Mining Big Data - Map-Reduce

Assignment 1 - Hadoop

M. Vincent & A. Phansalkar
a1148120 & a1776879

School of Computer Science, The University of Adelaide

March 29, 2020

Report submitted for **COMP SCI 3306 Mining Big Data** at the School of Computer Science, University of Adelaide

Exercise 1

In this exercise, we consider an attempt to identify suspected evil-doers from a population of 5 billion people, observed over a period of 5,000 days. For a pair of individuals to be suspected of evil-doing, they must visit the same hotel at the same time, on four different days.

The number of hotels is set at 1% of the population, hence there are $500,000$ hotels. The chance of two individuals being at the same hotel on the same day is .0001. The chance that they will visit the same hotel on the same day, is this divided by the number of hotels, which is 2^{-10}. The chance that a pair will visit the same hotel on 4 different days is this to the power of four, which is 1.6^{-39}.

Using the fact that $\binom{n}{2}$ can be approximated by $n^2/2$, we find that the number of pairs of people is $\binom{5^{10}}{2} = 1.25^{19}$, and the number of pairs of days is $\binom{5000}{4} = 2.6^{13}$. The expected number of events that look like evil-doing is then calculated as,

$$1.25^{19} \times 2.6^{13} \times 1.6^{-39} = 5.2^{-7} \tag{1}$$

The expected number of events that look like evil-doing is 5.2^{-7}.

Exercise 2

The output from the tutorial example can be found in the folder Exercise 2/WordCount/Output. The source code, input file, and the executable JAR file used to run the Hadoop job in pseudo-distributed mode, can all be found in the folder Exercise 2/WordCount.

Exercise 3

The output from the FriendRecommend algorithm can be found in the folder Exercise 3/Output.

Exercise 4

Parts 1 and 2

To answer the first two parts of this question, we developed a new MapReduce job. In the map function for this job, the length of each word is used as the key in each key-value pair, rather than the text of the word itself. The value remains 1, as in the WordCount algorithm in the tutorial. The reduce function then sums the values for each key with the same number, giving the final output.

The output from this MapReduce job can be found in folder Exercise 4/WordLength/Output. This output was used to answer the questions below.

1. There are 3102 words of length 10 in FirstInputFile.
2. There are 7019 words of length 4 in FirstInputFile.
3. The longest word in FirstInputFile is 21 characters long, and it appears once.
4. There are 306 words of length 2 in SecondInputFile.
5. There are 105 words of length 5 in SecondInputFile.
6. The most frequent length in SecondInputFile is 0 and it appears 297337 times.

Parts 3 and 4

To answer Parts 3 and 4, we used the original WordCount MapReduce job to generate a unique list of the words appearing in each of the input files, and then took this as the input for the WordLength job.

The WordCount job that was developed using the tutorial produced a list of words that included punctuation and other characters. When this output was run through the WordLength job, it showed a very large number of zero and single character words. Obviously this is not possible, so we tried modifying the WordCount job by removing all characters that were not alphabetical. This produced output that at least passed a simple sense check.

The output generated using both of the MapReduce jobs together can be found in the folder Exxercise 4/WordLength/Output. This output was used to answer the questions below.

1. There are 2263 words of length 10 in FirstInputFile.
2. There are 1911 words of length 4 in the FirstInputFile.
3. The most frequent length in FirstInputFile is 7, and it appears 4908 times.
4. There are 1819 words of length 5 in SecondInputFile.
5. There are 65 words of length 2 in SecondInputFile.
6. The second most frequent length in SecondInputFile is 8 and it appears 2800 times.

Exercise 5

There are mainly few experimental systems which are called Clustera from the University of Wisconsin and Hyracks from the University of California at Irvine extend MapReduce from the workflow of any collected function, with an acyclic graph represent workflow among the functions. Whic is an acyclic flow graph whose arcs a b represents the fact that function as output is input to function b. A moment assume that, function h takes its input from a pre-existing file of the distributed file system. Every h of output elements is passed to at least one of the functions i and j.

There are many advantage which are very helpful on implementing such cascades as a single workflow from those here is one of them . The flow of data which is among tasks, and its replication are able to be managed by the master controller without any need of storing the temporary file which is the output of one MapReduce job from the distributed file system. Now by placing them in the tasks to compute nodes from which that have a blueprint of their input so we can avoid much of the communication that would be very necessary. We stored the blueprint of one MapReduce job and after that we initiated a second MapReduce job which is Hadoop and other map Reduce systems also try to locate Map tasks where a copy of their input is already present.

Here is one more approach to handling the failures while implementation lof recursive algorithms on the basis of computing cluster which is represented by the Pregel system. System shows its own data as a graph. Every node of the graph corresponds roughly to a task . Every graph node gives output messages that are destined for every other nodes of the graph and every graph node processes the inputs it receives from other nodes.

Lets assume that an algorithm is implementing an acyclic network of tasks. These tasks would be Map tasks which feeds to reduce tasks as in a standard MapReduce algorithm, or they could be several MapReduce jobs cascaded, or a more general workflow structure, such as a collection of tasks each of which implementation .The communication cost of each and every task is denoted as the size of the input to the task. This size will be measured in bytes. Although we can use relational database operations as examples we can use the number of tuples as a measure of size. While communication cost often influences our choice of algorithm to use in a cluster-computing environment, we must also be aware of the importance of wall-clock time, the time it takes a parallel algorithm to finish. Using careless reasoning, one could minimize total communication cost by assigning all the work to one task, and thereby minimize total communication.