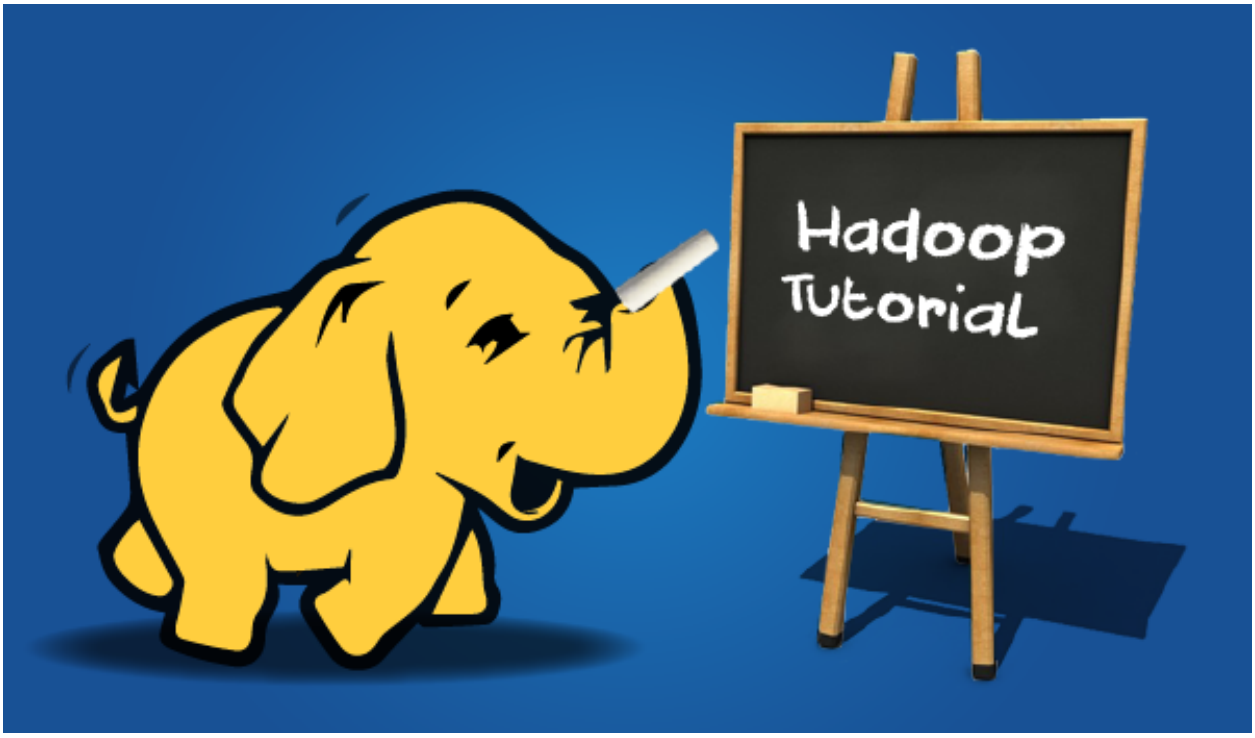
NoSQL 2015
TD 1

Professeurs

Houda Chabbi Drissi
et
Benoît Perroud

Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg



DAPLAB
Data Analysis and Processing Lab

Assistant : Christophe Bovigny

List of Frequently Asked Questions

HDFS (Hadoop Distributed File System)

Manipuler des répertoires dans HDFS

Tout d'abord nous allons créer un répertoire TD1 dans le système de fichier HDFS.

```
hdfs dfs -mkdir TD1
```

Créez un sous-répertoire EX1 dans TD1.

```
hdfs dfs -mkdir TD1/EX1
```

Listez tous les fichiers et répertoire de votre home.

```
hdfs dfs -ls
```

Vous devriez y voir votre dossier TD1 fraîchement créé.

```
cbovigny@daplab-wn-03:~$ hdfs dfs -ls
Found 1 item
drwxr-xr-x   cbovigny daplab_users          0 2015-09-17 11:24 TD1
```

Maintenant listez récursivement vos dossier afin de voir le dossier EX1 qui se trouve dans TD1.(Astuce : Utilisez l'option -R)

```
drwxr-xr-x   - cbovigny daplab_users          0 2015-09-17 11:24 TD1
drwxr-xr-x   - cbovigny daplab_users          0 2015-09-17 11:24 TD1/EX1
```

Solution :

```
cbovigny@daplab-wn-03:~$ hdfs dfs -ls -R
drwxr-xr-x   - cbovigny daplab_users          0 2015-09-17 11:24 TD1
drwxr-xr-x   - cbovigny daplab_users          0 2015-09-17 11:24 TD1/EX1
```

Effacez les deux dossiers TD1 et EX1 créés précédement en une seule commande :

```
cbovigny@daplab-wn-03:~$ hdfs dfs -rm -r TD1
```

Et à nouveau créer un dossier et sous-dossier TD1/EX1 mais cette fois-ci en une seule commande

```
cbovigny@daplab-wn-03:~$ hdfs dfs -mkdir -p TD1/EX1
```

Créer un dossier et un sous-dossier TD1_LOCAL et EX1_LOCAL, mais cette fois-ci en utilisant que les commandes linux standard et en n'employant pas le système de fichier HDFS

Manipuler des fichiers dans HDFS

Allez dans le répertoire EX1_LOCAL et téléchargez le fichier data.txt à l'addresse suivante :

```
wget https://raw.githubusercontent.com/HortonworksUniversity/DevPH_Rev4/master/labs/
    Lab2.1/data.txt
```

```
cbovignyi@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL$ wget https://raw.githubusercontent.com/
    HortonworksUniversity/DevPH_Rev4/master/labs/Lab2.1/data.txt
...
Length: 1529355 (1.5M) [text/plain]
Saving to: 'data.txt'
```

```
100%[==============================>] 1,529,355   9.23MB/s   in 0.2s

2015-09-17 12:08:46 (9.23 MB/s) - data.txt saved [1529355/1529355]
######################################################################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL$ ls
data.txt
```

Nous allons transférer le fichier data.txt de notre dossier local :
/votrehome/TD1_LOCAL/EX1_LOCAL/data.txt
dans le répertoire EX1 du système de fichier HDFS
TD1/EX1/
(Astuce : Pour ceci utilisez la commande copyFromLocal)

Solution :

```
cbovigny@daplab-wn-03:~$ hdfs dfs -copyFromLocal /home/cbovigny/TD1_LOCAL/EX1_LOCAL/
    data.txt TD1/EX1/
######################################################################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL$ hdfs dfs -ls TD1/EX1
Found 1 items
-rw-r--r--   3 cbovigny daplab_users    1529355 2015-09-17 12:14 TD1/EX1/data.txt
```

Afin de voir le contenu du fichier data.txt se trouvant dans le système de fichier HDFS, vous pouvez utiliser la commande cat :

```
cbovigny@daplab-wn-03:~$ hdfs dfs -cat TD1/EX1/data.txt
```

Nous allons effacer le fichier local data.txt et ensuite copier le fichier data.txt de HDFS en Local

```
######################### Remove local file data.txt ####################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL$ rm data.txt
######################################################################
cbovigny@daplab-wn-03:~$ hdfs dfs -copyToLocal HD1/EX1/data.txt /home/cbovigny/
    TD1_LOCAL/EX1_LOCAL/
######################################################################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL$ ls
data.txt
```

Exercice : Downloader en Local deux fichiers via les commandes ci-dessous dans le repertoire /votrehome/TD1_LOCAL/EX1_LOCAL/FILES_LOCAL/ :

```
wget https://raw.githubusercontent.com/HortonworksUniversity/DevPH_Rev4/master/labs/
    Lab6.3/data/part-m-00000
wget  https://raw.githubusercontent.com/HortonworksUniversity/DevPH_Rev4/master/labs/
    Lab6.3/data/part-m-00001
```

Créez un répertoire TD1/EX1/FILES sous HDFS et copiez les fichiers part-m-00000 et part-m-00001 à l'intérieur de ces derniers.

Solution :

```
##################################CREATE FILES_LOCAL#####################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL/$ mkdir FILES_LOCAL ; cd FILES_LOCAL
##################################WGET#################################################
```

```
wget https://raw.githubusercontent.com/HortonworksUniversity/DevPH_Rev4/master/labs/
    Lab6.3/data/part-m-00000
wget  https://raw.githubusercontent.com/HortonworksUniversity/DevPH_Rev4/master/labs/
    Lab6.3/data/part-m-00001
##################################CREATE FILES DIRECTORY HDFS###############
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL/FILES_LOCAL$ hdfs dfs -mkdir TD1/EX1/FILES
##################################COPY TWO FILES#############################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL/FILES_LOCAL$ hdfs dfs -copyFromLocal /home
    /cbovigny/TD1_LOCAL/EX1_LOCAL/FILES_LOCAL/* TD1/EX1/FILES
############################################################################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL$ hdfs dfs -ls TD1/EX1/FILES
Found 2 items
-rw-r--r--   3 cbovigny daplab_users     971339 2015-09-17 14:51 TD1/EX1/FILES/part-m
    -00000
-rw-r--r--   3 cbovigny daplab_users     142850 2015-09-17 14:51 TD1/EX1/FILES/part-m
    -00001
```

Après cet exercice il serait intéressant de concaténer ces deux fichiers en un et de les analyser en local. hdfs dfs -getmerge permet de tout faire en une seule commande. Une petite variante existe, si vous désirez insérer une ligne vide dans le fichier final entre chaque fichiers concaténés, ajouter l'option -nl

```
########################## Remove local file data.txt ####################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL/FILES_LOCAL$ rm part*
############################################################################
cbovigny@daplab-wn-03:~$ hdfs dfs -getmerge TD1/EX1/FILES/ /home/cbovigny/TD1_LOCAL/
    EX1_LOCAL/FILES_LOCAL/allfilesmerged.txt
############################################################################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL$ ls
allfilesmerged.txt  part-m-00000  part-m-00001
############################################################################
cbovigny@daplab-wn-03:~$ hdfs dfs -getmerge -nl TD1/EX1/FILES/ /home/cbovigny/
    TD1_LOCAL/EX1_LOCAL/FILES_LOCAL/allfilesmerged_newline.txt
############################################################################
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL$ ls
allfilesmerged_newline.txt  allfilesmerged.txt  part-m-00000  part-m-00001
```

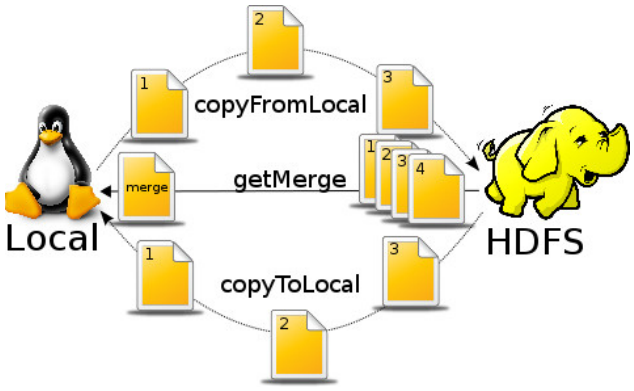


FIGURE 1 – copyFromLocal/copyToLocal/getmerge

Grandeur de Block dans HDFS

Effacez le fichier part-m-00000 dans hdfs avec la commande appropriée et ensuite essayez de le

remettre avec un blocksize de 30 bytes. (Astuce : placez l'option -D dfs.blocksize=bytesize juste après hdfs dfs)
Réponse :(Message affiché en sortie de la commande et expliquez) :

Solution :

```
cbovigny@daplab-wn-03:~/TD1_LOCAL/EX1_LOCAL/FILES_LOCAL$ hdfs dfs -D dfs.blocksize=30
    -copyFromLocal part-m-00000 TD1/EX1/FILES
copyFromLocal: Specified block size is less than configured minimum value (dfs.
    namenode.fs-limits.min-block-size): 30 < 1048576
```

Map/Reduce

Dans cette partie, nous allons essayer de comprendre comment fonctionne le map/reduce de Hadoop. Ce dernier a pour but de compter la fréquence de chaque mots dans un ou plusieurs fichiers textes.
Le programme a plusieurs sections, la première partie est composée de la classe MapClass. Que fait-elle et que retourne-t-elle ?

```
public static class MapClass extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, IntWritable> {
    // TO COMMENT
    //
    //
    //
    //
    //
    //
    private final static IntWritable one = new IntWritable(1);
    private Text word = new Text();
    // TO COMMENT
    //
    //
    //
    //
    //
    //
    //
    public void map(LongWritable key, Text value,
                    OutputCollector<Text, IntWritable> output,
                    Reporter reporter) throws IOException {
      // TO COMMENT
      //
      //
      //
      //
      //
      //
      String line = value.toString();
      // TO COMMENT
      //
      //
      //
      //
```

```
      //
      StringTokenizer itr = new StringTokenizer(line);
      // TO COMMENT
      //
      //
      //
      //
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        output.collect(word, one);
      }
    }
  }
```

La deuxième partie est naturellement la classe Reduce. Comme précédement veuillez commenter et expliquer que fait cette classe ?

```
 public static class Reduce extends MapReduceBase
    implements Reducer<Text, IntWritable, Text, IntWritable> {
   // TO COMMENT
   //
   //
   //
   //
   //
   //
    public void reduce(Text key, Iterator<IntWritable> values,
                       OutputCollector<Text, IntWritable> output,
                       Reporter reporter) throws IOException {
      int sum = 0;
      // TO COMMENT
      //
      //
      //
      //
      //
      //
      //
        while (values.hasNext()) {
        sum += values.next().get();
      }
      // TO COMMENT
      //
      //
      //
      //
      //
      //
      //
      //
      output.collect(key, new IntWritable(sum));
    }
  }
```

Voici le code source final :

```
package org.apache.hadoop.examples;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
```

```java
import java.util.List;
import java.util.StringTokenizer;

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.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobClient;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.util.Tool;
import org.apache.hadoop.util.ToolRunner;

/**
 * This is an example Hadoop Map/Reduce application.
 * It reads the text input files, breaks each line into words
 * and counts them. The output is a locally sorted list of words and the
 * count of how often they occurred.
 *
 * To run: bin/hadoop jar build/hadoop-examples.jar wordcount
 *            [-m <i>maps</i>] [-r <i>reduces</i>] <i>in-dir</i> <i>out-dir</i>
 */
public class WordCount extends Configured implements Tool {

  /**
   * Counts the words in each line.
   * For each line of input, break the line into words and emit them as
   * (<b>word</b>, <b>1</b>).
   */
  public static class MapClass extends MapReduceBase
    implements Mapper<LongWritable, Text, Text, IntWritable> {

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

    public void map(LongWritable key, Text value,
                    OutputCollector<Text, IntWritable> output,
                    Reporter reporter) throws IOException {
      String line = value.toString();
      StringTokenizer itr = new StringTokenizer(line);
      while (itr.hasMoreTokens()) {
        word.set(itr.nextToken());
        output.collect(word, one);
      }
    }
  }

  /**
   * A reducer class that just emits the sum of the input values.
   */
  public static class Reduce extends MapReduceBase
```

```
    implements Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text key, Iterator<IntWritable> values,
                       OutputCollector<Text, IntWritable> output,
                       Reporter reporter) throws IOException {
      int sum = 0;
      while (values.hasNext()) {
        sum += values.next().get();
      }
      output.collect(key, new IntWritable(sum));
    }
  }

  static int printUsage() {
    System.out.println("wordcount [-m <maps>] [-r <reduces>] <input> <output>");
    ToolRunner.printGenericCommandUsage(System.out);
    return -1;
  }


  public int run(String[] args) throws Exception {
    JobConf conf = new JobConf(getConf(), WordCount.class);
    conf.setJobName("wordcount");

    // the keys are words (strings)
    conf.setOutputKeyClass(Text.class);
    // the values are counts (ints)
    conf.setOutputValueClass(IntWritable.class);

    conf.setMapperClass(MapClass.class);
    conf.setCombinerClass(Reduce.class);
    conf.setReducerClass(Reduce.class);

    List<String> other_args = new ArrayList<String>();
    for(int i=0; i < args.length; ++i) {
      try {
        if ("-m".equals(args[i])) {
          conf.setNumMapTasks(Integer.parseInt(args[++i]));
        } else if ("-r".equals(args[i])) {
          conf.setNumReduceTasks(Integer.parseInt(args[++i]));
        } else {
          other_args.add(args[i]);
        }
      } catch (NumberFormatException except) {
        System.out.println("ERROR: Integer expected instead of " + args[i]);
        return printUsage();
      } catch (ArrayIndexOutOfBoundsException except) {
        System.out.println("ERROR: Required parameter missing from " +
                           args[i-1]);
        return printUsage();
      }
    }
    // Make sure there are exactly 2 parameters left.
    if (other_args.size() != 2) {
      System.out.println("ERROR: Wrong number of parameters: " +
                         other_args.size() + " instead of 2.");
      return printUsage();
    }
    FileInputFormat.setInputPaths(conf, other_args.get(0));
```

```
    FileOutputFormat.setOutputPath(conf, new Path(other_args.get(1)));

    JobClient.runJob(conf);
    return 0;
  }


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

}
```

Maintenant nous allons exécuter ce programme sous le cluster Hadoop du DAPLAB. Connectez-vous au cluster et ensuite créer un répertoire TD1_MAPREDUCE dans votre home. Ensuite copiez les deux fichiers 4300.txt et hadoop-0.20.1-examples.jar dans ce répertoire précédement créer.

Le pas d'après sera de placer le fichier 4300.txt dans un répertoire HDFS qui se nommera également TD1_MAPREDUCE. (CF :exercice 1)

Ensuite lancez la commande :

```
hadoop jar /home/cbovigny/TD1_MAPREDUCDE//hadoop-0.20.1-examples.jar wordcount /user/
    cbovigny/TD1_MAPREDUCE/4300.txt TD1_MAPREDUCE/OUTPUT
```

Veuillez transférer le resultat obtenu de HDFS en local et commentez l'output obtenu.