UF : BIG DATA

TP 1: Initiation à Hadoop et MapReduce

Découvrir et acquérir les compétences nécessaires pour travailler dans un environnement Hadoop

Auteurs :
Nicolas Rossard

Responsable :
S. YANGUI

16 avril 2020

Table des matières

1. ÉTAPE 1 : Installation et démarrage de Hadoop

Information : Vous pouvez exécuter le script shell.sh pour obtenir tous les résultats de tous les codes dans le dossier ./resultat. Vous trouverez dans le dossier compressé :

- data/ le dossier des données
- javaCode/ le dossier contenant le code java pour l'étape 2 et pour l'étape 3 et le code pour wordcount (dossier autres)
- pythonCode/ idem pour le code en Python
- output/ tous les résultats de tous les codes
- resultat/ le résultat du code exécuter avec le script shell.sh

1.1. Authentification sur OpenStack

Ok

1.2. Créer une nouvelle instance

Ok

1.3. Utilisation de VM

Ok

1.4. Création d'un répertoire TP

Ok

1.5. Importer le fichier purcharses.txt

Ok

1.6. Tester les commandes Hadoop

Ok

1.6.1. Créer le répertoire myinput

Ok

1.6.2. Copier le fichier purchases.txt dans /myinput

Ok

1.6.3. Afficher les dernières lignes du fichier

J'ai affiché les premières lignes du fichier pour voir comment sont agencées les données.

Listing 1.1 – Données purchases

```
[cloudera@quickstart un]$ head -50 data/purchases.txt
2012-01-01  09:00 San Jose  Men
                                  s Clothing  153.57  Visa
2012-01-01  09:00 San Diego Music 66.08 Cash
2012-01-01  09:00 Pittsburgh  Pet Supplies  493.51  Discover
2012-01-01  09:00 Omaha Children
                               s Clothing  247.18  MasterCard
2012-01-01  09:00 Austin  Cameras 379.6 Visa
2012-01-01  09:00 New York  Consumer Electronics  296.8 Cash
2012-01-01  09:00 Corpus Christi  Toys  25.38 Discover
2012-01-01  09:00 Fort Worth  Toys  213.88  Visa
2012-01-01  09:00 Las Vegas Video Games 53.26 Visa
2012-01-01  09:00 Newark  Video Games 39.75 Cash
2012-01-01  09:00 Austin  Cameras 469.63  MasterCard
2012-01-01  09:00 Greensboro  DVDs  290.82  MasterCard
2012-01-01  09:00 San Francisco Music 260.65  Discover
```

2. ÉTAPE 2 : Premiers pas avec MapReduce

2.1. Le mapper

2.1.1. Créer un fichier mapper.py

Ok

2.1.2. Étudier le mapper

Listing 2.1 – Mapper.py

```
#!/usr/bin/python
import sys
# Boucle pour recuperer les elements
for line in sys.stdin:
  # Separation des colonnes via la tabulation "\t"
  # et enregistrement des donnees
  data = line.strip().split(    )
  # Verification que data possede  elements a savoir:
  # date temps magasin produit cout paiement
  if len(data) == 6:
    # Stockage des donnees
    data, time, store, item, cost, payment = data
    # Affichage du magasin et du cout
    print           .format(store,cost)
```

2.1.3. Tester le mapper en local

Listing 2.2 – résultat local du mapper

```
[cloudera@quickstart un]$ head -50 ./data/purchases.txt|./pythonCode/mapper.py
San Jose  214.05
Fort Worth  153.57
San Diego 66.08
Pittsburgh  493.51
Omaha 235.63
Stockton  247.18
Austin  379.6
New York  296.8
Corpus Christi  25.38
Fort Worth  213.88
Las Vegas 53.26
Newark  39.75
Austin  469.63
Greensboro  290.82
San Francisco 260.65
```

2.1.4. Commenter le résultat d'exécution obtenu

Il affiche :
— Le nom de l'enseigne
— Le prix du produit vendu par l'enseigne

Pour le moment toutes les ventes de chaque magasin ne sont pas regroupées sur une seule ligne. On peut le voir comme ceci :

Listing 2.3 – Analyse du mapper pour le magasin de Buffalo

```
head -3000 ./data/purchases.txt|./pythonCode/mapper.py | grep Buffalo
Buffalo 483.82
Buffalo 344.08
Buffalo 292.38
Buffalo 269.4
Buffalo 439.22
Buffalo 263.18
Buffalo 30.71
Buffalo 291.75
```

2.2. Le reducer

2.2.1. Créer le fichier reducer.py

Ok

2.2.2. Étudier le reducer

Listing 2.4 – reducer.py

```
#!/usr/bin/python
import sys

salesTotal = 0
oldKey = None
for line in sys.stdin:
  # Parcourt les donnees crees par le mapper (magasin,cout)
  # Le nom du magasin est la clef
  data = line.strip().split(    )

  # Si la longueur des donnees n'est pas la bonne
  # on passe ce "couple clef,valeur)
  if len(data) !=2:
    continue
  # enregistrement des donnees dans data
  thisKey, thisSale = data
  # print oldKey and oldKey
  # Dans cette partie les donnees sorties par mapper doivent etre triees
  # Sinon cela ne fonctionne pas
  # On regarde si la clef est la meme
  if oldKey and oldKey != thisKey:
    # Si faux
    # On affiche la somme des produits par le magasin
    print           .format(oldKey,salesTotal)
    # Remise a 0 de la vente totale
    salesTotal = 0
  # On met la nouvelle clef
  oldKey = thisKey

  # On ajoute le cout du produit vendu au reste (0 si nouveau magasin)
  salesTotal += float (thisSale)

# Affiche la derniere clef si non-nulle
if oldKey != None:
  print oldKey,    ,salesTotal
```

2.2.3. Tester le reducer en local

Listing 2.5 – Résultat en local du reducer

```
head -50 ./data/purchases.txt |./pythonCode/mapper.py |sort |./pythonCode/reducer.py
Anchorage 327.6
Aurora  117.81
Austin  1176.98
Boston  418.94
Buffalo 483.82
Chandler  758.17
Chicago 31.08
Corpus Christi  25.38
Fort Wayne  370.55
```

2.2.4. Commenter le résultat d'exécution obtenu

Le reducer récupère les couples créés par le mapper (ils ont été triés avant par la fonction sort). Puis il fait la somme des ventes pour chaque magasin.

L'avantage d'avoir trié les couples sortis par le mapper permet au reducer de savoir quand il n'y a plus de vente à additionner pour une enseigne donnée. Quand la clé qui correspond au nom de l'enseigne est différente, on envoie le total des ventes de l'enseigne et on passe à la nouvelle enseigne.

2.3. Lancer un job entier

2.3.1. Exécuter un job Hadoop et récupérer le résultat dans un fichier local

Remarque : Pour l'alias hs le *.jar se situe à l'adresse suivante :
/usr/lib/hadoop-0.20-mapreduce/contrib/streaming/hadoop-streaming-2.6.0-mr1-cdh5.12.0.jar

2.3.2. Quel est le résultat obtenu pour la totalité des ventes du magasin de Buffalo ?

Remarque : Le résultat n'est pas tout à fait exact il est différent de celui trouvé avec le mapreduce en Python. Je ne sais pas ce qui engendre cette imprécision (l'utilisation d'un Double, le cast ou des tokens ?).

Listing 2.6 – Exécution d'un Job et résultat pour le magasin de Buffalo

```
[cloudera@quickstart un]$ hs ./pythonCode/mapper.py ./pythonCode/reducer.py tp1/input tp1/↘
    joboutputPython

[cloudera@quickstart un]$ hadoop fs -get tp1/joboutputPython/part*

[cloudera@quickstart un]$ grep Buffalo part-00000
Buffalo 10001941.19
```

2.4. Coder avec JAVA

Remarque : J'ai utilisé un projet wordcount que j'avais déjà créé. Je n'ai pas changé le nom du projet ni les noms des classes mais j'ai modifié et détaillé le projet pour qu'il exécute ce que vous demandiez. Vous trouverez le code en annexe (annexe A)

Vous trouverez le projet en entier dans le dossier ZIP : *javaSalesTotal.zip*. Pour exécuter le mapReduce, il faut utiliser le dossier JAR : *wordcount_demo.jar*

Listing 2.7 – Exécution du code java avec le fichier jar

```
[cloudera@quickstart wordcount_demo_jar]$ hadoop fs -ls tp1
Found 3 items
drwxr-xr-x   - cloudera cloudera          0 2020-04-12 03:22 tp1/input
drwxr-xr-x   - cloudera cloudera          0 2020-04-14 08:09 tp1/joboutputPython
drwxr-xr-x   - cloudera cloudera          0 2020-04-12 02:59 tp1/joboutputWC

[cloudera@quickstart wordcount_demo_jar]$ hadoop jar wordcount_demo.jar wordcount.↘
    WordCountDriver tp1/input tp1/joboutputJava

[cloudera@quickstart wordcount_demo_jar]$ hadoop fs -cat tp1/joboutputJava/part* |grep ↘
    Buffalo
Buffalo 1.0001941189999972E7
```

3. ÉTAPE 3 : Première application

Pour cette étape, j'ai créé un unique projet MAVEN regroupant tous les mappers et reducers ainsi que les drivers. Le dossier JAR se nomme *tp1etape3.jar*. Ce dossier prend 3 arguments :

- chemin du dossier/fichier en entrée
- chemin du dossier de sortie
- drivers à utiliser :
  - 1 : la liste des ventes par catégorie de produits
  - 2 : Le montant de la vente le plus élevé pour chaque magasin
  - 3 : La valeur totale de toutes les ventes de tous les magasins

J'ai aussi créé ces mappers et reducers en Python que vous trouverez dans le dossier **pythonCode**. Je voulais voir s'il y avait un écart comme observé lors de l'étape 2.

Les résultats sont dans le dossier **output** :

- **outputCategory** : La liste des ventes par catégorie de produits
- **outputSaleMax** : Le montant de la vente le plus élevé pour chaque magasin
- **outputTotal** : La valeur totale de toutes les ventes de tous les magasins

3.1. Donner la liste des ventes par catégorie de produits

Pour cette partie, le mapper renvoie <le nom du produit, son prix> et le reducer <le nom du produit, la somme de tous les prix de ce produit> Pour exécuter :
JAVA : *$ hadoop jar tp1etape3.jar RunClass ../purchases.txt ../joboutput 1*
Python : *$ hs ../mapCat.py ../redCat.py ../purchases.txt ../outputCatPyth*

3.1.1. Quelle est la valeur des ventes pour la catégorie Toys ?

Toys = 5.7463477109999225E7 en Java
Toys = 57463477.11 en Python

3.1.2. Quelle est la valeur des ventes pour la catégorie Consumer electronics ?

Consumer Electronics 5.7452374129999146E7 en Java
Consumer Electronics 57452374.13 en Python

3.2. Donner le montant de la vente le plus élevé pour chaque magasin :

Dans cette partie le mapper renvoie <nom du magasin, prix du produit> et le reducer compare les prix du produit pour chaque magasin. Pour exécuter :
Java : *$ hadoop jar tp1etape3.jar RunClass ../purchases.txt ../joboutput 2*
Python : *$ hs ../mapSaleMax.py ../redSalemax.py ../purchases.txt ../outputSMPyth*

3.2.1. Quelle est cette valeur pour le magasin Reno ?

Reno 499.99 en Python et Java

3.2.2. Quelle est cette valeur pour le magasin Toledo ?

Toledo 499.98 en Python et Java

3.2.3. Quelle est cette valeur pour le magasin Chandler ?

Chandler 499.98 en Python et Java

3.3. Quel est le nombre total de ventes et la valeur totale des ventes de tous les magasins confondus ?

Pour cette partie, j'ai créé un mapper qui renvoie <"Total", prix du produit>, je n'ai pas changé le reducer qui renvoie <la clé, la somme totale de cette clé> (ici nous avons qu'une seule clé).

Remarque : Le test unitaire du MapReduce ne fonctionnait pas alors que le résultat avec purchases.txt est bon. Il semblerait que le mapreduceDriver utilisé avec mrunit n'arrive pas à la fin de l'exécution du reducer (le mapper et le reducer fonctionnent normalement, c'est après que cela déclenche l'erreur).

Pour exécuter :
Java *$ hadoop jar tp1etape3.jar RunClass ../purchases.txt ../joboutput 3*
Python : *$ hs ../mapTotal.py ../redTotal.py ../purchases.txt ../outputTotalPyth*
Nb de ventes = 4138476 Total = 1.0344579532599444E9 en Java
Nb de ventes = 4138476 Total = 1034457953.26 en Python

A. Valeur totale des ventes par magasin

A.1. Mapper.java

Listing A.1 – WordCountMapper.java

```
1  package wordcount;
2
3  //import org.apache.commons.logging.Log;
4  //import org.apache.commons.logging.LogFactory;
5  import org.apache.hadoop.io.DoubleWritable;
6  import org.apache.hadoop.io.LongWritable;
7  import org.apache.hadoop.io.Text;
8  import org.apache.hadoop.mapreduce.Mapper;
9
10 import java.io.IOException;
11 import java.util.StringTokenizer;
12 public class WordCountMapper extends Mapper<LongWritable, Text, Text, DoubleWritable> {
13     //public static final Log log = LogFactory.getLog(WordCountMapper.class);
14     private final static DoubleWritable price = new DoubleWritable();
15     private Text shop = new Text();
16
17     @Override
18     public void map(LongWritable key, Text value, Context context) throws IOException, ↘
           InterruptedException {
19         String line = value.toString();
20         //System.out.println(line);
21         /*
22          * Use Delimiter to separate tokens
23          */
24         StringTokenizer tokenizer = new StringTokenizer(line,    );
25         /*
26          * Check nb of tokens = 6
27          */
28         if (tokenizer.countTokens() != 6){
29             //System.out.println("Erreur");
30             //System.out.println(tokenizer.countTokens());
31             return;
32         }
33         /*
34          * Loop all elements of the line
35          * For now we use only two datas but maybe we will use more later
36          */
37         int i = 0;
38         while (tokenizer.hasMoreTokens()){
39             switch(i){
40                 case 2:
41                     shop.set(tokenizer.nextToken());
42                     break;
43                 case 4:
44                     price.set(Double.parseDouble(tokenizer.nextToken()));
45                     break;
46                 default:
47                     tokenizer.nextToken();
48                     break;
49             }
50             i++;
51         }
52         /*
53          * Write Results
```

```
54          */
55         //System.out.println("Shop: " + shop +"\tprice: " +price);
56         //log.info("Map Key" + shop + "\tprice: " + price);
57         context.write(shop, price);
58     }
59
60     public void run(Context context) throws IOException, InterruptedException {
61         setup(context);
62         //System.out.println("———————————————— MAPPER ————————————————");
63         while (context.nextKeyValue()) {
64             map(context.getCurrentKey(), context.getCurrentValue(), context);
65         }
66         cleanup(context);
67         //System.out.println("————————————————————————————————————————");
68
69     }
70
71 }
72 /* I find this code on internet:
73  * String line = value.ToString();
74  * String[] singleShopData = line.split("\t");
75  * shop.set(singleShopData[2]);
76  * price.set(new DoubleWritable(singleShopData[4]);
77  * context.write(shop,price);
78  */
```

A.2. Reducer.java

Listing A.2 – Reducer.java

```
1  package wordcount;
2  import org.apache.hadoop.io.DoubleWritable;
3  import org.apache.hadoop.io.Text;
4  import org.apache.hadoop.mapreduce.Reducer;
5  //import org.apache.commons.logging.Log;
6  //import org.apache.commons.logging.LogFactory;
7
8  import java.io.IOException;
9  import java.util.Iterator;
10
11 public class WordCountReducer extends Reducer<Text, DoubleWritable, Text, DoubleWritable> {
12     //public static final Log log = LogFactory.getLog(WordCountMapper.class);
13
14     private DoubleWritable salesTotal = new DoubleWritable();
15
16     @Override
17     public void reduce(final Text key, final Iterable<DoubleWritable> values,
18                        final Context context) throws IOException, InterruptedException {
19         //System.out.println("———————————————— Reducer ————————————————");
20         double sum = 0;
21         Iterator<DoubleWritable> iterator = values.iterator();
22         //System.out.println(iterator.toString());
23         while (iterator.hasNext()) {
24             sum += iterator.next().get();
25         }
26         salesTotal.set(sum);
27         //System.out.println(key + "\t" + sum);
28         //log.info("RED key:" + key + " Sum : " + sum);
29
30         context.write(key, salesTotal);
31         //System.out.println("————————————————————————————————————————");
32
33     }
34 }
```

A.3. Driver.java

Listing A.3 – Driver.java

```
1  package wordcount;
2
3  import org.apache.hadoop.conf.Configured;
4  import org.apache.hadoop.fs.FileSystem;
5  import org.apache.hadoop.fs.Path;
6  import org.apache.hadoop.io.DoubleWritable;
7  import org.apache.hadoop.io.Text;
8  import org.apache.hadoop.mapreduce.Job;
9  import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
10 import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;
11 import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
12 import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;
13 import org.apache.hadoop.util.Tool;
14 import org.apache.hadoop.util.ToolRunner;
15
16 public class WordCountDriver extends Configured implements Tool {
17     public int run(String[] args) throws Exception {
18         /*
19          * Validate that two arguments were passed from the command line.
20          */
21         if (args.length != 2) {
22             System.out.println(                          );
23             System.exit(-1);
24         }
25
26         /*
27          * Instantiate a Job object for your job's configuration.
28          * Specify an easily-decipherable name for the job.
29          */
30         Job job = Job.getInstance(getConf());
31         job.setJobName(            );
32
33         /*
34          * Specify the jar file that contains your driver, mapper, and reducer.
35          * Hadoop will transfer this jar file to nodes in your cluster running
36          * mapper and reducer tasks.
37          */
38         job.setJarByClass(WordCountDriver.class);
39         job.setMapperClass(WordCountMapper.class);
40         job.setReducerClass(WordCountReducer.class);
41
42         /*
43          * Define output tuple
44          */
45         job.setOutputKeyClass(Text.class);
46         job.setOutputValueClass(DoubleWritable.class);
47
48         /*
49          * Define input/output format to use for the job
50          */
51         job.setInputFormatClass(TextInputFormat.class);
52         job.setOutputFormatClass(TextOutputFormat.class);
53
54         /*
55          * Save paths
56          */
57         Path inputFilePath = new Path(args[0]);
58         Path outputFilePath = new Path(args[1]);
59
60         /*
61          * Allow recursive
62          */
63         FileInputFormat.setInputDirRecursive(job, true);
64         /*
65          * Configure input/output file
66          */
67         FileInputFormat.addInputPath(job, inputFilePath);
68         FileOutputFormat.setOutputPath(job, outputFilePath);
```

```
69          FileSystem fs = FileSystem.newInstance(getConf());
70
71          /*
72           * if output file already exists it will delete it
73           */
74          if (fs.exists(outputFilePath)) {
75              fs.delete(outputFilePath, true);
76          }
77
78          /*
79           * Start the MapReduce job and wait for it to finish.
80           * If it finishes successfully, return 0. If not, return 1.
81           */
82          return job.waitForCompletion(true) ? 0: 1;
83      }
84
85      public static void main(String[] args) throws Exception {
86          WordCountDriver wordcountDriver = new WordCountDriver();
87          int res = ToolRunner.run(wordcountDriver, args);
88          System.exit(res);
89      }
90  }
```

A.4. TestMapReduce

Listing A.4 – Test.java

```
1  import org.apache.hadoop.io.DoubleWritable;
2  import org.apache.hadoop.io.LongWritable;
3  import org.apache.hadoop.io.Text;
4  import org.apache.hadoop.mrunit.mapreduce.MapDriver;
5  import org.apache.hadoop.mrunit.mapreduce.MapReduceDriver;
6  import org.apache.hadoop.mrunit.mapreduce.ReduceDriver;
7  import org.junit.Before;
8  import org.junit.Test;
9  import wordcount.WordCountMapper;
10 import wordcount.WordCountReducer;
11
12 import java.util.ArrayList;
13 import java.util.List;
14
15 public class TestWordCount {
16     MapReduceDriver<LongWritable, Text, Text, DoubleWritable, Text, DoubleWritable> ↘
           mapReduceDriver;
17     MapDriver<LongWritable, Text, Text, DoubleWritable> mapDriver;
18     ReduceDriver<Text, DoubleWritable, Text, DoubleWritable> reduceDriver;
19
20     @Before
21     public void setUp() {
22         /*
23          * Initialize mapper/reducer/mapReduce drivers
24          */
25         WordCountMapper mapper = new WordCountMapper();
26         WordCountReducer reducer = new WordCountReducer();
27         mapDriver = new MapDriver<LongWritable, Text, Text, DoubleWritable>();
28         mapDriver.setMapper(mapper);
29         reduceDriver = new ReduceDriver<Text, DoubleWritable, Text, DoubleWritable>();
30         reduceDriver.setReducer(reducer);
31         mapReduceDriver = new MapReduceDriver<LongWritable, Text, Text, DoubleWritable, Text,↘
               DoubleWritable>();
32         mapReduceDriver.setMapper(mapper);
33         mapReduceDriver.setReducer(reducer);
34     }
35
36     @Test
37     public void testMapper() {
38         Text value1 = new Text(                                                    )↘
               ;
39         //Text value2 = new Text("2012-01-01\t09:00\tBahamas\tMen's Clothing\t300\tAmex");
```

```
40
41          mapDriver.withInput(new LongWritable(0),value1);
42          //mapDriver.withInput(new LongWritable(1),value2);
43
44          mapDriver.withOutput(new Text(          ), new DoubleWritable(214.05));
45          //mapDriver.withOutput(new Text("Bahamas"), new DoubleWritable(300.0));
46
47          mapDriver.runTest();
48      }
49
50      /* @Test
51       public void testMapperSeveralInputs() throws IOException {
52          //Text value1 = new Text("2012-01-01\t09:00\tNew York\tMen's Clothing\t214.05\tAmex\n↘
                ");
53          //Text value2 = new Text("2012-01-01\t09:00\tBahamas\tMen's Clothing\t300\tAmex");
54          Text val = new Text("2012-01-01\t09:00\tNew York\tToys\t25.38\tDiscover\n" +
55                  "2012-01-01\t09:00\tNew York\tToys\t213.88\tVisa");
56
57          //mapDriver.withInput(new LongWritable(1),value1);
58          //mapDriver.withInput(new LongWritable(2),value2);
59          mapDriver.withInput(new LongWritable(0),val);
60          final Pair ny =new Pair<Text,DoubleWritable>(new Text("New York"), new DoubleWritable↘
                (214.05));
61          final Pair bh =new Pair<Text,DoubleWritable>(new Text("Bahamas"), new DoubleWritable↘
                (300));
62          final List<Pair<Text, DoubleWritable>> result;
63          result = mapDriver.run();
64          assertThat(result)
65                  .isNotNull()
66                  .hasSize(2)
67                  .contains(ny,bh);
68
69          mapDriver.runTest();
70      }*/
71
72      @Test
73      public void testReducer() {
74          List<DoubleWritable> values = new ArrayList<DoubleWritable>();
75          values.add(new DoubleWritable(100.8));
76          values.add(new DoubleWritable(255));
77          reduceDriver.withInput(new Text(         ), values);
78          reduceDriver.withOutput(new Text(         ), new DoubleWritable(355.8));
79          reduceDriver.runTest();
80      }
81
82      @Test
83      public void testMapReduce() {
84          Text value1 = new Text(                                                              )↘
                ;
85          Text value2 = new Text(                                                          );
86          Text value3 = new Text(                                                             );
87
88          mapReduceDriver.addInput(new LongWritable(0),value1);
89          mapReduceDriver.addInput(new LongWritable(1),value2);
90          mapReduceDriver.addInput(new LongWritable(2),value3);
91          mapReduceDriver.addOutput(new Text(         ), new DoubleWritable(300));
92          mapReduceDriver.addOutput(new Text(          ), new DoubleWritable(464.05));
93          mapReduceDriver.runTest();
94
95      }
96
97  }
```