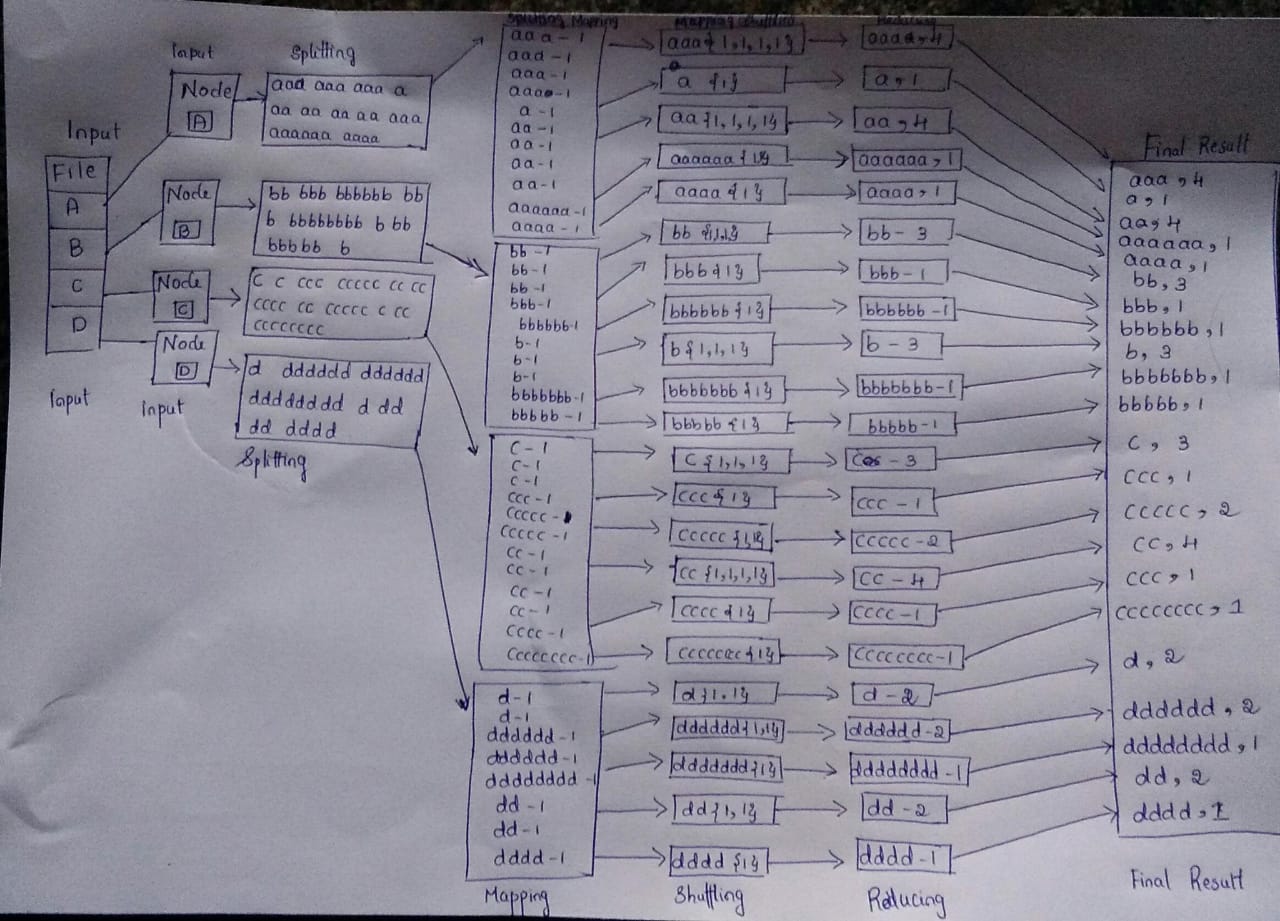
In order to utilize multiple machines in a MapReduce application, multiple mappers run in parallel in a *map phase*, and multiple reducers run in parallel in a *reduce phase*. In between these phases, the *sort phase* groups together key-value pairs by sorting them, so that all key-value pairs with the same key are adjacent.

**Question 3 [15 marks]**. Create a text file containing 4 real sentences. Assume you have a cluster with 4 nodes. Initially, the 4 sentences are distributed onto the 4 nodes. Each sentence is on a node (see Figure 1.) Describe the MapReduce process for computing the frequency of each word in the file. List the steps and show the intermediate and results.



**Figure 1: Initial Setting of a 4-node Cluster for Computing Word Counts by MapReduce**

**Solution :**

****

**Map reduce is a programming model used for processing of huge amounts of data. Map reduce work in two phases namely map and reduce. Map deals with the splitting and mapping of data while Reduce tasks is to shuffle and reduce the data.**

**Steps in map reduce :**

**1)splitting**

**2)Mapping**

**3)Shuffling**

**4)Reducing**

**5)output**

**Result of map reduce:**

**Question 4 [15 marks]**. **Hands-On Exercise.**

Create a GCP Dataproc cluster and SSH connect to the master node. Complete the following hands-on exercises in a terminal of the master node. At each step, you are required to capture screenshots to show the commands and results.

**Step 1**. In the terminal of the master node, create a folder named “info323-assign2” in the home directory on the master node.

Text

Description automatically generated with medium confidence

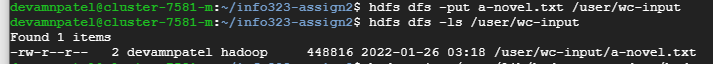
**Step 2**. Download a novel from https://www.gutenberg.org/browse/scores/top. Upload it to the folder “info323-assign2” as “a-novel.txt”.

Text

Description automatically generated

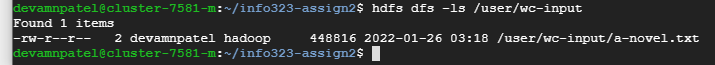
**Step 3**. Create a directory /user/wc-input in HDFS and copy the file a-novel.txt to the HDFS folder:

hdfs dfs -put ~/info323-assign2/a-novel.txt /user/wc-input/



**Step 4**. List the file in the HDFS folder:

hdfs dfs -ls /user/wc-input\



**Step 5**. Check the Hadoop MapReduce example programs: Hadoop comes with several example MapReduce applications. You can check the examples by running:

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar.

Text

Description automatically generated

*We are interested in running WordCount*.

**Step 6.** Check the WordCount command line arguments: You can learn how to run WordCount by examining its command-line arguments:

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount

**Step 7.** Run WordCount on “a-novel.txt”. If the output folder /user/wc-output exists in HDFS, you need delete the folder first.

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-mapreduce-examples.jar wordcount /user/wc-input/a-novel.txt /user/wc-output

Text

Description automatically generated

Text

Description automatically generated

**Step 8.** Look inside the output directory /user/wc-output**:** The directory created by WordCount contains several files. List the content of the directory:

hdfs dfs -ls /user/wc-output

Graphical user interface, text

Description automatically generated

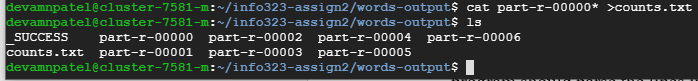
**Step 9.** Copy the output folder in the HDFS to a folder named output in the master node:

hdfs dfs -get /user/wc-output output



**Step 10**. Combine all the partitions in the output folder as a file named counts.txt:

cat part-r-0000\* > counts.txt



**Step 11.** Check how many lines in the file counts.txt**.**



**Step 12.** View some of the WordCount results**:** Run a command to print out 20 lines **in the middle of the file** counts.txt.

Text

Description automatically generated

**Question 5 [40 marks]**. **Programming Exercises in Spark**.

In this exercise, you are asked to write Spark program to count the flights’ destination countries in the given file “flights-dest-origin.csv”. Each line in the file consists of 3 fields, Destination, Origin, and NumberOfFlights. The Spark program should parse the lines in the file and output each unique destination country name along with its total number of occurrences in the Destination field.

You are asked to complete this question in the Jupyter notebook “INFO323-Assignment2-Question5.ipynb”:

* Create a Dataproc cluster which links to a Google storage bucket.
* Download the Jupyter notebook and the data file “flights-dest-origin.csv” to your local file system.
* Upload the Jupypter notebook to the folder ‘notebooks/jupyter’ in the Google storage bucket that has been linked to the Dataproc cluster.
* Open the Jupyter notebook in the Dataproc cluster.
* Follow the steps in the notebook to complete the exercises.

**Submit your completed Jupyter notebook and the two output files (no screenshot needed). Answer the following question**:

* Discuss the differences between the two exercises of using Google File System and HDFS in terms of data and resource management (such as data partitioning, data replication, fault tolerance management, resources management, etc).

**C. What to Hand In**

1. A Microsoft Word document including the following items:
2. Your name
3. Course number and title
4. Assignment number
5. Assignment questions and detailed answers to the questions
6. The completed Jupyter notebook and required outputs for the programming exercise.

**D. How to Hand In**

1. Please name your assignment WORD file as **INFO323-assign2-yourFirstName-yourLastName.docx**.
2. Submit your assignment files through the course website in the **Blackboard Learn** system.

**E. When to Hand In**

1. Submit your assignment no later than **11:59pm** in the due date.
2. There will be a 10% (absolute value) deduction for each day of lateness, to a maximum of 3 days; assignments will not be accepted beyond that point. Missing work will earn a zero grade.

**F. Written Presentation Requirements**

Images must be clear and legible. Assignments will be judged on the basis of visual appearance, grammatical correctness, and quality of writing, as well as their contents. Please make sure that the text of your assignments is well-structured, using paragraphs, full sentences, and other features of well-written presentation. Text font size should be at least 11 points.