

Лабораторная работа №3. Кокарев Д. В. РИМ-201211

Задание 0

1. Открыть презентацию по HIVE и повторить команды со слайдов про Mapreduce\YARN.

```
export YARN_EXAMPLES=${HADOOP_HOME}/share/hadoop/mapreduce
```

```
yarn jar ${YARN_EXAMPLES}/hadoop-mapreduce-examples-3.3.0.jar
```

```
hduser@master:~$ export YARN_EXAMPLES=${HADOOP_HOME}/share/hadoop/mapreduce
hduser@master:~$
hduser@master:~$ yarn jar ${YARN_EXAMPLES}/hadoop-mapreduce-examples-3.3.0.jar
An example program must be given as the first argument.
Valid program names are:
aggregatewordcount: An Aggregate based map/reduce program that counts the words in the input files.
aggregatewordhist: An Aggregate based map/reduce program that computes the histogram of the words in the input files.
bbp: A map/reduce program that uses Bailey-Borwein-Plouffe to compute exact digits of Pi.
dbcount: An example job that counts the pageview counts from a database.
distbbp: A map/reduce program that uses a BBP-type formula to compute exact bits of Pi.
grep: A map/reduce program that counts the matches of a regex in the input.
join: A job that effects a join over sorted, equally partitioned datasets
multifilewc: A job that counts words from several files.
pentomino: A map/reduce tile laying program to find solutions to pentomino problems.
pi: A map/reduce program that estimates Pi using a quasi-Monte Carlo method.
randtextwriter: A map/reduce program that writes 10GB of random textual data per node.
randomwriter: A map/reduce program that writes 10GB of random data per node.
secondarysort: An example defining a secondary sort to the reduce.
sort: A map/reduce program that sorts the data written by the random writer.
sudoku: A sudoku solver.
terasgen: Generate data for the terasort
terasort: Run the terasort
teravalidate: Checking results of terasort
wordcount: A map/reduce program that counts the words in the input files.
wordmean: A map/reduce program that counts the average length of the words in the input files.
wordmedian: A map/reduce program that counts the median length of the words in the input files.
wordstandarddeviation: A map/reduce program that counts the standard deviation of the length of the words in the input files.
hduser@master:~$
```

```
yarn jar ${YARN_EXAMPLES}/hadoop-mapreduce-examples-3.3.0.jar pi 16 10000
```

Результат выполнения:

```
Job Finished in 56.21 seconds
Estimated value of Pi is 3.14127500000000000000
hduser@master:~$
```

```
yarn jar ${YARN_EXAMPLES}/hadoop-mapreduce-examples-3.3.0.jar pi 16 100000
```

Результат выполнения:

```
Job Finished in 41.983 seconds
Estimated value of Pi is 3.14157500000000000000
hduser@master:~$
```

2. Выполнить пример расчёта pi с числом сэмплов 1M.

```
yarn jar ${YARN_EXAMPLES}/hadoop-mapreduce-examples-3.3.0.jar pi 16 1000000
```

Результат выполнения:

```
Job Finished in 43.147 seconds  
Estimated value of Pi is 3.14159125000000000000  
hduser@master:~$
```

3. Записать в отчёт полученное значение.

Полученное значение Pi: 3.14159125000000000000

Задание 1.

Для начала выполним настройку:

Добавим строки в файл `/.bashrc` на master, node1, node2:

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
```

```
export PATH=${JAVA_HOME}/bin:${PATH}
```

```
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
```

```
export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64  
export PATH=${JAVA_HOME}/bin:${PATH}  
export HADOOP_CLASSPATH=${JAVA_HOME}/lib/tools.jar
```

```
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
```

После редактирования файла не забываем выполнить команду: `source ~/.bashrc`

На мастере создаем файл `/usr/local/hadoop/WordCount.java` и копируем в него строки из примера wordcount:

```
hduser@master: ~  
GNU nano 4.8 /usr/local/hadoop/WordCount.java  
import java.io.IOException;  
import java.util.StringTokenizer;  
  
import org.apache.hadoop.conf.Configuration;  
import org.apache.hadoop.fs.Path;  
import org.apache.hadoop.io.IntWritable;  
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 {  
  
    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);  
            }  
        }  
    }  
}
```

Далее необходимо скомпилировать java файл, командами:

```
hadoop com.sun.tools.javac.Main WordCount.java
```

```
jar cf wc.jar WordCount*.class
```

```
hduser@master:~$ cd /usr/local/hadoop  
hduser@master:/usr/local/hadoop$ hadoop com.sun.tools.javac.Main WordCount.java  
hduser@master:/usr/local/hadoop$ jar cf wc.jar WordCount*.class  
hduser@master:/usr/local/hadoop$
```

1. Подготовьте тестовые папки в HDFS для запуска задачи и положите в папку input любой текстовый файл для анализа

```
hadoop fs -mkdir -p /user/hadoop/wordcount/input
```

```
hadoop fs -mkdir -p /user/hadoop/wordcount/output
```

```
hduser@master:~$ hadoop fs -mkdir -p /user/hadoop/wordcount/input  
hduser@master:~$ hadoop fs -mkdir -p /user/hadoop/wordcount/output
```

Положим в папку input подготовленный текстовый файл:

```
hadoop fs -put /home/hduser/Desktop/ttt/textfile.txt /user/hadoop/wordcount/input
```

```
hduser@master:~$ hadoop fs -put /home/hduser/Desktop/ttt/textfile.txt /user/hado  
op/wordcount/input
```

2. Запустите пример wordcount по аналогии с примером выше.

```
hadoop jar wc.jar WordCount /user/hadoop/wordcount/input /user/hadoop/wordcount/output
```

Во время запуска Hadoop заругался на уже существующую директорию output. Поэтому поместим результат в output1.

Выполняем еще раз:

```
hadoop jar wc.jar WordCount /user/hadoop/wordcount/input /user/hadoop/wordcount/output1
```

3. После завершения просмотрите результаты в папке output и в отчёт включите несколько первых строк из файла результата (обычно называется part-r-00000)/

```
hadoop fs -cat /user/hadoop/wordcount/output1/part-r-00000
```

```
hduser@master:~$ hadoop fs -cat /user/hadoop/wordcount/output1/part-r-00000
& 1
--> 2
/> 14
<!-- 2
</datatype><table 1
</group> 4
</key> 13
</row> 14
</sql> 1
</table> 13
<?xml 1
<datatype>DATE</datatype> 1
<datatype>INTEGER</datatype> 61
<datatype>MEDIUMTEXT</datatype> 2
<datatypes 1
<default>NULL</default></row> 50
<default>NULL</default><relation 14
<group 4
<key 13
<part>id</part> 13
< Help 64
< 1
```