Course : Distributed Computing in Grids and Clouds

Course ID : NWEN406

Title : Part 2 - Your Tasks

Submitted By : Sharan R Prasad

Student ID : 300443276

# 1. Introduction

The project involves running map reduce jobs on the Hadoop cluster provided. The Hadoop version used in this project is 2.8.

# 2. Task 1

Find the search history for a specified user. The output format is {AnonID, Query, Item Rank, Click URL }

## 2.1 Program Design

The program takes the ID who’s search history has to be extracted from the dataset as the command line Input . The program uses KeyValueTextInputFormat as the input formatter. The key being the AnonID of all the searches and the value being the rest of the search data and the same is passed on to the mapper. The mapper class is defined as follows

**public static class SearchMapper extends Mapper<Text, Text, Text, Text>**

The mapper class checks the user id specified in the command line input with the key and filters based on it. The mapper class also clears the search time column from the value to match the output format.

The reducer class is defined as follows

**public static class SearchReducer extends Reducer<Text ,Text ,Text ,Text>**

The reducer class has very little functionality in this context. It plainly prints it’s input to the output, mainly done to have all the input in a single file otherwise the job could have been a map only job.

## 2.2 Make File

Make files can automate the compilation, execution, and show the search history of the anon ID specified in the make file execution command. The make file expects all the java class files names to be same as the name with which they are submitted. The make file can be executed with the following command with SEARCHID as the argument.

make -f task1make.mk SEARCHID=297170

After execution all the generated output directories, files and jars can be cleaned using command

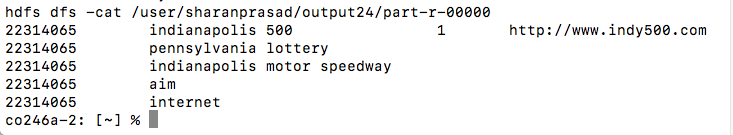
make -f task1make.mk clean

To get the performance of the Hadoop task the make file can be executed with the following command

time make -f task1make.mk SEARCHID=297170

## 2.3 Results

The following results were obtained when the map reduce job was run in the Hadoop cluster for the anon id 22314065



When the standard word count map reduce job was run in the Hadoop cluster with the /tmp/aol as the argument the number of times the word 22314065 was found was 5 and the number of search histories found for the same 22314065 in my above result is also 5 thus verifying the results are correct.



# 3. Task 2

The task is to find various statistics for the search data set. The stats to be found are (1) total number of searches; (2) total number of unique users; and (3) total number of clicks.

## 3.1 Program Design

I have made use of counters provided by map reduce framework to calculate all the statistics in a single program. I have crated three custom counters for the three stats needed to be found.

The input format for the program is same as previous task KeyValueTextInputFormat, the key being the anon ID and the value being the rest of the search data of that ID. The mapper class is defined as follows

**public static class StatsMapper** **extends Mapper<Text ,Text ,Text , LongWritable>**

The mapper class checks if each value has a URL in it. If it has URL it is considered as a click and if no URL is present it is considered as a search and corresponding custom counters are updated. The mapper class also the key and a long writable and passes it to the reducer

The reducer class defined as follows

**public static class StatsReducer extends Reducer<Text, LongWritable, Text, LongWritable>**

takes the key and updates the unique user counter for each key. At the end of the job all the custom counters are written in to a file in the output directory.

## 3.2 Make File

Make file can automate the compilation execution and display all the statistics of the search data set. Similar to previous file the make file expects the name of the java class files provided to be the same. The command to execute the make file

make -f task2make.mk

The following command cleans up all the output directory and jar files generated by the job

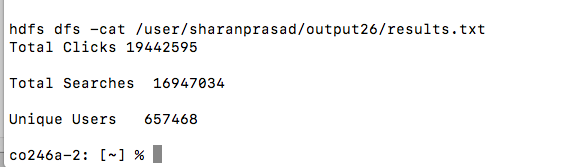
make -f task2make.mk clean

To get the performance of the Hadoop task the make file can be executed with the following command

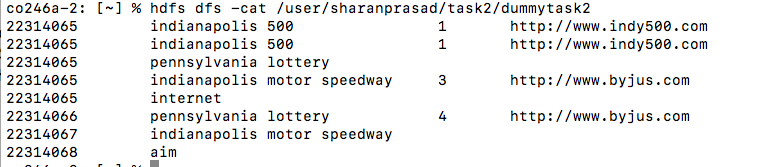
time make -f task2make.mk

## 3.3 Results

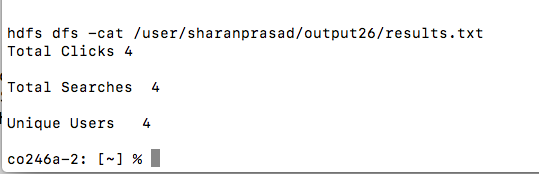
The following results were obtained when the map reduce job was run against the search data set provided.



To verify that the program is working correctly I had made a dummy input file containing search dataset of four unique users with four searches and four clicks.



When the program was run against this input file it provided correct results verifying that the program is working correctly.



# 4. Task 3

## 4.1 Hypothesis

The hypothesis is that as the number of reducers increase the time taken by the map reduce job will decrease because the reducer job should be split equally among all the reducers which run in parallel. However when tested on standard Word count map reduce job the results tend to disagree.

## 4.2 Performance Evaluation Method

The built in time command in Linux was used to measure the time taken by the map reduce job. I made a make which accepts the number of reducers as the input and would pass on the same to word count program.

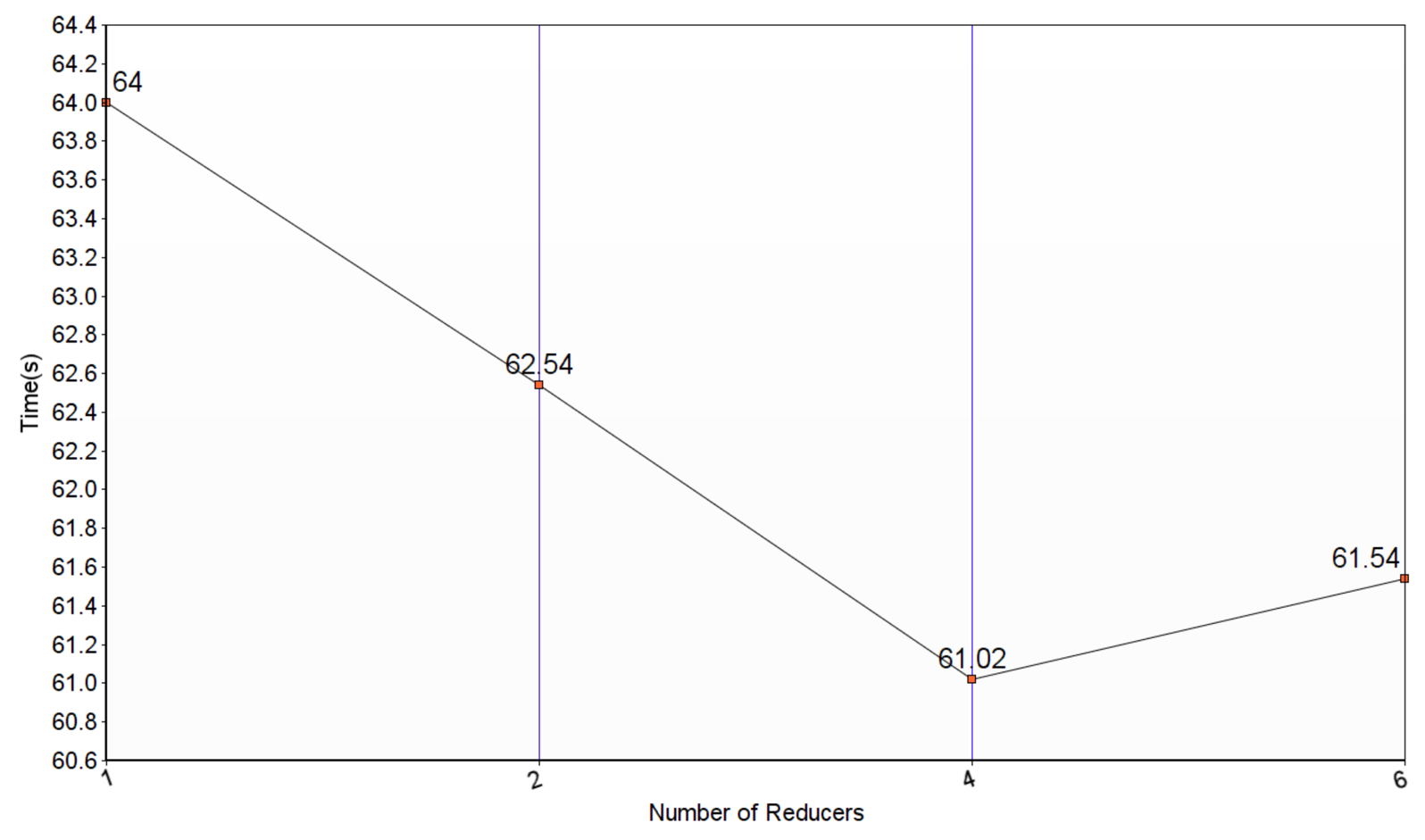
The Execution command looked like this

time make -f performance.mk NUMREDUCER=6

where NUMREDUCER value was passed as an argument to job.setnumReduceTasks() function.

## 4.3 Performance Evaluation Results

## 



## 4.4 Performance Evaluation Discussion

The reason I think that even though the number of reducers were increased the time taken by all the jobs above is almost same because

* As the reducers increase the overload on the network transferring the data from the shuffle phase to the reducers also increases thus taking more time and if the mapping phase output is not huge enough it causes an unnecessary overload.
* The data needs to be split to all the reducers which requires further parsing.
* As the reducers increases the I/O operation also increases as each reducer writes the output to its own file.
* Reducers need to be setup and started
* The numbers of reducers need to be optimum. Both too less and too many reducers cause problems