Hadoop cluster and MapReduce

Introduction

Learning Objectives

This project will encompass the following learning objectives:

1. Explore a large-scale dataset
2. Process a large-scale dataset using MapReduce
3. Use Hadoop MapReduce to run a MapReduce job on the cloud
4. Understand the benefits of frameworks such as MapReduce to analyse large volumes of data

In this part of the Project, we will be processing a much larger portion of the dataset. Using similar filtering techniques used in the previous part, we will aggregate all the data about the page views for all of February 2017.

Further Analysis of the Data

Processing a single file sequentially like we did in the previous assignment does not really answer the question of "What was the most popular page for the month of February 2017" or how many hits did any particular page get on a particular day. To answer this we must:

- aggregate the view counts and
- generate a daily timeline of page views for each article we are interested in

In order to process such a large dataset, we will setup MapReduce job on Hadoop Cluster. You will have to write a simple Map and Reduce program in the language of your choice.
In this part of the project you will understand some of the key aspects of MapReduce and run an MapReduce job flow.

Introduction to MapReduce

The MapReduce programming model, pioneered by Google, is designed to process big data using a large number of machines. In a MapReduce program, data, stored as Key/Value pairs, is processed by a Map function. Mappers output a transformed set of Key/Value pairs, which are subsequently processed by a Reduce function (See detailed pictures in lecture slides)

MapReduce was designed to run in large server farms similar to machines that are deployed at Google's data centers and is proprietary. Hadoop is an open-source implementation of Google's MapReduce, which we already discussed in detail during the lecture. Cloudera Hadoop Cluster is designed for quick provisioning of Hadoop clusters. In this part of the project you will understand some of the key aspects of MapReduce and run a MapReduce job flow.

Example of MapReduce Job Flow: Wordcount

The following video will walk through the process of writing a Streaming MapReduce job flow using Python and Java:

https://www.youtube.com/watch?v=7n0XtbFEBBI

In the example explained in the video above, the map phase Word Count takes a directory of text input files, scans through all of them, outputting a single word (key) and the number "1" (value) per each line. The aggregate reducer takes input, line-by-line, from all the mappers and

sums each instance of that Key/Value pair. The final output is a list with each word, and the number of times that word occurred in all the input files.

You can see the code here

Starting your own Hadoop cluster

Go to Google Cloud Console and create a new cluster on Cloud Dataproc page. Review and set needed configuration for your master and worker nodes.

To run your MapReduce job it is advised to ssh to the master node and provide all job details in terminal.

Writing your own Mappers and Reducers

Now lets get back to processing the wikipedia dataset. In this part you will write your own mappers and reducers to perform the following tasks on the entire 1-month input dataset.

To complete this assignment, you need to finish the following tasks:

1. Design a MapReduce job flow to:
    1. Read all the wikimedia entries for February 2017 from https://dumps.wikimedia.org/other/pageviews/2017/2017-02/ (you will need to download the entries)
    2. Filter out elements based on the rules discussed in previous assignment.
    3. In addition to what you may have filtered, there are some malformed entries which need to be filtered. Malformed entries are entries with missing article name. Make sure that you filter these entries.
    4. Aggregate the pageviews from hourly views to daily views.
    5. Calculate the total pageviews for each article.
    6. For every article that has page-views over 100,000, print the following line as output (**\t** is the tab character):

    <total month views>\t<article name>\t<date1:page views for date1>\t<date2:page views for date2> ...

    7. Getting the input filename from within a Mapper: As the date/time information is encoded in the filename, Hadoop streaming makes the filename available to every map task through the environment variables **mapreduce_map_input_file**, **map_input_file** or **map.input.file**. For example, the filename can be accessed in python using the statement **os.environ["mapreduce_map_input_file"]**, or in Java using the statement **System.getenv("mapreduce_map_input_file")**

2. Once you have designed and tested your MapReduce job flow on a small portion of the dataset, please run it on the entire dataset of February 2017 using MapReduce.
3. Please note the **cluster configuration and runtime in minutes of your solution**.
4. In addition, put top 10 lines from your MapReduce results in the report
5. Once the results are ready, submit your results
6. SECRET NOTE: First 5 students to reach 100% will get 5% bonus for this Lab GOOD LUCK!!!!