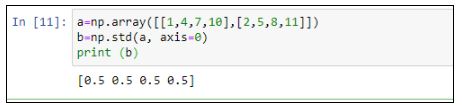
MapReduce Example to Analyze Call Data Records

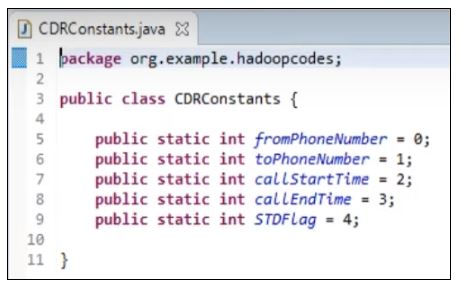
Shown below is a sample data of call records. It has the information regarding phone numbers from which the call was made, and to which phone number it was made. The data also gives information about the total duration of each call. It also tells you if the call made was a local (0) or an STD call (1).



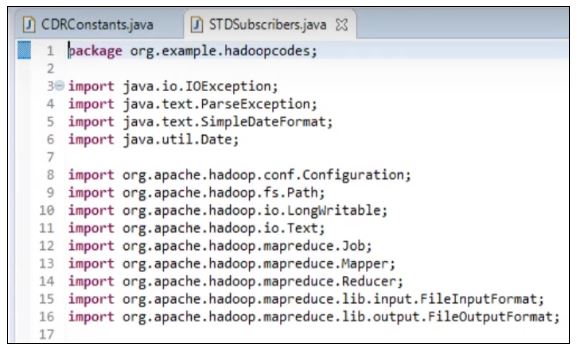
We’ll use this data to perform certain operations with the help of a MapReduce algorithm. One of the operations you can perform is to find all the phone numbers that made more than 60 minutes of STD calls.

We’ll use Java programming language to do this task.

1. Let’s first declare our constants for the fields.



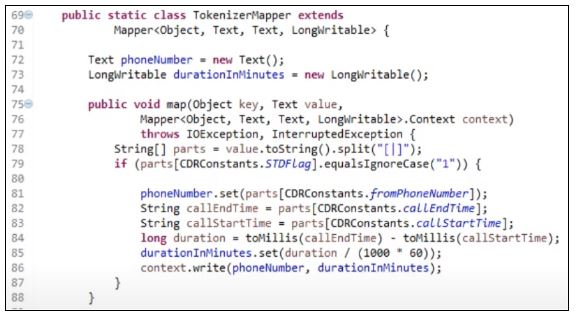
2. Import all the necessary packages to make sure we use the classes in the right way.



3. The order of the driver, mapper, and reducer class does not matter. So, let’s create a mapper that will do the map task.

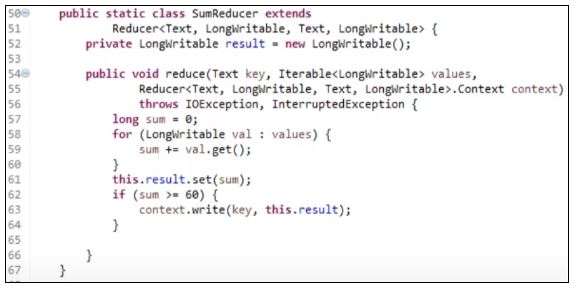
* We will create a TokenizerMapper that will extend our Mapper class. It accepts the desired data types (line 69-70).
* We’ll assign phone numbers and the duration of the calls in minutes (line 72-73).
* The map task works on a string, and it breaks it into individual elements based on a delimiter (line 75-78).
* Then, we’ll check if the string that we are looking for has an STD flag (line 79).
* We will then set the phone numbers using the constant class and find the duration (line 81-83).
* Finally, we’ll extract the phone numbers and the duration of the call made by a particular phone number (line 84-86).

This mapper class will return an intermediate output, which would then be sorted and shuffled and passed on to the reducer.

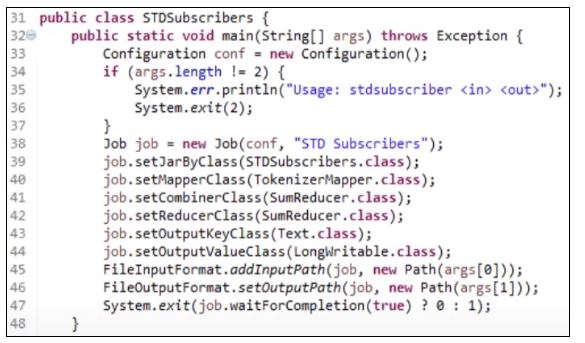


4. Next, we define our reducer class.

* So, we define our reducer class called SumReducer. The reducer uses the right data types specific to Hadoop MapReduce (line 50-52).
* The reduce (Object, Iterable, Context) method is called for each <key, (collection of values)> in the sorted inputs. The output of the reduce task is written to a RecordWriter via TaskInputOutputContext.write(Object, Object) (line 54-56).
* It looks into all the keys and values. Wherever it finds that the keys that are repeating and the duration is more than 60 minutes, it would return an aggregated result (line 57-63).



5. The driver class has all the job configurations, mapper, reducer, and also a combiner class. It is responsible for setting up a MapReduce job to run in the Hadoop cluster. You can specify the names of Mapper and Reducer Classes long with data types and their respective job names.



6. Now, package the files as .jar and transfer it to the Hadoop cluster and run it on top of YARN.

You can locate your call records file using hdfs dfs -ls “Location of the file”

7. Now, we’ll input the call records file for processing. Use the command below to locate the file and give the class name, along with another file location to save the output.

hadoop jar STDSubscribers.jar org.example.hadoopcodes.STDSubscribers sampleMRIn/calldatarecords.txt sampleMROutput-2

8. Once you run the above command successfully, you can see the output by checking the directory.

hdfs dfs -cat sampleMROutput-2/part-r-00000

