Deployment requirements and steps

- Windows 10 64-bit
- JDK 1.8.0_361
- Hadoop 2.7.7
- Spark 3.0.3

Runtime screenshots

Hadoop

First run “start-all.cmd” to start clusters.

```
Administrator: Command Prompt
*************************************************************/

C:\hadoop-2.7.7\sbin>start-all.cmd
This script is Deprecated. Instead use start-dfs.cmd and start-yarn.cmd
starting yarn daemons

C:\hadoop-2.7.7\sbin>cd C:\Users\user\Downloads\Hadoop
```

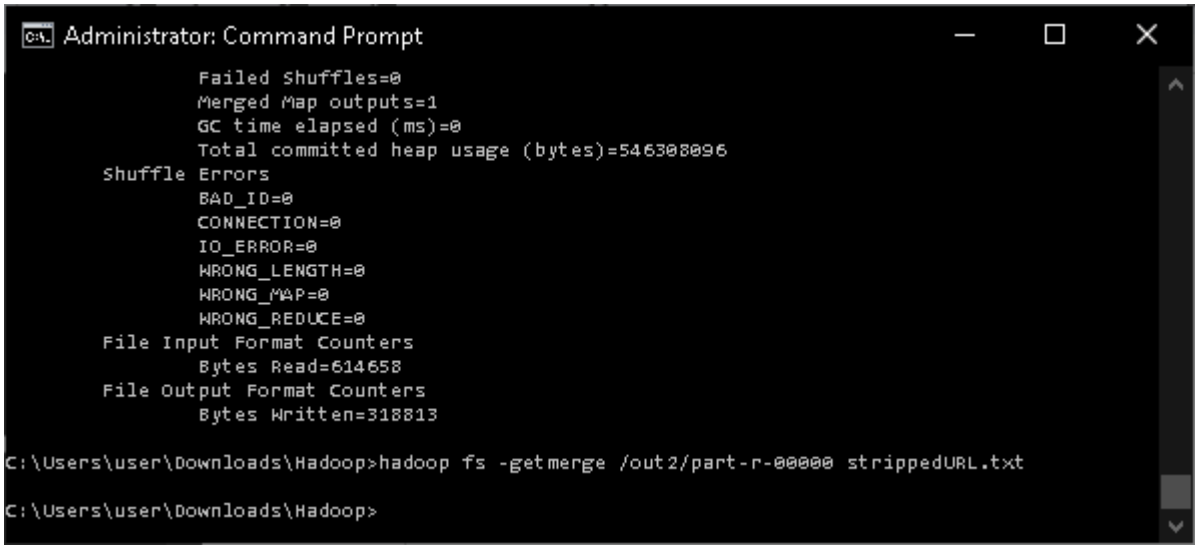
After making an input directory in HDFS, search_data.sample is put into it.

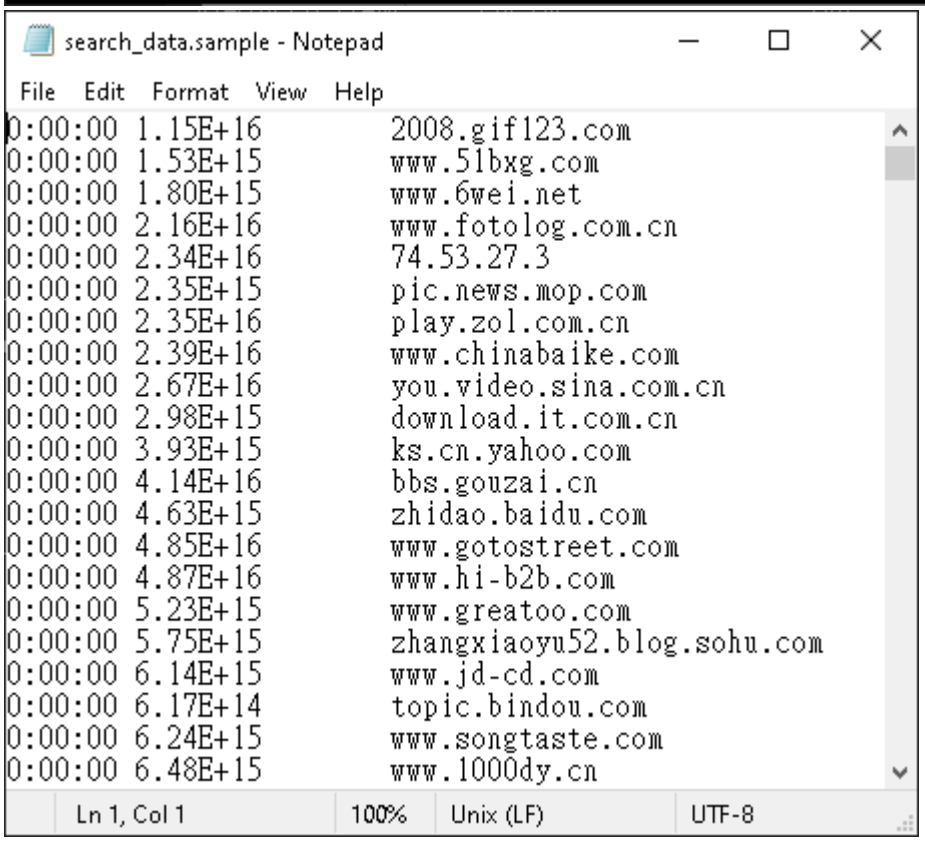
```
Administrator: Command Prompt
C:\Users\user\Downloads\Hadoop>jps
12212 DataNode
13240 NameNode
5800 ResourceManager
7640 Jps
7452 NodeManager

C:\Users\user\Downloads\Hadoop>hadoop fs -mkdir /input

C:\Users\user\Downloads\Hadoop>hadoop fs -put search_data.sample /input

C:\Users\user\Downloads\Hadoop>hadoop fs -ls
ls: `.': No such file or directory

C:\Users\user\Downloads\Hadoop>hadoop fs -ls /input
Found 1 items
-rw-r--r--   1 user supergroup     614658 2023-05-24 00:53 /input/search_data.sample
```

Task 1:

Now we compile and run URLStripper and get the output.

```
Administrator: Command Prompt
C:\hadoop-2.7.7\sbin>cd C:\Users\user\Downloads\Hadoop

C:\Users\user\Downloads\Hadoop>hadoop com.sun.tools.javac.Main URLStripper.java

C:\Users\user\Downloads\Hadoop>jar cf us.jar URLStripper*.class

C:\Users\user\Downloads\Hadoop>hdfs -ls /input
Unrecognized option: -ls
Error: Could not create the Java Virtual Machine.
Error: A fatal exception has occurred. Program will exit.

C:\Users\user\Downloads\Hadoop>hadoop fs -ls /input
Found 1 items
-rw-r--r--   1 user supergroup     614658 2023-05-24 00:53 /input/search_data.sample

C:\Users\user\Downloads\Hadoop>hadoop jar us.jar URLStripper /input /output0
23/05/24 20:23:46 INFO Configuration.deprecation: session.id is deprecated. Instead, use dfs.
```





Note that some duplicate lines are eliminated.

<u>Spark</u>

Before the data processing, I renamed search_data.sample as search_data.csv so the whole file can be read directly into a dataframe in Spark. Also note that MapReduce messes up the column names, so I just copy and paste it manually to fix it.

search_data.sample - Notepad

File Edit Format View Help

queryTime	userId	clickUrl
0:00:00	1.15E+16	2008.gif123.com
0:00:00	1.53E+15	www.51bxg.com
0:00:00	1.80E+15	www.6wei.net
0:00:00	2.16E+16	www.fotolog.com.cn
0:00:00	2.34E+16	74.53.27.3
0:00:00	2.35E+15	pic.news.mop.com
0:00:00	2.35E+16	play.zol.com.cn
0:00:00	2.39E+16	www.chinabaike.com
0:00:00	2.67E+16	you.video.sina.com.cn
0:00:00	2.98E+15	download.it.com.cn
0:00:00	3.93E+15	ks.cn.yahoo.com
0:00:00	4.14E+16	bbs.gouzai.cn
0:00:00	4.63E+15	zhidao.baidu.com
0:00:00	4.85E+16	www.gotostreet.com
0:00:00	4.87E+16	www.hi-b2b.com
0:00:00	5.23E+15	www.greatoo.com
0:00:00	5.75E+15	zhangxiaoyu52.blog.sohu.com
0:00:00	6.14E+15	www.jd-cd.com
0:00:00	6.17E+14	topic.bindou.com
0:00:00	6.24E+15	www.songtaste.com

Ln 1, Col 10 | 100% | Unix (LF) | UTF-8

Now we can run them in Spark shell. Then we can copy and paste the .scala codes to run them.

C:\Windows\System32\cmd.exe - spark-shell

```
Microsoft Windows [Version 10.0.19045.2728]
(c) Microsoft Corporation. All rights reserved.

C:\Users\user\Downloads>spark-shell
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
Spark context Web UI available at http://hp:4040
Spark context available as 'sc' (master = local[*], app id = local-1684941293205).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 3.0.3
      /_/

Using Scala version 2.12.10 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_361)
Type in expressions to have them evaluated.
Type :help for more information.

scala> 23/05/24 23:15:05 WARN ProcfsMetricsGetter: Exception when trying to compute pagesize, a
s a result reporting of ProcessTree metrics is stopped
```

Task 2:

```
C:\Windows\System32\cmd.exe - spark-shell

scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row

scala>

scala> import org.apache.spark.rdd.RDD
import org.apache.spark.rdd.RDD

scala>

scala> val df = spark.read.format("csv").option("header",true).option("sep","\t").load("C:/Users/user/
Downloads/Hadoop/search_data.csv")
df: org.apache.spark.sql.DataFrame = [queryTime: string, userId: string ... 1 more field]

scala>

scala> val lineRDD: RDD[String] = df.select("clickUrl").rdd.map(_.mkString(""))
lineRDD: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[15] at map at <console>:27

scala>

scala> val wordRDD: RDD[String] = lineRDD.flatMap(line => line.split("\\."))
wordRDD: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[16] at flatMap at <console>:27

scala>

scala> val kvRDD: RDD[(String, Int)] = wordRDD.map(word => (word, 1))
kvRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[17] at map at <console>:27

scala>

scala> val wordCounts: RDD[(String, Int)] = kvRDD.reduceByKey((x, y) => x + y)
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[18] at reduceByKey at <console>:27

scala>

scala> val exchangeRDD: RDD[(Int, String)] = wordCounts.map{case (k,v)=>(v,k)}
exchangeRDD: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[19] at map at <console>:27

scala>

scala> val sortRDD: RDD[(Int, String)] = exchangeRDD.sortByKey(false)
sortRDD: org.apache.spark.rdd.RDD[(Int, String)] = ShuffledRDD[20] at sortByKey at <console>:27

scala>

scala> sortRDD.take(10).foreach(println)
(7820,com)
(4108,www)
(2322,cn)
(769,baidu)
(633,news)
(594,net)
(526,zhidao)
(493,bbs)
(492,sina)
(413,sohu)

scala>
```

Task 3:

```
C:\Windows\System32\cmd.exe - spark-shell

scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row

scala>

scala> import org.apache.spark.rdd.RDD
import org.apache.spark.rdd.RDD

scala>

scala> val df = spark.read.format("csv").option("header",true).option("sep","\t").load("C:/Users/user/Downloads/H
adoop/search_data.csv")
df: org.apache.spark.sql.DataFrame = [queryTime: string, userId: string ... 1 more field]

scala>

scala> val lineRDD: RDD[String] = df.select("queryTime").rdd.map(_.mkString(""))
lineRDD: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[15] at map at <console>:27

scala>

scala> val wordRDD: RDD[String] = lineRDD.map(line => line.dropRight(3))
wordRDD: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[16] at map at <console>:27

scala>

scala> val kvRDD: RDD[(String, Int)] = wordRDD.map(word => (word, 1))
kvRDD: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[17] at map at <console>:27

scala>

scala> val wordCounts: RDD[(String, Int)] = kvRDD.reduceByKey((x, y) => x + y)
wordCounts: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[18] at reduceByKey at <console>:27

scala>

scala> val exchangeRDD: RDD[(Int, String)] = wordCounts.map{case (k,v)=>(v,k)}
exchangeRDD: org.apache.spark.rdd.RDD[(Int, String)] = MapPartitionsRDD[19] at map at <console>:27

scala>

scala> val sortRDD: RDD[(Int, String)] = exchangeRDD.sortByKey(false)
sortRDD: org.apache.spark.rdd.RDD[(Int, String)] = ShuffledRDD[20] at sortByKey at <console>:27

scala>

scala> sortRDD.take(10).foreach(println)
(1050,0:02)
(1019,0:00)
(1018,0:03)
(1017,0:04)
(1009,0:01)
(1004,0:06)
(994,0:05)
(990,0:08)
(972,0:07)
(616,0:09)

scala>
```

Code description

Hadoop: URLStripper.java

S3Sample.java | test.java | URLStripper.... | S3FileUploa... | HomeServlet.... | upload.jsp

```java
import java.io.IOException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.NullWritable;
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 URLStripper {

  public static class UrlStripMapper
      extends Mapper<Object, Text, Text, NullWritable>{

    public void map(Object key, Text value, Context context
                    ) throws IOException, InterruptedException {
        String line = value.toString();
        if (line != null && !line.isEmpty()){context.write(new Text(line),NullWritable.get());}
    }
  }

  public static class UrlStripReducer
      extends Reducer<Text,NullWritable,Text,NullWritable> {

    public void reduce(Text key, Iterable<NullWritable> values,
                       Context context
                       ) throws IOException, InterruptedException {

      context.write(new Text(key.toString().split("/")[0]), NullWritable.get());
    }
  }

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

map

reduce

Map: Key is each line in search_data.sample; Value is null

Reduce: Write each Key keeping only texts before the first “/”

Spark

task2.scala - Notepad

File Edit Format View Help

```
import org.apache.spark.sql.Row

import org.apache.spark.rdd.RDD

val df = spark.read.format("csv").option("header",true).option("sep","\t").load
("C:/Users/user/Downloads/Hadoop/search_data.csv")

val lineRDD: RDD[String] = df.select("clickUrl").rdd.map(_.mkString(""))

val wordRDD: RDD[String] = lineRDD.flatMap(line => line.split("\\."))

val kvRDD: RDD[(String, Int)] = wordRDD.map(word => (word, 1))

val wordCounts: RDD[(String, Int)] = kvRDD.reduceByKey((x, y) => x + y)

val exchangeRDD: RDD[(Int, String)] = wordCounts.map{case (k,v)=>(v,k)}

val sortRDD: RDD[(Int, String)] = exchangeRDD.sortByKey(false)

sortRDD.take(10).foreach(println)
```

Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8

1. Load the dataset to a dataframe. One should change the path of the source dataset in their environment.
2. Select a column and convert it to RDD[String]
   (for task3.scala, select("queryTime") is used instead)
3. Split each line by "." (Tokenize the URLs)
   (for task3.scala, replace flatMap(...) with map(line => line.dropRight(3)))
4. Word count MapReduce.
5. Exchange key and value and sort by key descendingly.
6. Print top 10 results.