INFSCI 2750: Cloud Computing

Mini Project 1

Objective

The objective of this mini project is to get familiar with setting up the Hadoop system and to start programming in Hadoop. This project can be done in groups of 3 members.

Part 1: Setting up Hadoop in Docker: (30 points)

This part requires you to build container from Docker (https://www.docker.com/) images, which provide the environment you need for part 2 and 3.

First, you need to learn what is a Docker image and how to build it. Docker is an emerging virtualization tool which is similar to the previous tool, Sandbox. It provides several good features to the system developers. You can treat it as the fusion of version control system (VCS) and Programming Language (like JAVA) for virtual machines. You can start with building a small Ubuntu (one of the Linux distributions) image first. Check *get started with Docker for Windows (*https://docs.docker.com/desktop/windows/), *how to build a Docker image (https://docs.docker.com/get-started/)* for details.

Second, based on Ubuntu Docker image, you can build the Hadoop Docker image with the help of the bash files we uploaded. You can start with a basic Hadoop which runs the tasks locally. The major setups of building the image can be done by writing several basic commands in Dockerfile, for example:

```
    #Set the basic image for this Docker image
    FROM ubuntu:16.04
    #Set the id of the maintainer of the image which can be assigned on the Docker Hub
    MAINTAINER your-id
    # Assign the user to run the system calls
    USER root

    # ENV is a basic command in Dockerfile to set the Environment variables
    ENV JAVA_HOME /usr/java/default
    ENV PATH $PATH:$JAVA_HOME/bin
    ENV java_path /usr/lib/jvm/java-8-openjdk-amd64

    # RUN is a basic command in Dockerfile to run system call as in the terminal for buildi
ng the Docker image
    # You can install packages and download Hadoop using RUN command

    # ADD file, hdfs-site.xml in the current directory, to the directory, $HADOOP_PREFIX/e
tc/hadoop/hdfs-site.xml, in the image
    ADD hdfs-site.xml $HADOOP_PREFIX/etc/hadoop/hdfs-site.xml
    # EXPOSE informs Docker that the container listens on the specified network ports at r
untime
     EXPOSE 50010

     # CMD setups a call which is run when the docker image starts, which can be called on
ce in the Dockerfile
     CMD ["/etc/bootstrap.sh", "-d"]
```

Check [Dockerfile reference (https://docs.docker.com/engine/reference/builder/)](https://docs.docker.com/engine/reference/builder/) for basic docker commands, and [*Hadoop: Setting up a Single Node Cluster (https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html)*](https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html) for instruction on configurations.

Third, you can build the Docker image based on the Dockerfile. Run it locally and test it with running a Wordcount job on it. You can check the docker_commands.sh file we uploaded for instruction on how to run bash command inside container.

Your submission should contain the Dockerfile, any other support files like yarn configuration xml, hdfs configuration xml, etc. and one bootstrap script for starting up the yarn services and the hdfs services in a single node mode which is called in the Dockerfile using the “CMD” command.

Part 2: Developing a Hadoop program (20 points)

In Part 2 of the project, you are required to develop a Hadoop program that produces the n-gram frequencies of the text in a given input file. *n-gram* is a contiguous sequence of n characters from a given sequence of text.

An n-gram of size 1 is referred to as a "unigram"; size 2 is a "bigram" (or, less commonly, a "digram"); size 3 is a "trigram". For example, the 2-gram frequency in the text, “Helloworld” is as follows:

He-1, el-1, ll-1, lo-1, ow-1, wo-1, or-1, rl-1, ld-1

Your program should accept *n* as an input parameter and produce the n-gram frequencies in the text as an output file.

Part 3: Developing a Hadoop program to analyze real logs (50 points)

In this part you need to develop MapReduce programs to analyze a real anonymous logs to answer several questions based on the log. The log is in access_log, this log has 2 more fields than the Common Log Format, you can ignore the last two fields and Problems listed below will not involve any information of these two fields.

The log file is in **Common Log Format**:

```
     10.223.157.186 - - [15/Jul/2009:15:50:35 -0700] "GET /assets/js/lowpro.js HTTP/1.1"
200 10469
```

```
%h %l %u %t \"%r\" %>s %b
```

Where:

- %h is the IP address of the client
- %l is identity of the client, or "-" if it's unavailable
- %u is username of the client, or "-" if it's unavailable
- %t is the time that the server finished processing the request. The format is [day/month/year:hour:minute:second zone]
- %r is the request line from the client is given (in double quotes). It contains the method, path, query-string, and protocol or the request.
- %>s is the status code that the server sends back to the client. You will see see mostly status codes 200 (OK - The request has succeeded), 304 (Not Modified) and 404 (Not Found). See more information on status codes **in W3C.org**
- %b is the size of the object returned to the client, in bytes. It will be "-" in case of status code 304.

Problems:

1. How many hits were made to the website directory "/images/smilies/"(including subdirectories and files)?
2. How many hits were made from the IP: 96.32.128.5?
3. How many HTTP request methods are used in this file? What are they?
4. Which path in the website has been hit most? How many hits were made to the path?
5. Which IP accesses the website most? How many accesses were made by it?
6. How many POST request were made?
7. How many requests received a 404 status code?
8. How much data was requested on 19/Dec/2020?
9. List 3 IPs that access the most, and what is the total data flow size of each IP?
10. How much data(in bytes) was successfully(with status code 200) requested on 16/Jan/2022?

For each question, you can write one or two MapReduce programs to get the answer. You can use parts of the code from some questions to build the programs for others.

Your submission should contain the source code for each question, a short report to answer the above questions and a readme file to illustrate the process of running your programs.

Guidelines:

1. Testbed: a portable testbed (Hadoop runs in a local environment) is convenient to program and debug. You can use Docker (*Get started with Docker for Windows*) image

which is built in Part 2 as the testbed. You can also manually setup a Hadoop in Virtualbox with any Linux distribution (Ubuntu, Fedora, etc.) if you want.

2. Test set: use at least one synthetic data for which you exactly know the result – e.g. “Helloworld” and one dataset that is large enough to test the performance of your algorithm – e.g, a dataset of several megabytes in size generated by RandomTextWriter in Hadoop.

3. Debugging: debugging on distributed systems is a challenge. In Hadoop, there are several methods to correct the programs: Unit test (verify your algorithm in the program), Job Tracker (check the status of the job of your running algorithms), Logs (use log to identify the problems). You can check Debug and Tuning Hadoop for some details.

Project Submission: Submit a **single ZIP file** with your *Pitt email ID* as its filename via the Canvas system. The package should contain all your source files and a *readme* file that explains how to execute your program, **for part 2 and part 3, you need to provide your screenshots of the program outputs in the Docker environment you built in part 1.**