

OPERASI MAPREDUCE

MENGGUNAKAN HADOOP CLUSTER



**Disusun untuk memenuhi Tugas pada mata kuliah Komputasi Tersebar Paralel yang
diampu oleh Bapak Panji Wisnu Wirawan, ST, MT**

DISUSUN OLEH :

- | | |
|--------------------------------|-----------------------|
| 1. Bernardinus Hendra N | 24060118130107 |
| 2. Aziiza Yvellicchia | 24060118130104 |
| 3. Fauzia Nur Fitria | 24060118130118 |

**PROGRAM STUDI S1-INFORMATIKA
FAKULTAS SAINS DAN MATEMATIKA
UNIVERSITAS DIPONEGORO
SEMARANG**

2020

A. Langkah-Langkah membuat *Cluster Hadoop*

Untuk membuat *Cluster Hadoop* pada OS Linux, dilakukan beberapa langkah sebagai berikut :

1. Pada Linux, pertama Java harus sudah ter-*install* terlebih dahulu.
2. *Install* ssh melalui terminal dengan cara seperti ini:
\$ sudo apt-get install ssh
\$ sudo apt-get install pdsh
3. *Download* Hadoop distribution di <http://www.apache.org/dyn/closer.cgi/hadoop/common/>, kemudian melakukan ekstraksi dan menambahkan beberapa konfigurasi tambahan di *file* etc/hadoop/hadoop-env.sh, sebagai berikut:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-1386/jre  
  
export HDFS_NAMENODE_USER=root  
  
export HDFS_DATANODE_USER=root  
  
export HDFS_SECONDARYNAMENODE_USER=root  
  
export PDSH_RCMD_TYPE=ssh  
  
export PDSH_SSH_ARGS_APPEND="${HADOOP_SSH_OPTS}" pdsh \  
  
export YARN_RESOURCEMANAGER_USER=root  
  
export YARN_NODEMANAGER_USER=root
```

4. Dalam melakukan eksekusi MapReduce digunakan dalam mode pseudo-distributed. Dimana setiap Hadoop daemon dapat dijalankan pada proses Java yang terpisah. Untuk konfigurasi dilakukan hal sebagai berikut:

etc/hadoop/core-site.xml:

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

etc/hadoop/hdfs-site.xml:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>1</value>
  </property>
</configuration>
```

etc/hadoop/mapred-site.xml:

```
<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/share/hadoop/mapreduce/lib/*</value>
  </property>
</configuration>
```

etc/hadoop/yarn-site.xml:

```
<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_CON
F_DIR,CLASSPATH_PREPEND_DISTCACHE,HADOOP_YARN_HOME,HADOOP_MAPRED_HOM
E</value>
  </property>
</configuration>
```

5. Melakukan *setup* passphraseless ssh. Dilakukan *command* berikut:

```
$ ssh-keygen -t rsa -P '' -f ~/.ssh/id_rsa
$ cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

```
$ chmod 0600 ~/.ssh/authorized_keys
```

B. Pembentukan Dataset dan Program

Dataset teks diambil dari beberapa *e-book* dari web <http://www.gutenberg.org/>. Beberapa teks yang diambil diantaranya adalah:

1. Travellers Stories by Eliza Lee Cabot Follen (<https://dev.gutenberg.org/ebooks/4030>)
2. Mastery of Self for Wealth, Power, Success by Frank C. Haddock (<https://dev.gutenberg.org/ebooks/4286>)
3. Classic Myths by Mary Catherine Judd (<https://dev.gutenberg.org/ebooks/9855>)

Teks tersebut di *copy* berkali-kali sehingga mendapatkan file teks dengan ekstensi .txt sebesar 10mb, 20mb, dan 30mb.

Untuk program MapReducernya, digunakan script sebagai berikut:

```
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
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.output.FileOutputFormat;
public class WordCount {
    // Map function
    public static class MyMapper extends Mapper<LongWritable, Text, Text,
IntWritable>{
        private Text word = new Text();
        public void map(LongWritable key, Text value, Context context)
            throws IOException, InterruptedException {
            // Splitting the line on spaces
            String[] stringArr = value.toString().split("\\s+");
            for (String str : stringArr) {
                word.set(str);
                context.write(word, new IntWritable(1));
            }
        }
    }

    // Reduce function
    public static class MyReducer 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, "WC");
        job.setJarByClass(WordCount.class);
        job.setMapperClass(MyMapper.class);
        job.setReducerClass(MyReducer.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 tersebut terbagi menjadi tiga bagian utama, yakni MyMapper yang merupakan fungsi Map, MyReducer yang merupakan fungsi Reduce, serta main yang merupakan driver utama aplikasi. Fungsi Map berjalan dengan memecah/split baris menjadi token-token setiap spasi baru ditemukan. Sedang pada reducer, dilakukan penjumlahan (*counting*) untuk kata-kata/token-token yang sama. Pada driver utama dilakukan pengaturan nama pekerjaan, pengaturan *mapper* dan *reducernya*, serta pengaturan I/O nya. Ketika kelas WordCount dieksekusi, maka akan langsung memanggil fungsi MyMapper dan MyReducer tersebut.

Setelah program ditulis, selanjutnya perlu di compile terlebih dahulu sebelum dapat dieksekusi. Untuk melakukan *compile*, diberikan beberapa perintah berikut pada terminal:

```

$ export HADOOP_HOME=/home/hendranata/hadoop

$ export HADOOP_CLASSPATH=$(HADOOP_HOME/bin/hadoop classpath)

$ sudo mkdir WordCountCompiled

$ sudo chmod -R 777 WordCountCompiled

$ javac -classpath $HADOOP_CLASSPATH -d WordCountCompiled/ WordCount.java

$ jar -cvf WordCount.jar -C WordCountCompiled/ .

```

Masing-masing perintah di atas, adalah untuk mengatur variabel `HADOOP_HOME` dan `HADOOP_CLASSPATH` (untuk proses *compile* kode java nantinya), pembuatan folder `WordCountCompiled`, pemberian hak akses penuh ke folder tersebut, lalu melakukan *compile* file `WordCount.java` menggunakan *classpath* yang telah diatur sebelumnya dan diarahkan ke folder yang telah kita buat, lalu mengemas folder tersebut menjadi *Java Archive* (jar) yang siap dieksekusi/dijalankan.

C. Cara Eksekusi MapReduce

Kami melakukan eksekusi untuk MapReduce pada WordCount menggunakan OS Linux distribusi Ubuntu. Cara-cara yang dilakukan adalah sebagai berikut :

1. Pertama, membuat format dari sistem file untuk membuat Hadoop baru dari HDFS.

```
$ bin/hdfs namenode -format
```
2. Kemudian memulai NameNode daemon dan DataNode daemon, serta Yarn Resource manager dan Node Manager, dengan mengeksekusi perintah berikut menggunakan *elevated (root) user access*.

```
$ sbin/start-dfs.sh  
$ sbin/start-yarn.sh
```
3. Setelah berhasil dijalankan web *interface* untuk NameNode, *default* web nya adalah :
<http://localhost:9870/>
dan untuk Resource Manager dapat diakses pada:
<http://localhost:8088/>
4. Setelah itu, membuat direktori HDFS yang diperlukan, untuk mengeksekusi MapReduce.

```
$ bin/hdfs dfs -mkdir /user  
$ bin/hdfs dfs -mkdir /user/root #karena perintah nanti akan  
dieksekusi menggunakan elevated  
(root) access.
```
5. Setelah itu, melakukan *copy* pada *file input* ke sistem *file* yang terdistribusi.

```
$ bin/hdfs dfs -mkdir input  
$ bin/hdfs dfs -put text/* input
```
6. Kemudian jalankan program MapReduce WordCount dengan memanggil:

```
$ bin/hadoop jar WordCount.jar WordCount input/text.txt output-result  
$ bin/hadoop jar WordCount.jar WordCount input/text20.txt output20
```

```
$ bin/hadoop jar WordCount.jar WordCount input/text30.txt output30
```

7. Langkah terakhir memeriksa *output* yang dihasilkan dengan cara melakukan *copy file* dari sistem *file* yang terdistribusi ke dalam sistem *file* lokal lalu dilakukan pengecekan terhadap *file* tersebut :

```
$ bin/hdfs dfs -get output-result output
$ bin/hdfs dfs -get output20 output20
$ bin/hdfs dfs -get output30 output30
$ cat output/*
```

8. Jika sudah selesai, maka berhentikan *daemon* dengan *command* berikut :

```
$ sbin/stop-yarn.sh
$ sbin/stop-dfs.sh
```

D. Hasil Eksekusi

a. File text.txt

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus
application_1609247267357_0005	root	WC	MAPREDUCE	default	0	Tue Dec 29 21:31:19 +0700 2020	Tue Dec 29 21:31:20 +0700 2020	Tue Dec 29 21:32:02 +0700 2020	FINISHED	SUCCEEDED

User: <u>root</u>	
Name: WC	
Application Type: MAPREDUCE	
Application Tags:	
Application Priority: 0 (Higher Integer value indicates higher priority)	
YarnApplicationState: FINISHED	
Queue: <u>default</u>	
FinalStatus Reported by AM: SUCCEEDED	
Started: Tue Dec 29 21:31:19 +0700 2020	
Launched: Tue Dec 29 21:31:20 +0700 2020	
Finished: Tue Dec 29 21:32:02 +0700 2020	
Elapsed: 43sec	
Tracking URL: <u>History</u>	
Log Aggregation Status: DISABLED	
Application Timeout (Remaining Time): Unlimited	
Diagnostics:	
Unmanaged Application: false	
Application Node Label expression: <Not set>	
AM container Node Label expression: <DEFAULT_PARTITION>	

Eksekusi berlangsung selama 43 detik, dengan hasil sebagai berikut:

```
part-r-00000(1)
File Edit Search Options Help
75286
""Adonis! 50
""Dare 50
""Give 100
""Here 50
""How 50
""I 50
""Is 50
""It 100
""My 50
""Now 50
""O, 50
""The 50
""What 50
""Where 50
""Whoever 50
""You 50
"A 105
"AS-IS". 5
"Aeneas 50
"Aeneas, 50
"After 50
"Age 50
"All200
"An6
"And 355
```

b. File text20.txt

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus
application_1609256445310_0002	root	WC	MAPREDUCE	default	0	Tue Dec 29 22:50:30 +0700 2020	Tue Dec 29 22:50:31 +0700 2020	Tue Dec 29 22:51:19 +0700 2020	FINISHED	SUCCEEDED

User:	<u>root</u>
Name:	WC
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	<u>default</u>
FinalStatus Reported by AM:	SUCCEEDED
Started:	Tue Dec 29 22:50:30 +0700 2020
Launched:	Tue Dec 29 22:50:31 +0700 2020
Finished:	Tue Dec 29 22:51:19 +0700 2020
Elapsed:	48sec
Tracking URL:	<u>History</u>
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

Program berhasil melakukan mapreduce dengan durasi sebesar 48 detik, dan hasil WordCount nya adalah sebagai berikut:

```

*part-r-00000(2)
File Edit Search Options Help
150572
""Adonis! 100
""Dare 100
""Give 200
""Here 100
""How 100
""I 100
""Is 100
""It 200
""My 100
""Now 100
""O, 100
""The 100
""What 100
""Where 100
""Whoever 100
""You 100
"A 210
"AS-IS". 10
"Aeneas 100
"Aeneas, 100
"After 100
"Age 100
"All400
"An12
"And 710
"Anti-poke-your-nose-into-other-folks'-business 12

```

c. File text30.txt

ID	User	Name	Application Type	Queue	Application Priority	StartTime	LaunchTime	FinishTime	State	FinalStatus
application_1609256445310_0003	root	WC	MAPREDUCE	default	0	Tue Dec 29 22:54:33 +0700 2020	Tue Dec 29 22:54:33 +0700 2020	Tue Dec 29 22:55:36 +0700 2020	FINISHED	SUCCEEDED

User:	root
Name:	WC
Application Type:	MAPREDUCE
Application Tags:	
Application Priority:	0 (Higher Integer value indicates higher priority)
YarnApplicationState:	FINISHED
Queue:	default
FinalStatus Reported by AM:	SUCCEEDED
Started:	Tue Dec 29 22:54:33 +0700 2020
Launched:	Tue Dec 29 22:54:33 +0700 2020
Finished:	Tue Dec 29 22:55:36 +0700 2020
Elapsed:	1mins, 3sec
Tracking URL:	History
Log Aggregation Status:	DISABLED
Application Timeout (Remaining Time):	Unlimited
Diagnostics:	
Unmanaged Application:	false
Application Node Label expression:	<Not set>
AM container Node Label expression:	<DEFAULT_PARTITION>

Eksekusi diselesaikan dalam waktu 1 menit 3 detik, dengan hasil WordCount sebagai berikut:

```

part-r-00000(3)
File Edit Search Options Help
225858
""Adonis! 150
""Dare 150
""Give 300
""Here 150
""How 150
""I 150
""Is 150
""It 300
""My 150
""Now 150
""O, 150
""The 150
""What 150
""Where 150
""Whoever 150
""You 150
"A 315
"AS-IS". 15
"Aeneas 150
"Aeneas, 150
"After 150
"Age 150
"All600
"An18
"And 1065
"Anti-poke-your-nose-into-other-folks'-business 18

```

Dari hasil eksekusi tiga dataset di atas, dapat diketahui bahwa MapReduce dapat melakukan fungsinya dengan durasi waktu yang cukup singkat. Pertambahan ukuran (dan jumlah data yang harus diproses) ternyata memiliki pengaruh yang tidak terlalu besar; terbukti dengan penambahan durasi pengerjaan yang berubah bertambah 5 detik (dari data 10mb ke 20mb: 43 dan 48 detik), serta bertambah 15 detik (dari 20mb ke 30mb: 48 detik dan 63 detik).

Perhitungan jumlah kata pada program WordCount ini masih menyisakan residu berupa tanda petik, titik, koma, tanda hubung (-), dan sebagainya, karena fungsi *splitting* yang digunakan adalah berdasar spasi, sehingga apabila terdapat tanda baca lain selain spasi, akan tetap dihitung sebagai suatu entri/token baru ('Also,' dengan koma dan 'Also' tanpa koma akan dihitung sebagai token yang berbeda, dan di hitung dengan jumlah masing-masing).