Project: MapReduce program+ Full Inverted Index

CS570 - Big Data Processing & Analytics

By Lidia Bereketeab(20029)

TABLE OF CONTENTS

1.Introduction

2.Overview

3.Partial Inverted Index

4. Full Inverted Index

5.Implementation

6. Result

7.Conclusion

8.References

01 Introduction

INVERTED INDEX & MAPREDUCE

Inverted Index

- Purpose: Enhances the efficiency of text retrieval by reorienting the document-term relationship.
- Structure: Rather than cataloging documents by their terms, it catalogs terms by their respective documents.

MapReduce Framework

- Functionality: Facilitates the parallel processing of extensive datasets.
- Suitability: Perfectly designed for managing and processing large-scale data.

01 Introduction

IMPLEMENTING INVERTED INDEX WITH MAPREDUCE

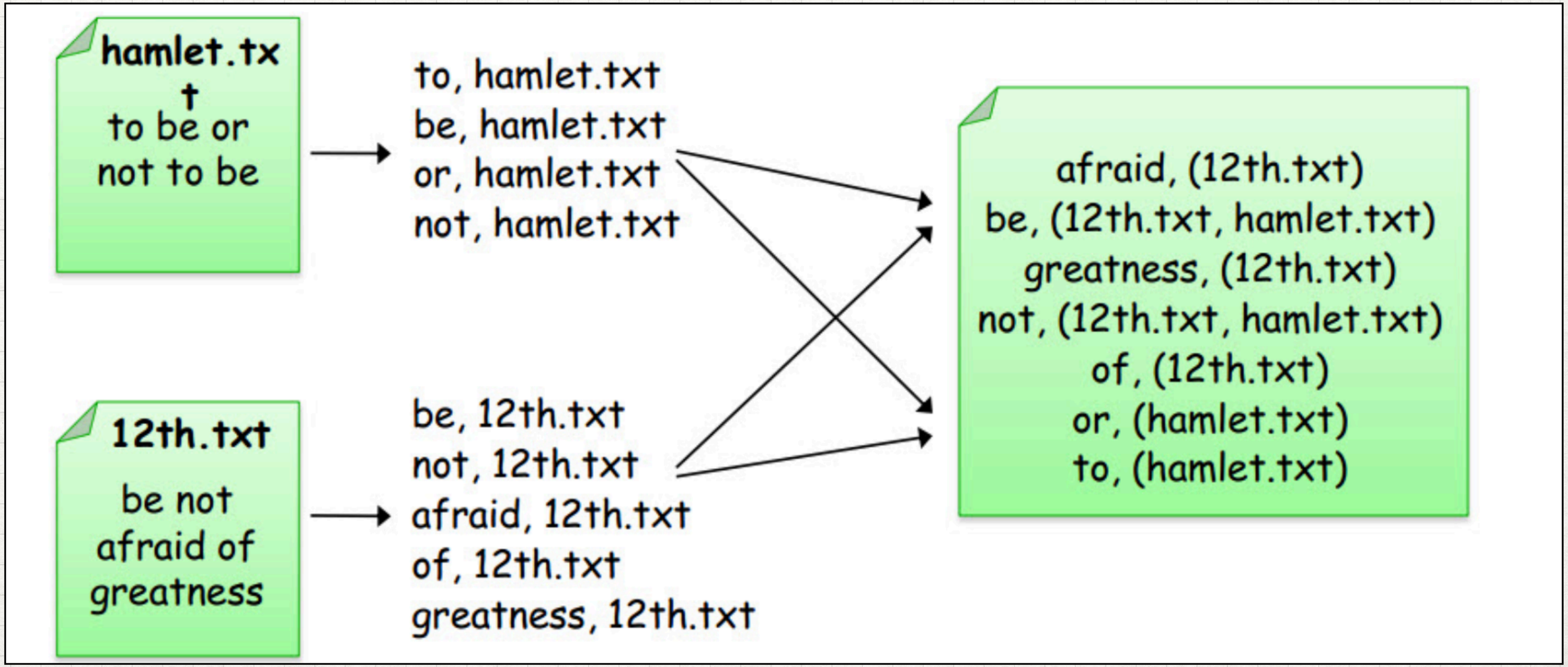
Process Overview

- **Map Phase**: Scans documents and extracts terms.
- **Reduce Phase**: Collects and organizes document lists corresponding to each term.

Benefits

- **Efficiency**: Enables swift retrieval of documents containing specific terms or term combinations.
- **Applications**: Crucial for search engines and information retrieval systems

02 OVERVIEW


hamlet.txt
to be or
not to be
12th.txt
be not
afraid of
greatness
to, hamlet.txt
be, hamlet.txt
or, hamlet.txt
not, hamlet.txt
be, 12th.txt
not, 12th.txt
afraid, 12th.txt
of, 12th.txt
greatness, 12th.txt
afraid, (12th.txt)
be, (12th.txt, hamlet.txt)
greatness, (12th.txt)
not, (12th.txt, hamlet.txt)
of, (12th.txt)
or, (hamlet.txt)
to, (hamlet.txt)

Partial Inverted Index

Advantages

- **Space Efficiency**: Requires less storage than a fully inverted index.
- **Suitability**: Ideal for environments with storage constraints.

Mapper				Reducer			
Input Key	Input Value	Output Key	Output Value	Input Key	Input Key	Output Key	Output Value
0	it is what it is	it	0	a	{2}	a	{2}
		is	0	banana	{2}	banana	{2}
		what	0	is	{0,0,1,2}	is	{0,1,2}
		it	0	it	{0,0,1,2}	it	{0,1,2}
		is	0	what	{0,1}	what	{0,1}
1	what is it	what	1				
		is	1				
		it	1				
2	it is a banana	it	2				
		is	2				
		a	2				
		banana	2				

Full Inverted Index

Benefits

- **Comprehensive Search**: Stores every term and its locations across all documents.
- **Speed**: Facilitates rapid and thorough searching.

Mapper				Reducer			
Input Key	Input Value	Output Key	Output Value	Input Key	Input Key	Output Key	Output Value
file0	it is what it is	it	(0,0)	a	{(2,2)}	a	{(2,2)}
		is	(0,1)	banana	{(2,3)}	banana	{(2,3)}
		what	(0,2)	is	{(0,1),(0,4),(1,1),(2,1)}	is	{(0,1),(0,4),(1,1),(2,1)}
		it	(0,3)	it	{(0,0),(0,3),(1,2),(2,0)}	it	{(0,0),(0,3),(1,2),(2,0)}
		is	(0,4)	what	{(0,2),(1,0)}	what	{(0,2),(1,0)}
file1	what is it	what	(1,0)				
		is	(1,1)				
		it	(1,2)				
file2	it is a banana	it	(2,0)				
		is	(2,1)				
		a	(2,2)				
		banana	(2,3)				

IMPLEMENTATION

STEPS (CONTINUED)

Create Directory:

- Name: inverted_index

Create Files with Content:

- file0.txt: "it is what it is"
- file1.txt: "what is it"
- file3.txt: "it is a banana"

```
lbereket625@cloudshell:~ (cs-570-big-data-424802)$ cd inverted_index
lbereket625@cloudshell:~/inverted_index (cs-570-big-data-424802)$ ls
file0.txt  file1.txt  file3.txt
lbereket625@cloudshell:~/inverted_index (cs-570-big-data-424802)$ cat file0.txt

it is what it is
lbereket625@cloudshell:~/inverted_index (cs-570-big-data-424802)$ cat file1.txt
what is it
lbereket625@cloudshell:~/inverted_index (cs-570-big-data-424802)$ cat file3.txt
it is a banana
lbereket625@cloudshell:~/inverted_index (cs-570-big-data-424802)$
```

```
lbereket625@cloudshell:~/inverted_index (cs-570-big-data-424802)$ ls
DocSunWritable.java  file0.txt  file1.txt  file3.txt  IndexMapper.java  IndexReducer.java  InvertedIndexDriver.java
lbereket625@cloudshell:~/inverted_index (cs-570-big-data-424802)$
```

IMPLEMENTATION

LOREM IPSUM

Transforming WordCount to Partial Inverted Index

Objective: Modify a WordCount MapReduce program to implement a Partial Inverted Index with three input files, demonstrating the expected output.

1. **File Creation:**
   - Create an InvertedIndex.java file within the Hadoop-3.3.4 directory.

```
rm: cannot remove 'InvertedIndex.java': No such file or directory
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$ vim InvertedIndex.java
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$ cat InvertedIndex.java

import java.io.IOException;
import java.util.ArrayList;
import java.util.List;
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.conf.Configured;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.mapreduce.Counter;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

public class InvertedIndex extends Configured implements Tool {

    public static class InvertedIndexMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
        public static final String MALFORMED_DATA = "MALFORMED";
        private Text outkey = new Text();
        private IntWritable outvalue = new IntWritable(1);

        public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
            String line = value.toString();
            String[] words = line.split("\\s+");
            for (String word : words) {
                word = word.trim();
                if (word.equals("#")) {
                    context.getCounter(MALFORMED_DATA, word).increment(1);
                } else {
                    outkey.set(word);
                    outvalue.set(1); // Setting value to 1 for each occurrence
                    context.write(outkey, outvalue);
                }
```

IMPLEMENTATION

PURPOSE

Inverted Index MapReduce

Purpose: Build an inverted index from text files using Hadoop MapReduce.

Components:

- **Mapper (InvertedIndexMapper):** Emits word counts from each line. Counts malformed data.
- **Reducer (InvertedIndexReducer):** Aggregates word occurrences and lists unique document IDs.
- **Driver (InvertedIndex):** Configures and runs the job, specifying input/output paths and classes.

Command: java -jar <jar-file> <input-path> <output-path>

Outcome: Maps words to document IDs where they appear.

```java
public static class InvertedIndexReducer extends Reducer<Text, IntWritable, Text, Text> {
    private Text outputkey = new Text();
    private List<Integer> outputvalue = new ArrayList<>();

    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
        outputkey.set(key.toString());
        outputvalue.clear();
        for (IntWritable val : values) {
            int value = val.get();
            if (!outputvalue.contains(value)) {
                outputvalue.add(value);
            }
        }
        context.write(outputkey, new Text(outputvalue.toString()));
    }
}

@Override
public int run(String[] args) throws Exception {
    if (args.length != 2) {
        System.out.printf("Usage: %s [generic options] <input> <output>\n", getClass().getSimpleName());
        return -1;
    }

    Configuration conf = getConf();
    Job job = Job.getInstance(conf, "InvertedIndex");
    job.setJarByClass(InvertedIndex.class);
    job.setMapperClass(InvertedIndexMapper.class);
    job.setReducerClass(InvertedIndexReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    job.setInputFormatClass(TextInputFormat.class);
    job.setOutputFormatClass(TextOutputFormat.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    boolean success = job.waitForCompletion(true);

    if (success) {
        for (Counter counter : job.getCounters().getGroup(InvertedIndexMapper.MALFORMED_DATA)) {
```

```java
        boolean success = job.waitForCompletion(true);

        if (success) {
            for (Counter counter : job.getCounters().getGroup(InvertedIndexMapper.MALFORMED_DATA)) {
                System.out.println(counter.getDisplayName() + ": " + counter.getValue());
            }
        }
        return success ? 0 : 1;
    }

    public static void main(String[] args) throws Exception {
        int exitCode = ToolRunner.run(new Configuration(), new InvertedIndex(), args);
        System.exit(exitCode);
    }
}
```

```
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$
```

IMPLEMENTATION

PROGRAMS

Java Programs for Inverted Index

- **`DocSumWritable.java`**: Stores document names and counts.
- **`IndexMapper.java`**: Emits pairs of words and document names.
- **`IndexReducer.java`**: Aggregates document names and counts for each word.
- **`InvertedIndexDriver.java`**: Manages the MapReduce job execution.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ ls
DocSumWritable.java  FullIndex  Hadoop-3.3.4  IndexMapper.java  IndexReducer.java  InvertedIndexDriver.java
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$
```

IMPLEMENTATION

PROGRAMS

Java Programs for Inverted Index

- **`DocSumWritable.java`**: Stores document names and counts.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ vim DocSumWritable.java
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ cat DocSumWritable.java

import java.io.DataInput;
import java.io.DataOutput;
import java.io.IOException;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.io.WritableUtils;

public class DocSumWritable implements WritableComparable<DocSumWritable> {
    private String documentName;
    private int count;

    public DocSumWritable() {}

    public DocSumWritable(String documentName, int count) {
        this.documentName = documentName;
        this.count = count;
    }

    @Override
    public void write(DataOutput out) throws IOException {
        WritableUtils.writeString(out, documentName);
        out.writeInt(count);
    }

    @Override
    public void readFields(DataInput in) throws IOException {
        documentName = WritableUtils.readString(in);
        count = in.readInt();
    }

    @Override
    public int compareTo(DocSumWritable o) {
        int cmp = documentName.compareTo(o.documentName);
        return (cmp != 0) ? cmp : Integer.compare(count, o.count);
    }

    @Override
    public String toString() {
        return documentName + ":" + count;
    }
```

IMPLEMENTATION

PROGRAMS

Java Programs for Inverted Index

- **`IndexMapper.java`**: Emits pairs of words and document names.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ vim IndexMapper.java
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ cat IndexMapper.java

import java.io.IOException;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

public class IndexMapper extends Mapper<LongWritable, Text, Text, DocSumWritable> {
    private Text outKey = new Text();
    private DocSumWritable outValue = new DocSumWritable();

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String[] parts = value.toString().split("\t");
        String documentName = parts[0];
        String[] words = parts[1].split("\\s+");

        for (String word : words) {
            outKey.set(word);
            outValue.set(documentName, 1);  // Document name with count 1
            context.write(outKey, outValue);
```

IMPLEMENTATION

PROGRAMS

Java Programs for Inverted Index

- **`IndexReducer.java`**: Aggregates document names and counts for each word.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ vim IndexReducer.java
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ cat IndexReducer.java

import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

public class IndexReducer extends Reducer<Text, DocSumWritable, Text, Text> {
    private Map<String, Integer> documentCounts = new HashMap<>();

    @Override
    public void reduce(Text key, Iterable<DocSumWritable> values, Context context) throws IOException, InterruptedException {
        documentCounts.clear();
        for (DocSumWritable val : values) {
            String documentName = val.getDocumentName();
            int count = val.getCount();
            documentCounts.merge(documentName, count, Integer::sum);
        }

        StringBuilder result = new StringBuilder();
        for (Map.Entry<String, Integer> entry : documentCounts.entrySet()) {
            result.append(entry.getKey()).append(":").append(entry.getValue()).append(", ");
        }

        context.write(key, new Text(result.toString()));
```

IMPLEMENTATION

PROGRAMS

Java Programs for Inverted Index

- **`InvertedIndexDriver.java`**: Manages the MapReduce job execution.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ vim InvertedIndexDriver.java
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ cat InvertedIndexDriver.java
import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

public class InvertedIndexDriver {
    public static void main(String[] args) throws Exception {
        if (args.length != 2) {
            System.err.println("Usage: InvertedIndexDriver <input path> <output path>");
            System.exit(-1);
        }

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Inverted Index");
        job.setJarByClass(InvertedIndexDriver.class);
        job.setMapperClass(IndexMapper.class);
        job.setReducerClass(IndexReducer.class);
        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(DocSumWritable.class);
        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        FileOutputFormat.setOutputPath(job, new Path(args[1]));

        System.exit(job.waitForCompletion(true) ? 0 : 1);
```

IMPLEMENTATION

PROGRAMS

Update .bashrc Configuration:

Add the following lines to your .bashrc file to set up Hadoop environment variables:

export HADOOP_HOME=/usr/local/hadoop

export HADOOP_INSTALL=$HADOOP_HOME

export HADOOP_MAPRED_HOME=$HADOOP_HOME

export HADOOP_COMMON_HOME=$HADOOP_HOME

export HADOOP_HDFS_HOME=$HADOOP_HOME

export YARN_HOME=$HADOOP_HOME

export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native

export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ echo 'export HADOOP_HOME=/usr/local/hadoop' >> ~/.bashrc && \
echo 'export HADOOP_INSTALL=$HADOOP_HOME' >> ~/.bashrc && \
echo 'export HADOOP_MAPRED_HOME=$HADOOP_HOME' >> ~/.bashrc && \
echo 'export HADOOP_COMMON_HOME=$HADOOP_HOME' >> ~/.bashrc && \
echo 'export HADOOP_HDFS_HOME=$HADOOP_HOME' >> ~/.bashrc && \
echo 'export YARN_HOME=$HADOOP_HOME' >> ~/.bashrc && \
echo 'export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native' >> ~/.bashrc && \
echo 'export PATH=$PATH:$HADOOP_HOME/sbin:$HADOOP_HOME/bin' >> ~/.bashrc
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$
```

IMPLEMENTATION

CONFIGURATION FILES

Hadoop Configuration Files

- **core-site.xml**:
  - Configures the default file system.
  - Sets fs.defaultFS to hdfs://localhost:9000.
- **hdfs-site.xml**:
  - Specifies HDFS configurations like replication factor and data directory.
- **mapred-site.xml**:
  - Contains MapReduce settings.
  - Configures the framework's job tracker and task tracker details.
- **yarn-site.xml**:
  - Configures YARN (Yet Another Resource Negotiator).
  - Sets resource manager and node manager settings.

IMPLEMENTATION

PROGRAMS

Hadoop Configuration Files

- **core-site.xml**:
  - Configures the default file system.
  - Sets fs.defaultFS to hdfs://localhost:9000.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ cat <<EOF > $HADOOP_HOME/etc/hadoop/core-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
 <property>
  <name>fs.defaultFS</name>
  <value>hdfs://localhost:9000</value>
 </property>
</configuration>
EOF
```

IMPLEMENTATION

PROGRAMS

Hadoop Configuration Files

- **mapred-site.xml**:
  - Contains MapReduce settings.
  - Configures the framework's job tracker and task tracker details.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ cat <<EOF > $HADOOP_HOME/etc/hadoop/mapred-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
 <property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
 </property>
</configuration>
EOF
```

IMPLEMENTATION

PROGRAMS

Hadoop Configuration Files

- **yarn-site.xml**:
  - Configures YARN (Yet Another Resource Negotiator).
  - Sets resource manager and node manager settings.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ cat <<EOF > $HADOOP_HOME/etc/hadoop/yarn-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
 <property>
  <name>yarn.nodemanager.aux-services</name>
  <value>mapreduce_shuffle</value>
 </property>
</configuration>
EOF
```

IMPLEMENTATION

PROGRAMS

Hadoop Configuration Files

- **hdfs-site.xml**:
  - Specifies HDFS configurations like replication factor and data directory.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ cat <<EOF > $HADOOP_HOME/etc/hadoop/hdfs-site.xml
<?xml version="1.0" encoding="UTF-8"?>
<configuration>
 <property>
  <name>dfs.replication</name>
  <value>1</value>
 </property>
 <property>
  <name>dfs.namenode.name.dir</name>
  <value>file:///usr/local/hadoop/hdfs/namenode</value>
 </property>
 <property>
  <name>dfs.datanode.data.dir</name>
  <value>file:///usr/local/hadoop/hdfs/datanode</value>
 </property>
</configuration>
EOF
```

IMPLEMENTATION

SETUP AND INITIALIZATION

Hadoop Setup and Initialization

1. **Start SSH Session:**
   - **Command:** `ssh localhost`
   - **Purpose:** Initiates an SSH session on the local machine for Hadoop operations.
2. **Start HDFS and YARN:**
   - **Command to Start HDFS:** `start-dfs.sh`
   - **Command to Start YARN:** `start-yarn.sh`
   - **Purpose:** Initializes HDFS (Hadoop Distributed File System) and YARN (Yet Another Resource Negotiator) services.
3. **Create HDFS Input Directory:**
   - **Command:** `hadoop fs -mkdir -p /user/your_username/input`
   - **Purpose:** Creates an HDFS directory for storing input data.

IMPLEMENTATION

PROGRAMS

Hadoop Setup and Initialization

1. Start SSH Session:

- **Command:** `ssh localhost`
- **Purpose:** Initiates an SSH session on the local machine for Hadoop operations.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ sbin/start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [cs-570-big-data-424802]
cs-570-big-data-424802: secondarynamenode is running as process 1820. Stop it first and ens
lbereket625-secondarynamenode.pid file is empty before retry.
2024-06-05 09:27:10,659 WARN util.NativeCodeLoader: Unable to load native-hadoop library fo
```

IMPLEMENTATION

PROGRAMS

Hadoop Setup and Initialization

Start HDFS and YARN:

- **Command to Start HDFS:** `start-dfs.sh`
- **Command to Start YARN:** `start-yarn.sh`
- **Purpose:** Initializes HDFS (Hadoop Distributed File System) and YARN (Yet Another Resource Negotiator) services.

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$  sbin/start-yarn.sh
Starting resourcemanager
Starting nodemanagers
```

IMPLEMENTATION

PROGRAMS

Hadoop Setup and Initialization

Upload Input Files to HDFS

hadoop fs -put ~/hadoop_input/file0.txt /user/your_username/input
hadoop fs -put ~/hadoop_input/file1.txt /user/your_username/input
hadoop fs -put ~/hadoop_input/file2.txt /user/your_username/input

Compile Java Files javac -classpath $(hadoop classpath) -d ~/hadoop_project/classes ~/hadoop_project/*.java

```
lbereket625@cloudshell:~/hadoop-3.3.4 (cs-570-big-data-424802)$ javac -classpath $(hadoop classpath) -d ~/hadoop_proj
ect/classes ~/hadoop_project/*.java
```

IMPLEMENTATION

RUNNING

Running a Hadoop MapReduce Job

1. **Create JAR File:**
   - **Command:** `jar -cvf ~/hadoop_project/invertedindex.jar -C ~/hadoop_project/classes/ .`
   - **Purpose:** Packages the compiled Java classes into a JAR file for execution.
2. **Run the MapReduce Job:**
   - **Command:** `hadoop jar ~/hadoop_project/invertedindex.jar InvertedIndexDriver /user/your_username/input /user/your_username/output`
   - **Purpose:** Executes the MapReduce job using the specified JAR file and driver class. Input and output directories are provided.
3. **List Output Directory:**
   - **Command:** `hadoop fs -ls /user/your_username/output`
   - **Purpose:** Displays the contents of the output directory in HDFS to check the results.

IMPLEMENTATION

RUNNING

The command runs a Hadoop job using the InvertedIndex.jar to process data from /tmp/inputfile and output results to /tmp/outputfile.

```
lbereket625@cloudshell:~/hadoop-3.3.4/bin (cs-570-big-data-424802)$ hadoop jar /home/lbereket625/hadoop-3.3.4/FullIndex/Invertedindex.jar InvertedIndex /tmp/inputfile /tmp/outputf
e

2024-06-05 09:50:09,353 INFO imp.MetricsConfig: Loaded properties from hadoop-metrics2.properties
2024-06-05 09:50:09,474 INFO impl.MetricsSystemImpl: Scheduled Metric snapshot period at 10 second (s) .
2024-06-05 09:50:09,475 INFO impl.MetricsSystemImpl: JobTracker metrics system started
2024-06-05 09:50:09,015 INFO input.FileInputFormat: Total input files to process : 3
2024-06-05 09:50:09,068 INFO mapreduce.JobSubmitter: number of splits: 3
2024-06-05 09:50:09,410 INFO mapreduce.JobSubmitter: Submitting tokens for job: job local305940441 0001
2024-06-05 09:50:09,410 INFO mapreduce.JobSubmitter: Executing with tokens: [J
2024-06-05 09:50:09,672 INFO mapreduce.Job: The url to track the job: http://localhost:8080/
2024-06-05 09:50:09,673 INFO mapreduce.Job: Running job: job local305940441 0001
2024-06-05 09:50:09,695 INFO mapred.LocalJobRunner: OutputCommitter set in config null
2024-06-05 09:50:09,716 INFO output.FileOutputCommitter: File Output Committer Algorithm version is 2
2024-06-05 09:50:09,716 INFO output.FileOutputCommitter: FileOutputCommitter skip cleanup temporary folders under output directory: false, ignore cleanup failure
2024-06-05 09:50:09,717 INFO mapred.LocalJobRunner: OutputCommitter is org.apache.hadoop.mapreduce.FileOutputCommitter
```

RESULTS

POST-JOB OPERATIONS

Post-Job Operations

1. View Output File:

- **Command:** `hadoop fs -cat /user/your_username/output/part-r-00000`
- **Purpose:** Displays the contents of the output file to review the results of the MapReduce job.

2. Stop Hadoop Services:

- **Command:** `stop-yarn.sh`
- **Purpose:** Stops the YARN ResourceManager and NodeManagers.
- **Command:** `stop-dfs.sh`
- **Purpose:** Stops the HDFS NameNode and DataNodes.

RESULTS

POST-JOB OPERATIONS

Partial Inverted Index Results:

- **"a"**: Appears in **Document 2**
- **"banana"**: Appears in **Document 2**
- **"is"**: Appears in **Documents 2, 1, 0**
- **"it"**: Appears in **Documents 0, 2, 1**
- **"what"**: Appears in **Documents 1, 0**

```
lbereket625@cloudshell:~/hadoop-3.3.4/bin (cs-570-big-data-424802)$ hdfs dfs -cat /tmp/outputfile/*
a        [2]
banana   [2]
is       [2, 1, 0]
it       [0, 2, 1]
what     [1, 0]
```

RESULTS

POST-JOB OPERATIONS

Convert to Full Inverted Index MapReduce Program

Create inverted_index.py File

- Ensure that the input files are available in the current directory.
- Implement the Full Inverted Index MapReduce logic in inverted_index.py.

```
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$ ls
file2.txt  file1.txt  file.txt  InvertedIndex.jar  InvertedIndex.java  inverted_index.py
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$
```

RESULTS

POST-JOB OPERATIONS

Convert to Full Inverted Index MapReduce Program

Create inverted_index.py File

- Ensure that the input files are available in the current directory.
- Implement the Full Inverted Index MapReduce logic in inverted_index.py.

```
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$ ls
file2.txt  file1.txt  file.txt  InvertedIndex.jar  InvertedIndex.java  inverted_index.py
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$
```

RESULTS

POST-JOB OPERATIONS

Convert to Full Inverted Index MapReduce Program

inverted_index.py File

- Implement the Full Inverted Index MapReduce logic in inverted_index.py.

```
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$ vim inverted_index.py
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$ cat inverted_index.py

from collections import defaultdict

# Map function
def map_function(file_index, line):
    words = line.strip().split()
    return [(word, (file_index, 1)) for i, word in enumerate(words)]

# Reduce function
def reduce_function(values):
    # Group and sort values
    return sorted(values, key=lambda x: (x[0], x[1]))

def main():
    # Input files
    input_files = {'file0': 0, 'file1': 1, 'file2': 2}
    intermediate_results = defaultdict(list)

    # Map Phase
    for file_name, index in input_files.items():
        with open(file_name, 'r') as file:
            for line in file:
                map_results = map_function(index, line)
                for word, value in map_results:
                    intermediate_results[word].append(value)

    # Reduce Phase
    inverted_index = {}
    for word, values in intermediate_results.items():
        inverted_index[word] = reduce_function(values)

    # Output results
    for word, indices in sorted(inverted_index.items()):
        print(f"{word}: {indices}")

if __name__ == "__main__":
    main()

lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$
```

RESULTS

POST-JOB OPERATIONS

Full Index Results

- **a**: [(2, 2)] - Appears in document 2, position 2.
- **banana**: [(2, 3)] - Appears in document 2, position 3.
- **is**: [(0, 1), (0, 4), (1, 1), (2, 1)] - Multiple positions across documents 0, 1, and 2.
- **it**: [(0, 0), (0, 3), (1, 2), (2, 0)] - Multiple positions across documents 0, 1, and 2.
- **what**: [(0, 2), (1, 0)] - Appears in documents 0 and 1.

```
lbereket625@cloudshell:~/hadoop-3.3.4/FullIndex (cs-570-big-data-424802)$ python3 inverted_index.py
a: [(2, 2)]
banana: [(2, 3)]
is: [(0, 1), (0, 4), (1, 1), (2, 1)]
it: [(0, 0), (0, 3), (1, 2), (2, 0)]
what: [(0, 2), (1, 0)]
```

05 ENHANCEMENTS

HOW CAN WE GET BETTER RESULTS ?

Enhanced Indexing

1. **Optimize Mapper and Reducer**
   - Refine the process of generating and processing key-value pairs for improved performance.
2. **Advanced Data Management**
   - Leverage optimized Hadoop configurations to efficiently handle larger datasets.

Enhanced Indexing: Refine the key-value pair generation and processing in the Mapper and Reducer for better performance. Utilize advanced Hadoop configurations to efficiently manage larger datasets

06 CONCLUSION

- **MapReduce** effectively handles and organizes extensive datasets.
- **Full Inverted Index** delivers comprehensive insights into word frequencies across multiple documents.

Learning Outcome:

- Acquired hands-on experience with Hadoop MapReduce and advanced data indexing methods.

07 REFERENCES

- https://github.com/imehrdadmahdavi/map-reduce-inverted-index
- https://taruntelang.medium.com/how-to-implement-inverted-index-data-structure-in-java-14067093acd4#:~:text=An%20inverted%20index%20is%20typically,column%20corresponds%20to%20a%20document.

- My Github Link for this Project: My_Github_Link