Assignment A

Simple analysis of large datasets using MapReduce

Weightage: 100 points (10%), with extra credit of 10 points

Posted date: Mon 25 Jan, 2016

Due date: Wed 3 Feb, 2016 before midnight

Intended Learning Objectives

1. Writing MapReduce applications from scratch, and improving the performance of MR applications using combiners and partitioners.
2. Defining simple analytics over large text and graph datasets, and translating them into MapReduce.
3. Generating large synthetic datasets for evaluation.
4. Coordination of MapReduce runs on a shared cluster. Awareness of memory and CPU usage by Mapper and Reducer tasks.
5. Analysis of MapReduce application logs and results to evaluate performance and scalability.

1. Simple Analytics over Text Data *[50 points]*

Introduction

One of the original motivating applications for MapReduce was to process content that was crawled from the World Wide Web (WWW). *Common Crawl*[1] is a project that crawls the Web every month and provides a dataset of over 1.8 billion webpages, >150 TB in size. While our class cluster cannot handle that datasize for 20+ students concurrently, you will be performing simple analytics over ~0.5% of the entire WWW.

The results of the crawl are stored in a Web ARChive (WARC) file with the HTTP request and response headers and the body for many URLs placed in a single WARC file. Each WARC file is about 4GB in size, and it is further compressed using GZip to be about 1GB in size. About 180 of the 36,000 files from the most recent crawl in Nov 2015[2] are present under **/SE256/CC** folder on HDFS in the cluster.

You are given a starter *Maven* project on the course website that uses a special Hadoop file reader for the GZipped WARC file directly[3], and creates a "split" for each WARC file. The file reader returns an `ArchiveReader` object[4] for each URL, and associated HTTP request and response headers, body, etc., and each map method is called with an archive record object as an input. This is an example of using custom readers over complex datasets to translate them into Key-Value pairs required by MapReduce.

Tasks (10 points each, 50 points total)

a) You are given a list of member and observer countries in the United Nations (`countries.txt` file). Write a MapReduce (MR) job that gives the *number of unique webpages* in which each country is mentioned, and *total number of times* a country is mentioned (i.e. multiple times in

[1] http://commoncrawl.org/the-data/get-started/
[2] http://blog.commoncrawl.org/2015/12/november-2015-crawl-archive-now-available/
[3] https://github.com/commoncrawl/cc-warc-examples
[4] https://github.com/iipc/webarchive-commons

the same webpage). Is there a correlation between the GDP of a country and its frequency of mentions?

b) Write a MR job to return the *Top 20 pairs of countries* that are most frequently mentioned together in the same webpage. Is there a logical reason that they co-occur together?

c) You can mine this web data to approximately find if one entity is *more popular* than the other, e.g. two movies nominated for the Oscar Awards ("Martian" vs. "The Big Short"), two smart phones ("Apple iPhone" vs. "Samsung Galaxy"), or two government programs ("Swacch Bharath" vs. "Jan Dhan"). Define a problem where you want to find which one of two entities are popular, and write a MapReduce job to find the same.

d) One of the reasons that Google crawled the webpage was to build a graph of the WWW, where vertices are URLs and directed edges are links present from the source URL to the sink URL. Write a MR job to generate the *graph of this crawl dataset* as an adjacency list. Each row of output should have a Source URL (ID), and the list of Sink URLs(IDs) that are links present in that webpage. Show how you're able to *run the default MR PageRank algorithm sample* on this graph you generate.

e) For one of the above problems, demonstrate if application exhibits weak scaling. You should show a plot where the X Axis has increasing number of containers (Mappers/Reducers), and Y axis shows the time taken for running the application on an input such that the *input data size per container is constant*. When does the weak scaling stop, and why?

2. Simple Analytics over Graph Data *[20 points+10 points extra credit]*

Introduction

MapReduce can be used for graph processing also, as evident from the use of MR to implement the PageRank algorithm. There are however shortcomings with the MR programming model for graph analytics that we will discuss in later lectures. As an introductory dataset for graph processing, you are given the Twitter social network graph from 2010[5]. This dataset gives the *follower-followee* **edge list** for the *directed* network, i.e., each row is a pair of source and sink vertex IDs, where the source vertex is the user ID of a follower, and the sink vertex ID is the user ID of followee. There are about 40 million unique users (vertices), and 1.46 billion follows relationships (edges). The source and sink vertex IDs are tab separated. The file is hosted at SE256/TWITTER on the HDFS cluster.

Tasks (10 points each)

a) The twitter MR code that has been provided counts the number of vertices and directed edges in the graph. It is inefficient taking more than one hour to run! Write a combiner that helps get the same output results, but in a much faster time of under 10 mins.

b) Write a MapReduce job that collects information from the Twitter graph that helps you test if it follows a Powerlaw distribution for the number incoming and the number of outgoing edges.

c) **EXTRA CREDIT (10 POINTS).** Social networks such as Twitter often recommend members to follow based on the similarity of the set of members you currently follow with the set of members that other users follow. Write one or more MR job(s) that recommends people to follow for a *given user* U_A based on the following logic. Say the given user U_A follows users $\{U_1, U_2, \ldots, U_i\}$. Identify other users $\{U_B, U_C, U_D, \ldots\}$. who follow the same set of users $\{U_1, U_2, \ldots, U_i\}$ as

[5] http://an.kaist.ac.kr/traces/WWW2010.html

U_A, plus additional ones, {U_k,..., U_m}. From these additional followees {U_k,..., U_m}, find the ones who are most frequently followed by {U_B, U_C, U_D,…} and recommend them to user U_A to follow.

3. GENERATING AND SORTING DATASETS *[30 POINTS]*

Introduction

It may not always be possible to get access to large datasets for testing your Big Data application or analytic. So often, there is a need to generate synthetic data that still has some statistical properties that makes the analytic meaningful. MR can be used to generate such large datasets quickly, in addition to writing analytics that can scale on them.

Tasks (10 points each, 30 points total)

a) Say you have to benchmark a new Aadhaar online service that has been developed, and you need to generate *random Aadhaar ID numbers* to simulate transactions by those users on the service. Aadhaar IDs are *12 digit positive random long numbers* in the range of `100,000,000,000` to `999,999,999,999`. You are asked to generate synthetic datasets of different sizes, the number of IDs in each being {$1*10^8$, $2*10^8$,…, $10*10^8$}, which is approximately 1.2 to 12GB of text data since each ID is 12 bytes in size. There can be duplicates IDs in the generated data since the same person could perform a transaction multiple times. Further, you are told that based on statistical observations, the frequency of transactions expected from different ranges of Aadhaar IDs are known and your random data should follow this distribution. Say that for each of the 90 intervals of width 10^{10} that partitions the Aadhaar ID range $(10\times10^{10},11\times10^{10}]$, $(11\times10^{10},12\times10^{10}]$,…, $(99\times10^{10},100\times10^{10}]$, you are given the probability distribution of IDs `(0.0-1.0)` falling in that interval.
Write a MR job that takes as input a probability distribution file with these 90 values, each in the range `(0.0-1.0)`, and as input the number of random numbers to be generated, and generates that many numbers into an output file. Also, write another MR job to output the actual distribution of the generated dataset, as an output file with 90 values.

b) Given a file with randomly generated numbers from (a) above, and also given the probability distribution of the numbers as an input, write an MR application to *sort these numbers* such that the *load on each reducer task is balanced*, i.e., the output file from each reducer has approximately the same number of sorted numbers.

c) Use the datasets from (a) and the sort program from (b) to analyze if MR weakly scales.

Submission Instructions

For all tasks, you should submit the source files integrated with Maven's pom.xml that compiles without error, the log files generated for your job, the list of HDFS output file directories along with the MD5 checksum of its files.

i. Name your source folder as **`username-se256-alpha/`**. Replace "username" you're your cluster account username. Make sure the root of the folder contains the **`pom.xml`**, **`output.csv`** CSV file with path of HDFS output files and their checksums, **`log/`** folder containing logs generated for the final output reported, and your assignment report **`username-se256-alpha-report.pdf`**. Do not include jar or class files in this folder.

ii. Tar your folder into a single file with the filename as below.

```
tar cvf username-se256-alpha-src.tar username-se256-alpha/
```

iii. Calculate its MD5 checksum for the tarred tile

```
md5sum username-se256-alpha-src.tar
```

iv. Zip the tarred file with a *strong password*.

```
zip –e username-se256-alpha-src.tar.zip username-se256-alpha-src.tar
```

Copy the encrypted file to **~se256/submission-alpha/** folder in the head node and email the *password* and *MD5 checksum* to Ravikant by the deadline with the subject line "**assignment submission username-se256-alpha**".

If you upload an unencrypted file to the folder, or you use a weak password, you will get 0 (zero) points.

Rules

- You are working in a shared cluster. So make sure that your source files are kept secure, and are NOT readable by any other student account. Disable group and world read and execute permissions on your home folder. Make sure the jar submitted to Hadoop does NOT contain source files. Do not store source files in /tmp, on HDFS or any other publicly readable locations. Any student who violates these rules will get an automatic 0 (zero) for that assignment.
- Do NOT look into the source code of others, even if others are in violation and source code is "lying around" in some folder. If it is not yours or a shared dataset, do not be curious. If you come across such violations, please bring it to the attention of the TA and Yogesh immediately.
- We will pass the submissions through *plagiarism checks*. If there are noticeable similarities between different submissions, you will get an automatic 0 (zero) for that assignment. Repeat offenders will get grade point reductions or failing grades.

Guidelines

- Given human readable names to your jobs and include your username in the job. This helps identify issues.
- Run your experiments on a subset of the data initially before you run it on the full dataset. Since it may take 10's of minutes or hours to analyze the whole data, limit your full dataset analytics (e.g. on 180 Common Crawl files) to 1 or 2 runs for the final results, and instead use, say 10 files, for testing and debugging.
- Watch your jobs for exceptions and errors. An incorrect map or reduce method can appear to run slowly while it generates a lot of exceptions in the background, overflowing the log files. Monitor the status using yarn application status, and tail the log output file watching for exceptions.
- If you notice someone else's application is taking up a lot of resources, is behaving abnormally or causing the cluster to be unstable, bring it to the attention of the TA and Yogesh. Do NOT kill others' job.
- Do NOT submit a large number of MR jobs at the same time. While the batch system does schedule jobs, do not hog the cluster with many long running jobs. If we find students dominating the cluster, they will be warned initially, and if repeated, their jobs may be involuntarily terminated.