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 2 or 3 members.

Each of you is provided with a virtual machine which can be accessed using the IP address and the root password provided to you. For accessing the VMs remotely, you can use any SSH client such as Putty (http://www.putty.org/). Please email the TA, Jinlai Xu (JIX67@pitt.edu) with the names of your team members to receive the IP address and root password of your VMs. The VMs provided to you are dedicated for this course during this term and any use of them for purposes outside of the course project is strictly prohibited.

Part 1: Setting up Hadoop: (50 points)

The first part of the project requires setting up Hadoop on the VMs. A project team consisting of 2 or 3 members will have 2 VMs to build the Hadoop cluster. In each of the VMs, you will need to follow a sequence of preparatory steps to install and run Hadoop. For this project, you can either use Hadoop Yarn or an older version of Hadoop. While the goal is to build a Hadoop cluster of 2 VMs, a good first step is to install and run Hadoop on each of the VMs using the single node Hadoop set up. This process will help make sure that the installation and settings are correct on each of the VMs before setting up the nodes for the 3 node multi-node setup.

A tutorial on setting up Hadoop on a single node is available at: https://www.digitalocean.com/community/tutorials/how-to-install-hadoop-in-stand-alone-mode-on-ubuntu-16-04

After ensuring that the single node setup works properly on all VMs, in the next step, you may change the Hadoop configuration settings to build the 2 VM multimode cluster.

Keyless SSH access is one of the requirements for building the multi-node Hadoop cluster and the following may be helpful in setting up keyless ssh on the VMs: https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys--2

Additionally, the following tutorials may be helpful in setting up the Hadoop Yarn/ Hadoop on the multi-node cluster.

https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html

While referring to these tutorials, please pay attention to the Hadoop versions used in the tutorials. If you are working with a different version of Hadoop, then the installation and configuration steps may not work exactly in the same way.

As part of the demo, you will be required to start the cluster and run the default wordcount program that comes as part of the Hadoop package. Given an input file, the wordcount program prints the number of occurrences of each word in the file.

The second step of the Part 1 requires you to build one Docker image to satisfy the industry requirements in the real production line for how to quick deploy a large scale Hadoop Cluster in several minutes.

Firstly, 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, how to build a Docker image* for details.

Secondly, you can based on the previous Ubuntu Docker image to build the Hadoop Docker image with the help of the steps in Part 1. 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 the this Docker image
FROM ubuntu:14.04
#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
# passwordless ssh
# RUN is a basic command in Dockerfile to run system call as in the terminal for buildi
ng the Docker image.
RUN ssh-keygen -q -N "" -t dsa -f /etc/ssh/ssh_host_dsa_key
RUN ssh-keygen -q -N "" -t rsa -f /etc/ssh/ssh_host_rsa_key
RUN ssh-keygen -q -N "" -t rsa -f /root/.ssh/id_rsa
RUN cp /root/.ssh/id_rsa.pub /root/.ssh/authorized_keys
# ENV is a basic command in Dockerfile to set the Environment variables
ENV JAVA_HOME /usr/java/default
ENV PATH $PATH:$JAVA_HOME/bin
# ADD file, hdfs-site.xml in the current directory, to the directory,  $HADOOP_PREFIX/
etc/hadoop/hdfs-site.xml, in the image
ADD hdfs-site.xml $HADOOP_PREFIX/etc/hadoop/hdfs-site.xml
#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"]
# EXPOSE informs Docker that the container listens on the specified network ports at r
untime
EXPOSE 50010
```

Check Dockerfile reference, *Hadoop: Setting up a Single Node Cluster* for some help.

Thirdly, build the Docker image based on the Dockerfile. Run it locally and test it with running a Wordcount job on it.

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

Part 3: Developing a Hadoop program (25 points)

In Part 3 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 4: Developing a Hadoop program to analyze real logs (25 points)

In this part you need to develop several MapReduce programs to analyze a real anonymous logs to answer several questions based on the log. The log is in access_log.zip.

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 item "/assets/img/home-logo.png"?
2. How many hits were made from the IP: 10.153.239.5
3. Which path in the website has been hit most? How many hits were made to the path?
4. Which IP accesses the website most? How many accesses were made by it?

For each question, you can write one or two MapReduce programs to get the answer. You can use parts of the code from the easier two questions (question 1 and 2) to build the programs for question 3 and 4.

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.) with the steps in Part 1.
2. Test set: at least one synthetic data that you exactly know the result like "Helloworld". And one data that is big enough to test the performance of your algorithm like several megabytes data generated by RandomTextWriter in Hadoop.
3. Debug: debug on distributed system 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 CourseWeb system. The package should contain all your source files and a *readme* file that explains how to execute your program in Hadoop. You also need to demonstrate your Hadoop cluster setup and the n-gram frequency computation program to the TA in person, where you may be asked to explain about different parts of your code. Your demos will be performed on the VMs that you have access to but you may be provided with a different test input.