Assignment 2

Lincoln Brown

DSC650-T301

Professor Ismaily

# Week 2 Assignment: Dive into HDFS and MapReduce

### Objective: Familiarize with the core functionalities of HDFS and get a practical understanding of MapReduce.

#### **1. Environment Initialization**

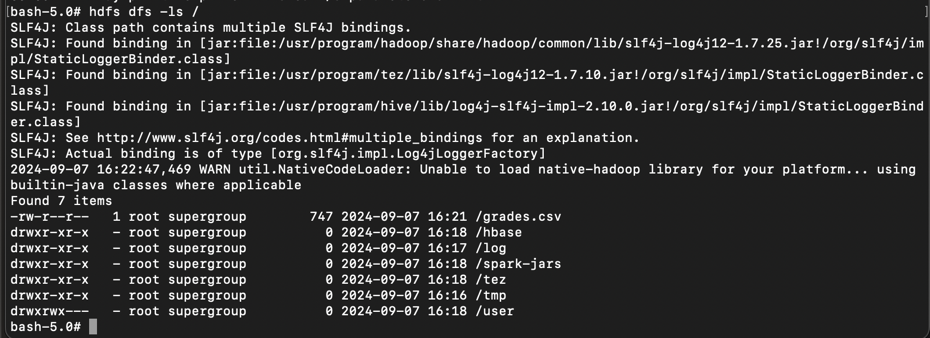
* Start by navigating to the required directory and initiating the Docker containers:
* cd bellevue-bigdata  
  cd hadoop-hive-spark-hbase  
  docker-compose up -d
* If you’re using Google Cloud, remember to set up port forwarding as outlined in the Week 1 assignment.
* Access the master container:
* docker-compose exec master bash

#### **2. Deep Dive into HDFS**

* Check the HDFS report:
* hdfs dfsadmin -report
* A screenshot of a computer

  Description automatically generated

**Deliverable:** Screenshot of the output.

* Load the grades.csv into HDFS:
* hdfs dfs -put /data/grades.csv /
* Verify that the data has been loaded:
* hdfs dfs -ls /  
    
  

**Deliverable:** Screenshot proving the data has been loaded.

* Exit the master Docker container:
* CTRL+D or exit
* SSH into each of the 3 worker nodes and verify the data:
* docker-compose exec worker1 bash  
  hdfs dfs -ls /  
  CTRL+D or exit  
    
  docker-compose exec worker2 bash  
  hdfs dfs -ls /  
  CTRL+D or exit
* All worker nodes should display the grades.csv file.
* Re-enter the master container:
* docker-compose exec master bash
* Explore more HDFS commands:
* hdfs dfs -help
* A screenshot of a computer screen

  Description automatically generated

A screen shot of a computer

Description automatically generated

Execute three other HDFS commands of your choice and observe their outputs.

A computer screen shot of a black screen

Description automatically generated

A computer screen shot of a computer program

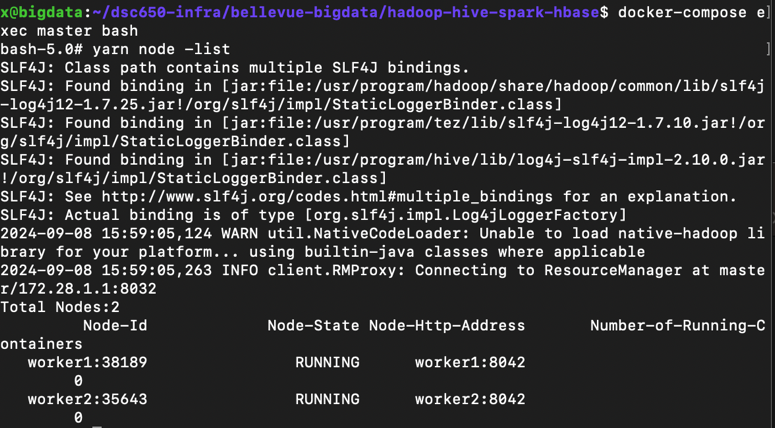
Description automatically generated

A computer screen shot of a computer

Description automatically generated

**Deliverable:** Screenshots of the three chosen HDFS command outputs.

#### **3. Introduction to YARN**

* Inside the master container, inspect the YARN nodes:
* yarn node -list
* 

**Deliverable:** Screenshot of the results.

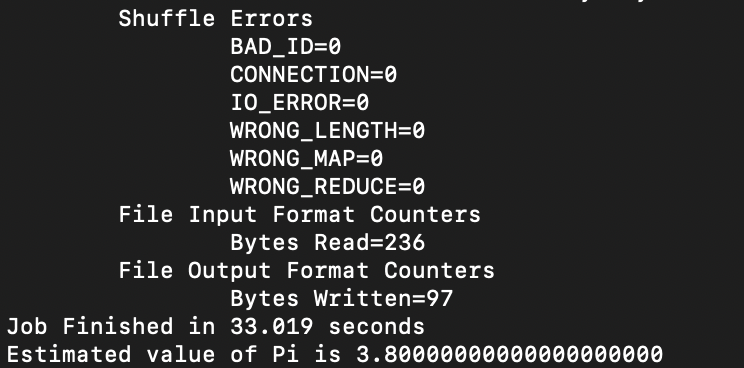
* Understand the yarn.scheduler.maximum-allocation-mb property. This is the maximum memory capacity available for a single container.
* Modify the maximum memory allocation:
* sed -i "/<name>yarn.scheduler.maximum-allocation-mb<\/name>/,/<\/property>/s/<value>.\*<\/value>/<value>2048<\/value>/" /usr/program/hadoop/etc/hadoop/yarn-site.xml
* Restart the ResourceManager:
* yarn --daemon stop resourcemanager  
  yarn --daemon start resourcemanager

**A screenshot of a computer

Description automatically generated**

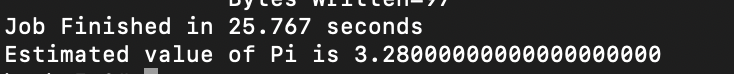
**Deliverable:** Screenshot from the YARN UI showing the updated maximum memory (2048 MB).

#### **4. Experimenting with MapReduce**

* Run the example MapReduce Pi job:
* libjars=$(find /usr/program/hadoop/share/hadoop/mapreduce -name "\*.jar" | tr '\n' ',')
* hadoop jar /usr/program/hadoop/share/hadoop/mapreduce/hadoop-mapreduce-examples-3.2.3.jar pi -libjars ${libjars} 2 10
* Calculating the value of pi with 2 maps and 10 samples per map yielded the result 3.80:
* 
* Why does the MR job produce a different result?

The MR job produces a different result because it is using multiple quasi Monte Carlo simulations to calculate the value of pi. Since we are working with limited resources and only gave it two maps and 10 samples per map, the result is not as accurate as if it would be if it had a larger sample size.

* How does the role of sample size impact the accuracy of the Pi calculation?
* The MapReduce job is using the Quasi Monte Carlo method to calculate the value of Pi. This method relies on approximating pi using multiple random samples. The larger the sample size, the more accurate the result will be.
* Increasing the number of maps from 2 to 4 improves the result from 3.80 to 3.40:
* A computer screen with white text

  Description automatically generated  
  Similarly, increasing the number of samples per map from 10 to 20, produces the same result, 3.40.
* Increasing the number of samples per map from 20 to 25 further improves the accuracy, resulting in a value of 3.28:  
  
* Therefore, this confirms that increasing the sample size improves the accuracy of the calculation.
* How does the quality of the random number generation process affect accuracy?
* The quality of the random number generation process is an important factor here. In this assignment, we are using QuaiMonteCarlo method which uses a Halton Sequence for quasi-random number generation. Using a Halton Sequence in Monte Carlo simulations is a good choice because they fill sample space uniformly, which is desirable over the potentially skewed distributions that could occur with pseudo-random generated distributions.
* Having a uniformly random space generated with a Halton sequence can help produce more reliable accuracy with fewer samples than purely random distributions due to the potential for skewing in purely random distributions.

**Deliverable:** A summary of the result and its significance.

## Shutting Down

Ensure all Docker containers are turned off with docker-compose down for each directory. If you’re using google cloud, please shut down your virtual machine to preserve cloud costs.