RK

Hadoop
K1 Group 10

Muhammad Najih Aflah - 2106653880
Mochammad Dyenta D - 2106731245
Rayhan Akbar Arrizky - 2106632655
Muhammad Fajri Alqomaril - 2106651635
Agung Firmansyah - 2006577454

RK

Apa Itu Hadoop

Hadoop adalah kerangka kerja (framework) open-source yang dirancang untuk memproses, menyimpan, dan menganalisis jumlah data besar (big data) secara terdistribusi. Dikembangkan oleh Apache Software Foundation, Hadoop menyediakan solusi untuk mengatasi tantangan dalam pengelolaan dan analisis data yang sangat besar yang tidak dapat ditangani oleh sistem tradisional.

Hadoop Ecosystem

oozie
(Work flow)

HCatalog
Table & schema Management

Pig
(Scripting)

Hive
(Sql Query)

mahout
(Machine Learning)

Drill
(Interactive Analysis)

AVRO
(JSON)

Thrift
(Cross Language Service)

HBASE
(Columnar Store)

Sqoop
(Data Collection)

Flume
(Data Collection)

Zookeeper
(Coordination)

Ambari
Apache Ambari
(Management & Monitoring)

Mapreduce
(Data Processing)

Yarn
(Cluster Resource Management)

HDFS
(Hadoop Distributed File system)

Komponen utama Hadoop

RK

HDFS (Hadoop Distributed File System)

HDFS adalah sistem file terdistribusi yang dirancang khusus untuk menyimpan data besar dalam cluster Hadoop.

Apache MapReduce

Apache MapReduce membagi tugas pemrosesan menjadi langkah "map" dan "reduce" untuk pemrosesan paralel di dalam cluster Hadoop.

YARN (Yet Another Resource Negotiator)

Apache YARN bertanggung jawab atas manajemen dan penjadwalan sumber daya di dalam cluster Hadoop, memungkinkan berbagai jenis aplikasi untuk berjalan secara bersamaan dan efisien.

Kelebihan dan Kekurangan

Kelebihan

1. Skalabilitas: Dapat diukur secara horizontal untuk menangani pertumbuhan data yang besar.
2. Pemrosesan Terdistribusi: Memungkinkan pemrosesan data secara paralel di seluruh cluster.
3. Toleransi Kesalahan: Mampu mengatasi kegagalan perangkat keras atau perangkat lunak tanpa kehilangan data.
4. Ketersediaan Data: Menggunakan replikasi data untuk memastikan ketersediaan data yang tinggi.
5. Ekosistem yang Kaya: Menyediakan berbagai komponen tambahan untuk fungsionalitas dan fleksibilitas yang lebih luas.

RK

Kekurangan

1. Kompleksitas: Membutuhkan pemahaman teknis yang baik dan pengelolaan cluster yang rumit.
2. Latensi: Penundaan dalam pemrosesan data real-time atau interaktif.
3. Overhead: Tingginya overhead untuk pemrosesan data yang lebih kecil.
4. Keterbatasan Analisis Kompleks: Mungkin tidak efisien untuk analisis kompleks dan iteratif.

Instalasi Hadoop

Langkah-langkah run Hadoop dengan Ubuntu

Konfigurasi Hadoop

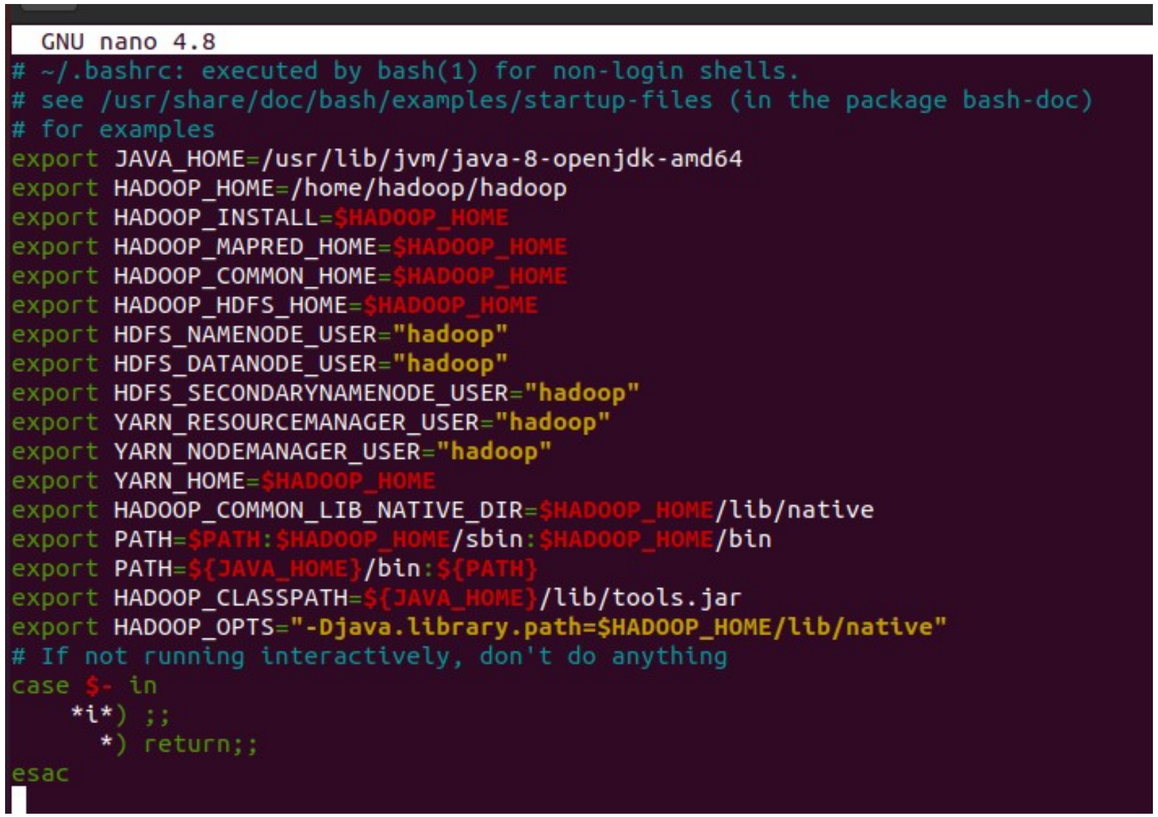
Konfigurasi Hadoop

Hadoop unggul ketika digunakan dalam mode terdistribusi penuh pada sekelompok besar server jaringan . Namun, jika Anda baru menggunakan Hadoop dan ingin menjelajahi perintah dasar atau menguji aplikasi, Anda dapat mengonfigurasi Hadoop di satu node.

Setup ini, juga disebut pseudo-distributed mode , memungkinkan setiap daemon Hadoop dijalankan sebagai satu proses Java. Hadoop environment dikonfigurasi dengan mengedit sekumpulan file konfigurasi yaitu :

- bashrc,
- hadoop-env.sh
- core-site.xml
- hdfs-site.xml
- mapred-site-xml
- yarn-site.xml

File bachrc



File bashrc merupakan sebuah file yang berguna untuk mengkonfigurasi variabel lingkungan. Kita dapat mengedit file bashrc dengan menjalankan command berikut :

$ sudo nano ~/.bashrc

Lalu menambahkan konfigurasi pada gambar, setelah itu kita dapat mengaktifkan variabel lingkungan dengan command :

$ source ~/.bashrc

File hadoop-env.sh

```
  GNU nano 4.8
# Generic settings for HADOOP
###

# Technically, the only required environment variable is JAVA_HOME.
# All others are optional.  However, the defaults are probably not
# preferred.  Many sites configure these options outside of Hadoop,
# such as in /etc/profile.d

# The java implementation to use. By default, this environment
# variable is REQUIRED on ALL platforms except OS X!
 export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
export HDFS_NAMENODE_USER="hadoop"
export HDFS_DATANODE_USER="hadoop"
export HDFS_SECONDARYNAMENODE_USER="hadoop"
export YARN_RESOURCEMANAGER_USER="hadoop"
export YARN_NODEMANAGER_USER="hadoop"
# Location of Hadoop.  By default, Hadoop will attempt to determine
# this location based upon its execution path.
# export HADOOP_HOME=

# Location of Hadoop's configuration information.  i.e., where this
# file is living. If this is not defined, Hadoop will attempt to
# locate it based upon its execution path.
#
# NOTE: It is recommend that this variable not be set here but in
# /etc/profile.d or equivalent.  Some options (such as
# --config) may react strangely otherwise.
#
# export HADOOP_CONF_DIR=${HADOOP_HOME}/etc/hadoop
```

File hadoop-env.sh berfungsi sebagai file master untuk mengonfigurasi pengaturan proyek terkait YARN, HDFS , MapReduce , dan Hadoop. Kita dapat mengedit file hadoop-env.sh dengan menjalankan command berikut :

sudo nano $HADOOP_HOME/etc/hadoop/hadoop-env.sh

Lalu menambahkan konfigurasi pada gambar.

File core-site.xml

```
  GNU nano 4.8
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>fs.defaultFS</name>
                <value>hdfs://localhost:9000</value>
        </property>
 </configuration>
```

File core-site.xml mendefinisikan properti inti HDFS dan Hadoop. Kita dapat mengedit file core-site.xml dengan menjalankan command berikut :

sudo nano $HADOOP_HOME/etc/hadoop/core-site.xml

Lalu menambahkan konfigurasi pada gambar untuk mengganti nilai default untuk direktori sementara dan tambahkan URL HDFS Anda untuk mengganti pengaturan sistem file lokal default:

File hdfs-site.xml

```
  GNU nano 4.8
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
        <property>
                <name>dfs.replication</name>
                <value>1</value>
        </property>
<!--
        <property>
                <name>dfs.name.dir</name>
                <value>file:///home/hadoop/hadoopdata/hdfs/namenode</value>
        </property>
-->
        <property>
                <name>dfs.data.dir</name>
                <value>file:///home/hadoop/hadoopdata/hdfs/datanode</value>
        </property>

</configuration>
```

Properti dalam file hdfs-site.xml mengatur lokasi untuk menyimpan metadata node, file fsimage, dan mengedit file log. Konfigurasikan file dengan menentukan direktori penyimpanan NameNode dan DataNode . Dalam file “hdfs-site.xml” ini, kita akan mengubah jalur direktori “datanode” dan “namenode”. Kita dapat mengedit file hdfs-site.xml dengan menjalankan command berikut :

sudo nano $HADOOP_HOME/etc/hadoop/hdfs-site.xml

Lalu menambahkan konfigurasi pada gambar

File mapred-site.xml

```
  GNU nano 4.8
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="configuration.xsl"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->

<!-- Put site-specific property overrides in this file. -->

<configuration>
    <property>
        <name>mapreduce.framework.name</name>
        <value>yarn</value>
    </property>
    <property>
        <name>mapreduce.application.classpath</name>
        <value>$HADOOP_MAPRED_HOME/share/hadoop/mapreduce/*:$HADOOP_MAPRED_HOME/sha
    </property>
</configuration>
```

Gunakan perintah berikut untuk mengakses file mapred-site.xml dan menentukan nilai MapReduce :

sudo nano $HADOOP_HOME/etc/hadoop/mapred-site.xml

Lalu ubah nilai nama framework MapReduce default menjadi yarn seperti konfigurasi pada gambar.

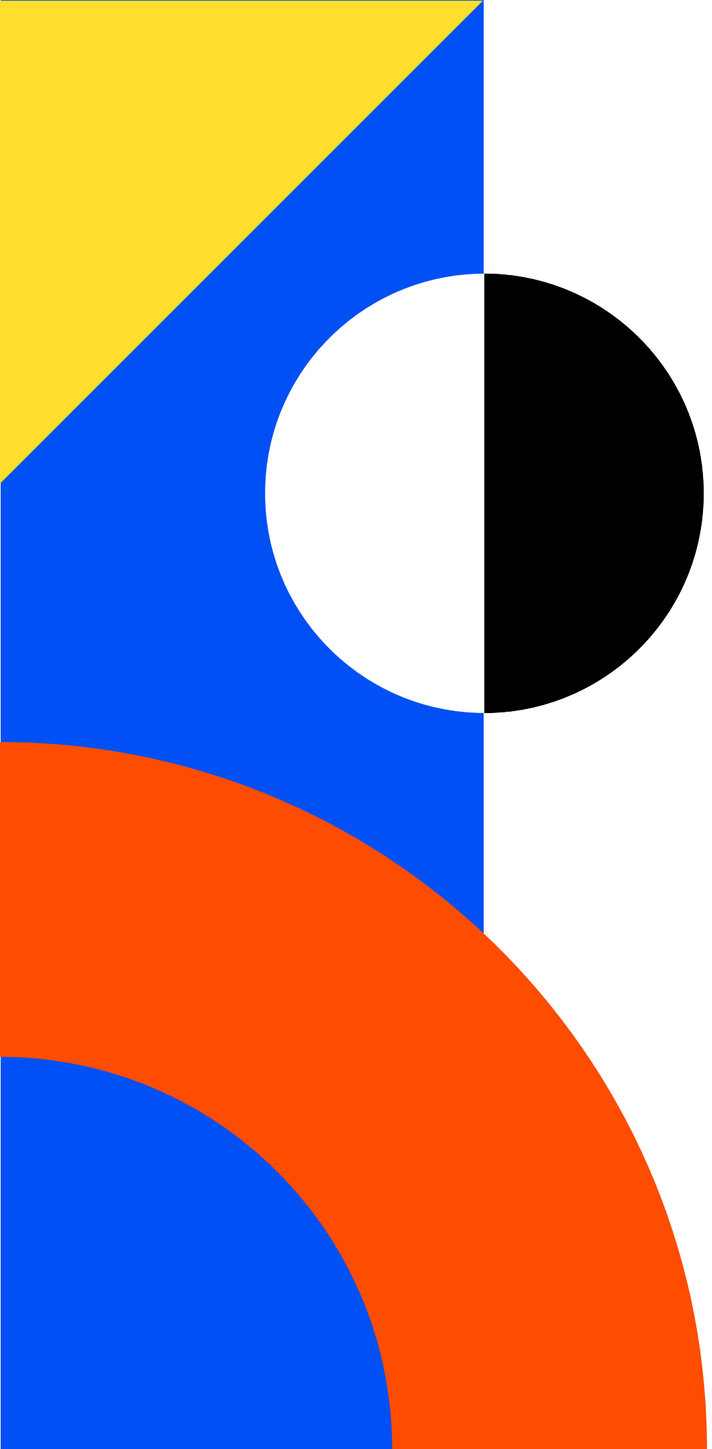
File yarn-site.xml

```
  GNU nano 4.8                                                yarn-site.xml
<?xml version="1.0"?>
<!--
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License. See accompanying LICENSE file.
-->


<configuration>
    <property>
        <name>yarn.nodemanager.aux-services</name>
        <value>mapreduce_shuffle</value>
    </property>
    <property>
        <name>yarn.nodemanager.env-whitelist</name>
        <value>JAVA_HOME,HADOOP_COMMON_HOME,HADOOP_HDFS_HOME,HADOOP_CONF_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_HOME,PATH,LANG,TZ,HADOOP_MAPRED_HOME</value>
    </property>
    <property>
        <name>yarn.nodemanager.disk-health-checker.max-disk-utilization-per-disk-percentage</name>
        <value>98.5</value>
    </property>
    <property>
        <name>yarn.nodemanager.resource.memory-mb</name>
        <value>32000</value>
    </property>

</configuration>
```

File yarn-site.xml digunakan untuk menentukan pengaturan yang relevan dengan YARN . Ini berisi konfigurasi untuk Node Manager, Resource Manager, Containers, dan Application Master. Kita dapat mengedit file yarn-site.xml dengan menjalankan command berikut :

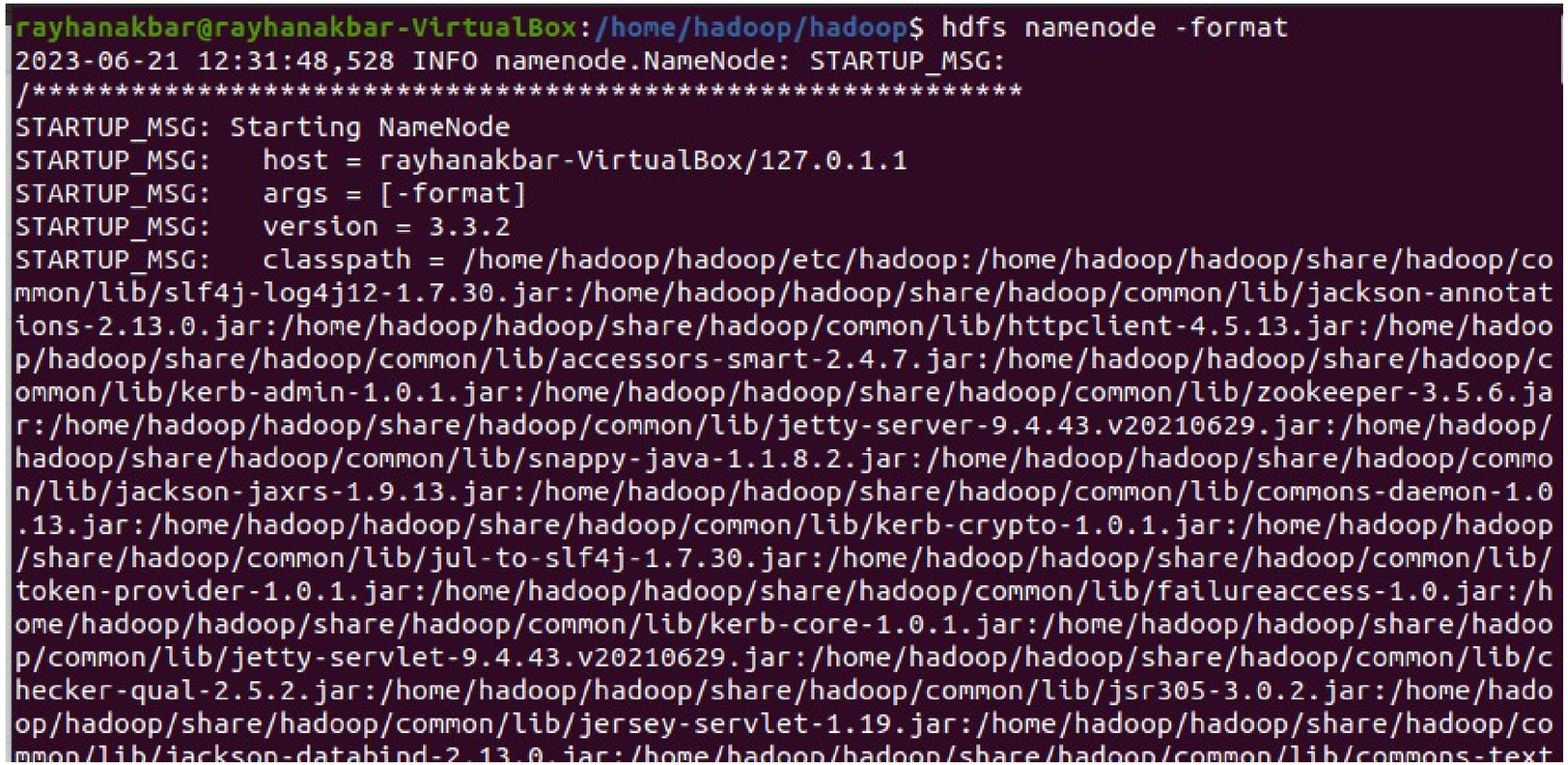
sudo nano $HADOOP_HOME/etc/hadoop/yarn-site.xml

Lalu menambahkan konfigurasi pada gambar.



Start Hadoop

Format HDFS NameNode



Penting untuk memformat NameNode sebelum memulai layanan Hadoop untuk pertama kalinya. Langkah ini penting untuk menginisialisasi metadata dan struktur data yang diperlukan oleh Hadoop Distributed File System (HDFS). lakukan dengan command :

hdfs namenode -format

Mulai Hadoop Cluster

```
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop/hadoop$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [rayhanakbar-VirtualBox]
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop/hadoop$ start-yarn.sh
Starting resourcemanager
Starting nodemanagers
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop/hadoop$ jps
10482 SecondaryNameNode
10691 ResourceManager
10083 NameNode
10266 DataNode
11195 Jps
10845 NodeManager
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop/hadoop$
```

Mulai Hadoop cluster dengan menjalankan kedua command berikut :

$ start-dfs.sh
$ start-yarn.sh

Kita dapat memverifikasi semua komponen yang berjalan dengan command :

$jps

Menjalankan Wordcount

Kode Wordcount

WordCount.java /home/.../bin × WordCount.java ./

home > hadoop > hadoop > bin > WordCount.java

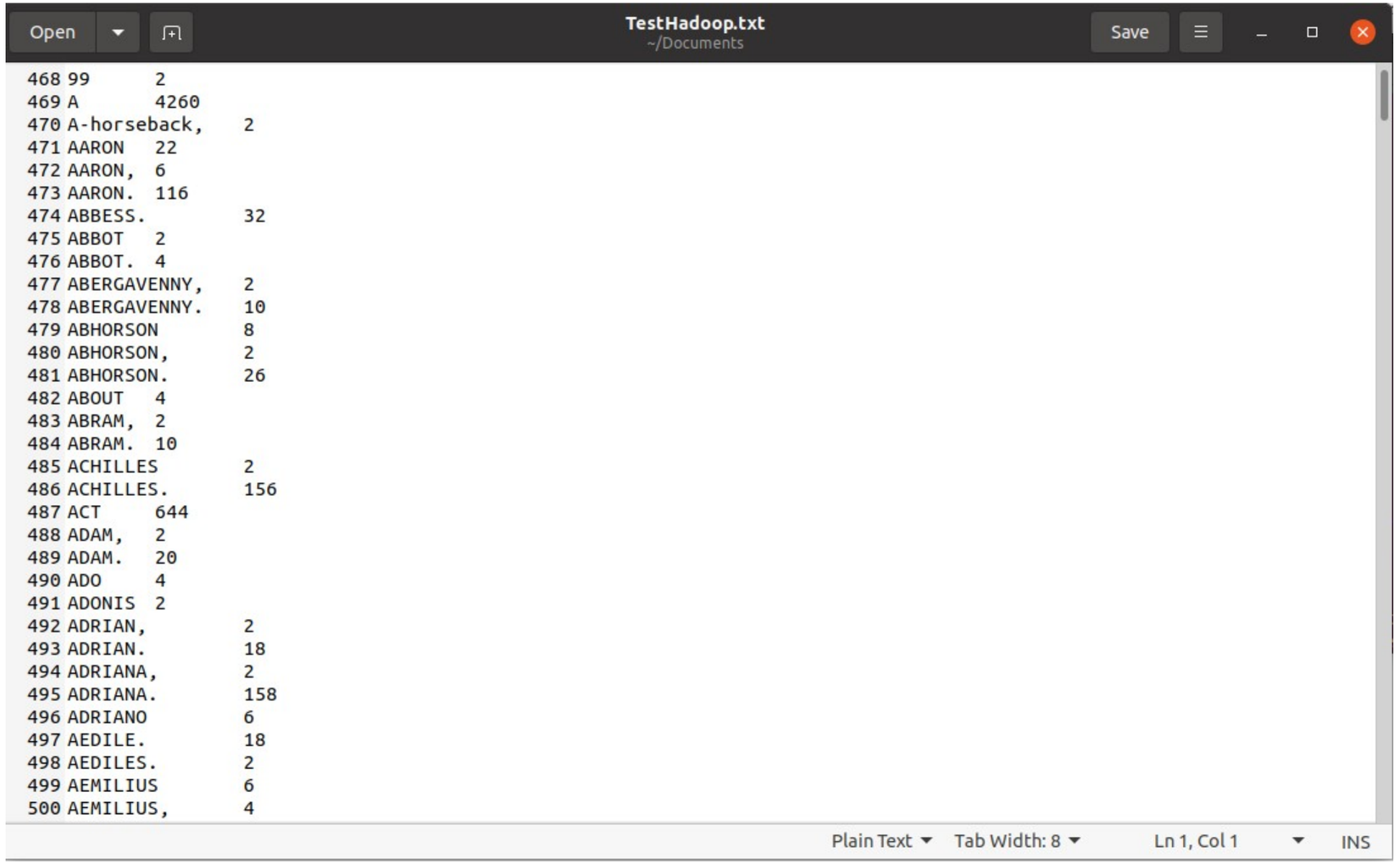
```java
public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Menjalankan Wordcount dengan Hadoop

```
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop/hadoop$ hdfs dfs -mkdir /WordCountInput
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop/hadoop$ hdfs dfs -put /home/rayhanakbar/Downloads/Dataset/text10
text100MB.txt  text10MB.txt
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop/hadoop$ hdfs dfs -put /home/rayhanakbar/Downloads/Dataset/text10MB.txt /WordCountInput
```

```
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop/hadoop/bin$ hadoop jar wc.jar WordCount /WordCountInput/text10MB.txt /text10MB_output
2023-06-21 12:47:00,300 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2023-06-21 12:47:00,535 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute you
2023-06-21 12:47:00,554 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/rayhanakbar/.staging/job_168732585
2023-06-21 12:47:00,725 INFO input.FileInputFormat: Total input files to process : 1
2023-06-21 12:47:01,189 INFO mapreduce.JobSubmitter: number of splits:1
2023-06-21 12:47:01,297 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1687325899166_0001
2023-06-21 12:47:01,297 INFO mapreduce.JobSubmitter: Executing with tokens: []
2023-06-21 12:47:01,464 INFO conf.Configuration: resource-types.xml not found
2023-06-21 12:47:01,464 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2023-06-21 12:47:01,687 INFO impl.YarnClientImpl: Submitted application application_1687325899166_0001
2023-06-21 12:47:01,728 INFO mapreduce.Job: The url to track the job: http://rayhanakbar-VirtualBox:8088/proxy/application_1687325899166_0001/
2023-06-21 12:47:01,729 INFO mapreduce.Job: Running job: job_1687325899166_0001
2023-06-21 12:47:07,798 INFO mapreduce.Job: Job job_1687325899166_0001 running in uber mode : false
2023-06-21 12:47:07,799 INFO mapreduce.Job:  map 0% reduce 0%
2023-06-21 12:47:12,866 INFO mapreduce.Job:  map 100% reduce 0%
2023-06-21 12:47:16,888 INFO mapreduce.Job:  map 100% reduce 100%
2023-06-21 12:47:17,902 INFO mapreduce.Job: Job job_1687325899166_0001 completed successfully
2023-06-21 12:47:17,963 INFO mapreduce.Job: Counters: 54
        File System Counters
                FILE: Number of bytes read=983264
                FILE: Number of bytes written=2516911
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=10000114
                HDFS: Number of bytes written=732767
                HDFS: Number of read operations=8
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
                HDFS: Number of bytes read erasure-coded=0
        Job Counters
                Launched map tasks=1
                Launched reduce tasks=1
                Data-local map tasks=1
                Total time spent by all maps in occupied slots (ms)=3381
                Total time spent by all reduces in occupied slots (ms)=1453
                Total time spent by all map tasks (ms)=3381
                Total time spent by all reduce tasks (ms)=1453
                Total vcore-milliseconds taken by all map tasks=3381
                Total vcore-milliseconds taken by all reduce tasks=1453
                Total megabyte-milliseconds taken by all map tasks=3462144
                Total megabyte-milliseconds taken by all reduce tasks=1487872
```

Hasil Wordcount



WORD COUNT

-With or without Hadoop comparison

System Specification

- **OS**

  Ubuntu 20.04 (Virtual Machine)

- **CPU allocated**

  6

- **Base Memory allocated**

  8 GB

Program with hadoop

WordCount.java /home/.../bin × WordCount.java ./

home > hadoop > hadoop > bin > WordCount.java

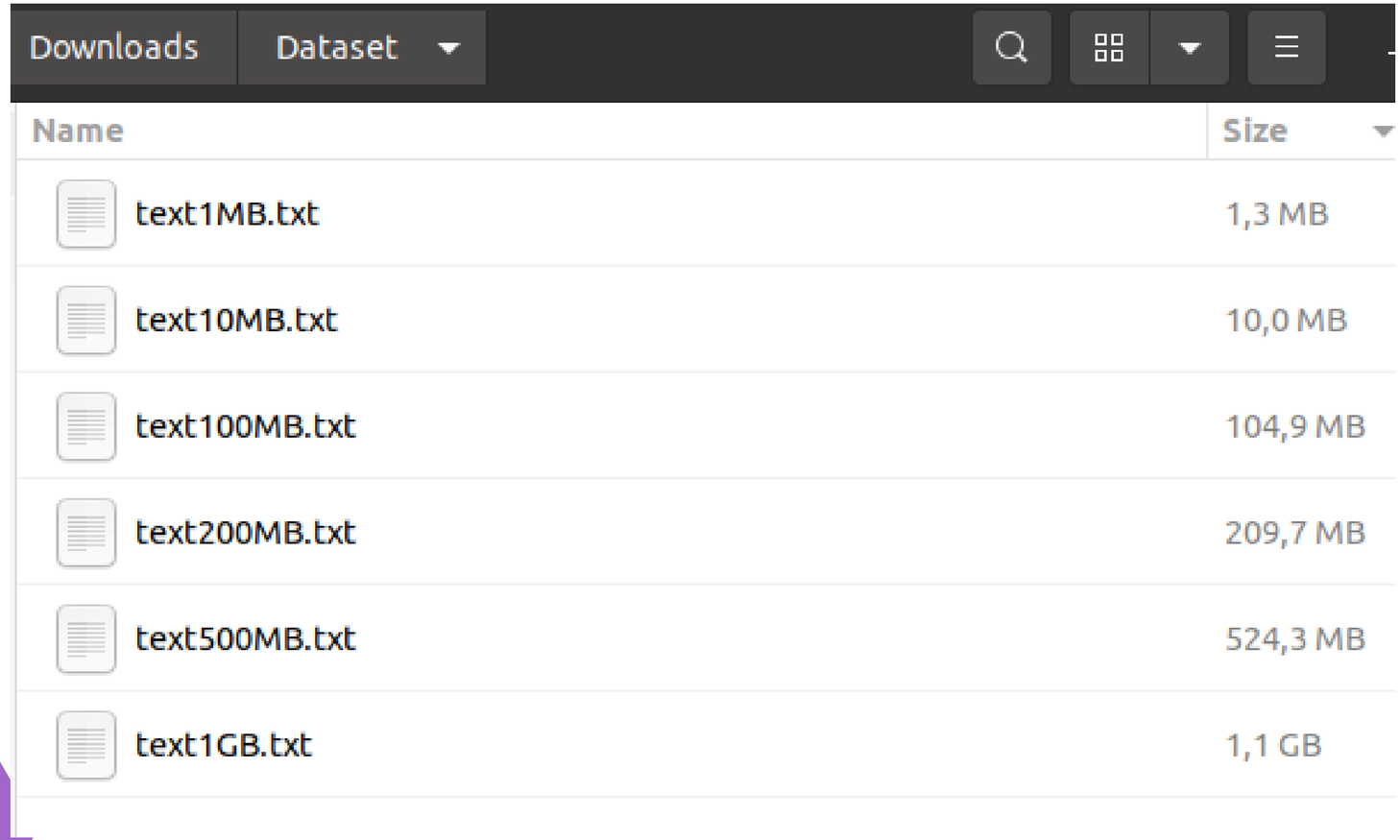
```java
public class WordCount {

  public static class TokenizerMapper
       extends Mapper<Object, Text, Text, IntWritable>{

    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
      StringTokenizer itr = new StringTokenizer(value.toString());
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        context.write(word, one);
      }
    }
  }

  public static class IntSumReducer
       extends Reducer<Text,IntWritable,Text,IntWritable> {
    private IntWritable result = new IntWritable();

    public void reduce(Text key, Iterable<IntWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {
      int sum = 0;
      for (IntWritable val : values) {
        sum += val.get();
      }
      result.set(sum);
      context.write(key, result);
    }
  }

  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "word count");
    job.setJarByClass(WordCount.class);
    job.setMapperClass(TokenizerMapper.class);
    job.setCombinerClass(IntSumReducer.class);
    job.setReducerClass(IntSumReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

Program without hadoop

WordCount.java

```java
public class WordCount {
    public static void main(String[] args) {
        String inputFile = "text500MB.txt"; // Path to the input file
        try {
            Map<String, Integer> wordCountMap = countWords(inputFile);
            // Print the word count results
            for (Map.Entry<String, Integer> entry : wordCountMap.entrySet()) {
                System.out.println(entry.getKey() + ": " + entry.getValue());
            }
        } catch (IOException e) {
            System.err.println("Error reading the input file: " + e.getMessage());
        }

    }
    public static Map<String, Integer> countWords(String inputFile) throws IOException {
        Map<String, Integer> wordCountMap = new HashMap<>();
        BufferedReader reader = new BufferedReader(new FileReader(inputFile));
        String line;
        // Read each line from the input file
        while ((line = reader.readLine()) != null) {
            // Split the line into words using whitespace as the delimiter
            String[] words = line.split("\\s+");
            // Count the occurrences of each word
            for (String word : words) {
                if (wordCountMap.containsKey(word)) {
                    // Increment the count if the word is already present
                    wordCountMap.put(word, wordCountMap.get(word) + 1);
                } else {
                    // Initialize the count to 1 for new words
                    wordCountMap.put(word, 1);
                }
            }
        }
        reader.close();
        return wordCountMap;
    }
}
```

Test Sample


Downloads
Dataset
Name
Size
text1MB.txt
1,3 MB
text10MB.txt
10,0 MB
text100MB.txt
104,9 MB
text200MB.txt
209,7 MB
text500MB.txt
524,3 MB
text1GB.txt
1,1 GB

Hadoop Setup

```
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop$ hdfs dfs -mkdir /WordCountInput
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop$ hdfs dfs -put /home/rayhanakbar/Downloads/Dataset/text1MB.txt /WordCountInput
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop$ hdfs dfs -put /home/rayhanakbar/Downloads/Dataset/text10MB.txt /WordCountInput
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop$ hdfs dfs -put /home/rayhanakbar/Downloads/Dataset/text100MB.txt /WordCountInput
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop$ hdfs dfs -put /home/rayhanakbar/Downloads/Dataset/text200MB.txt /WordCountInput
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop$ hdfs dfs -put /home/rayhanakbar/Downloads/Dataset/text500MB.txt /WordCountInput
rayhanakbar@rayhanakbar-VirtualBox:/home/hadoop$ hdfs dfs -put /home/rayhanakbar/Downloads/Dataset/text1GB.txt /WordCountInput
```

Browse Directory

/WordCountInput Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	rayhanakbar	supergroup	100 MB	Jun 09 16:38	1	128 MB	text100MB.txt
-rw-r--r--	rayhanakbar	supergroup	9.54 MB	Jun 09 16:37	1	128 MB	text10MB.txt
-rw-r--r--	rayhanakbar	supergroup	1 GB	Jun 09 16:38	1	128 MB	text1GB.txt
-rw-r--r--	rayhanakbar	supergroup	1.2 MB	Jun 09 16:37	1	128 MB	text1MB.txt
-rw-r--r--	rayhanakbar	supergroup	200 MB	Jun 09 16:38	1	128 MB	text200MB.txt
-rw-r--r--	rayhanakbar	supergroup	500 MB	Jun 09 16:38	1	128 MB	text500MB.txt

Showing 1 to 6 of 6 entries Previous 1 Next

Hadoop, 2022

File 1MB

With Hadoop

Command:

hadoop jar wc.jar WordCount
/WordCountInput/text1MB.txt /text1MB_output

User:	rayhanakbar
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Fri Jun 09 16:41:56 +0700 2023
Launched:	Fri Jun 09 16:41:57 +0700 2023
Finished:	Fri Jun 09 16:42:08 +0700 2023
Elapsed:	11sec

Java (Without Hadoop)

Command:

time java WordCount

```
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ time java WordCount
: 3438
spilling: 2
passionateness: 1
shrink?: 1
animosity;: 1
pretend: 7
Gros: 1
huzza!”: 1
him,’: 1
him,”: 3
mankind—devilish: 1
blast;: 1
capacity.: 2
coals: 2
halyards: 1
```

```
real    0m0,334s
user    0m0,422s
sys     0m0,197s
```

File 10MB

With Hadoop

Command:
hadoop jar wc.jar WordCount
/WordCountInput/text10MB.txt
/text10MB_output

User:	rayhanakbar
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Fri Jun 09 17:43:25 +0700 2023
Launched:	Fri Jun 09 17:43:26 +0700 2023
Finished:	Fri Jun 09 17:43:37 +0700 2023
Elapsed:	12sec
Tracking URL:	History

Java (Without Hadoop)

Command:
time java WordCount

```
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ time java WordCount
: 110633
Reserve: 6
frowning: 24
favour!: 4
spilling: 2
DUKE.: 548
Tisick,: 2
DUKE,: 14
abrupt: 2
favour-: 2
favour.: 36
TAURUS.: 2
clapper;: 2
TAURUS,: 2
favour,: 52
sithence,: 2
Grow: 12
real    0m0,836s
user    0m0,996s
sys     0m0,308s
```

File 100MB

With Hadoop

Command:
hadoop jar wc.jar WordCount
/WordCountInput/text100MB.txt
/text100MB_output

User:	rayhanakbar
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Fri Jun 09 17:44:58 +0700 2023
Launched:	Fri Jun 09 17:44:59 +0700 2023
Finished:	Fri Jun 09 17:45:24 +0700 2023
Elapsed:	25sec

Java (Without Hadoop)

Command:
time java WordCount

```
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ javac WordCount.java
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ time java WordCount
```

```
real    0m6,116s
user    0m5,194s
sys     0m1,674s
```

File 200MB

With Hadoop

Command:
hadoop jar wc.jar WordCount
/WordCountInput/text200MB.txt
/text200MB_output

User:	rayhanakbar
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Fri Jun 09 17:47:16 +0700 2023
Launched:	Fri Jun 09 17:47:17 +0700 2023
Finished:	Fri Jun 09 17:47:46 +0700 2023
Elapsed:	30sec

Java (Without Hadoop)

Command:
time java WordCount

```
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ javac WordCount.java
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ time java WordCount
```

```
real    0m10,416s
user    0m8,669s
sys     0m2,257s
```

File 500MB

With Hadoop

Command:
time hadoop jar wc.jar WordCount
/WordCountInput/text500MB.txt
/text500MB_output

User:	rayhanakbar
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Fri Jun 09 19:05:44 +0700 2023
Launched:	Fri Jun 09 19:05:45 +0700 2023
Finished:	Fri Jun 09 19:06:19 +0700 2023
Elapsed:	34sec

Java (Without Hadoop)

Command:
time java WordCount

```
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ javac WordCount.java
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ time java WordCount
```

```
real    0m22,642s
user    0m21,818s
sys     0m2,092s
```

File 1GB

With Hadoop

Command:
hadoop jar wc.jar WordCount
/WordCountInput/text1GB.txt
/text1GB_output

User:	rayhanakbar
Name:	word count
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Fri Jun 09 19:27:33 +0700 2023
Launched:	Fri Jun 09 19:27:34 +0700 2023
Finished:	Fri Jun 09 19:28:27 +0700 2023
Elapsed:	53sec

Java (Without Hadoop)

Command:
time java WordCount

```
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ javac WordCount.java
rayhanakbar@rayhanakbar-VirtualBox:~/Downloads/Dataset$ time java WordCount
```

```
real    0m57,493s
user    0m3,146s
sys     0m0,267s
```

Graph

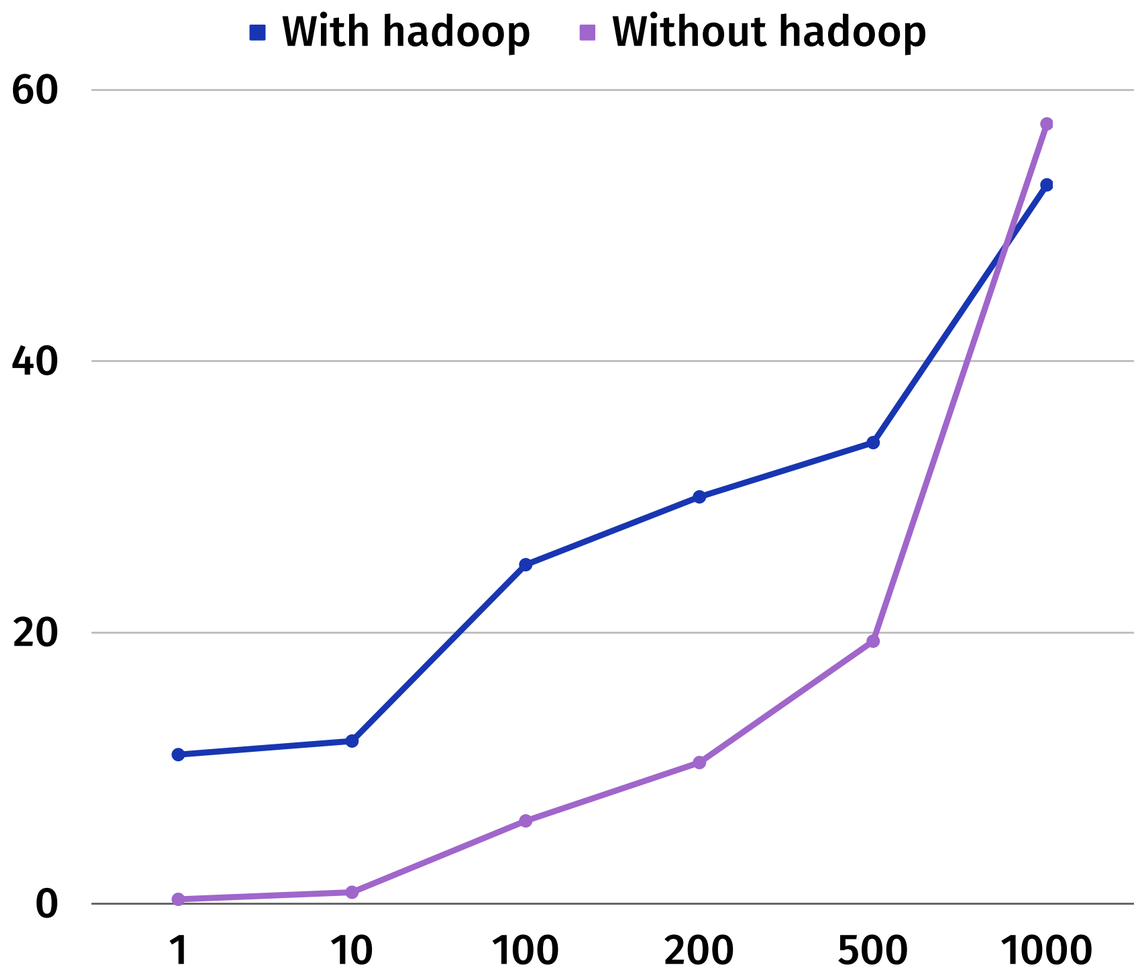
Analisis:

Pada ukuran file yang masih kecil mulai dari 1-500MB, Word Count tanpa hadoop terlihat lebih cepat dibanding word count dengan hadoop.

Namun berdasarkan grafik yang dihasilkan, dapat dilihat bahwa Word Count tanpa hadoop memiliki peningkatan waktu yang lebih signifikan dibandingkan Word Count dengan hadoop.

Kesimpulan:

Pada ukuran file yang besar, word count dengan hadoop lebih unggul dibandingkan word count tanpa hadoop



Thank you!