**Assignment 4:**

**The software/framework I used:**

Leader-election: ZooKeeper

Multi-server processing: Hadoop 2.x.x

Word Count: MapReduce 🡪 I builded a java maven project

Container: docker

**What I do:**

I created a Hadoop + Zookeeper cluster which includes 2 masters and 3 slaves. Each mater has a namenode. Since I have 2 masters in my cluster, I can set a active namenode and a standby namenode so once a failure occurs, such as a machine crash or the machine needs to be upgraded and maintained, I can quickly switch the namenode to another machine through an example. Also, since I only have one machine(my laptop), I deployed the cluster hosted on Docker container.

|  |  |
| --- | --- |
| Machine(container) | Usage |
| Master0 | namenode |
| Master1 | namenode |
| Slave0 | datanode |
| Slave1 | datanode |
| Slave2 | datanode |

**Step1:** Build Hadoop + Zookeeper Cluster

1. I created 5 containers, 2 maters and 3 slaves. Here’s the screenshot:

A screen shot of a computer

Description automatically generated

1. Each container has a Hadoop and zookeeper environment.

Start the master machine.

(1). ***master0***

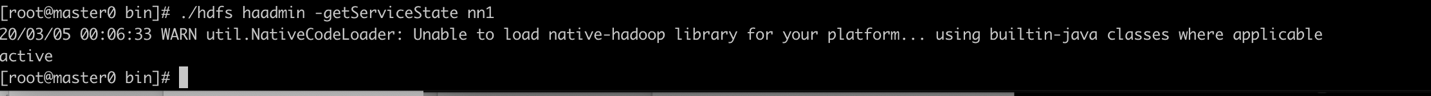
The screenshot of jps for master0:

A close up of a logo

Description automatically generated

As shown above, we can see the master0 is controlled by ZooKeeper. (DFSZKFailloverController)

Master0 is the active Namenode, here’s the screenshot of the namenode status:





Also from http://127.0.0.1:50070 we can check the status of the namenode:

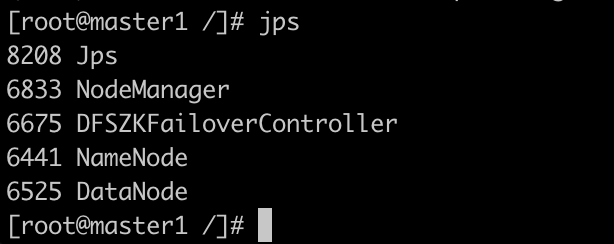
A screenshot of a cell phone

Description automatically generated



(2). ***master1***

The screenshot of jps for master1:



As shown above, we can see the master1 is controlled by ZooKeeper. (DFSZKFailloverController)

Master1 is the standby Namenode, here’s the screenshot of the namenode status:

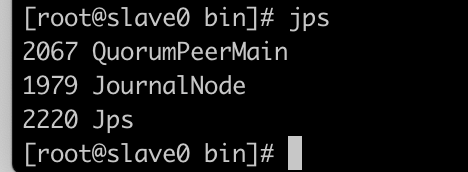
A close up of a logo

Description automatically generated



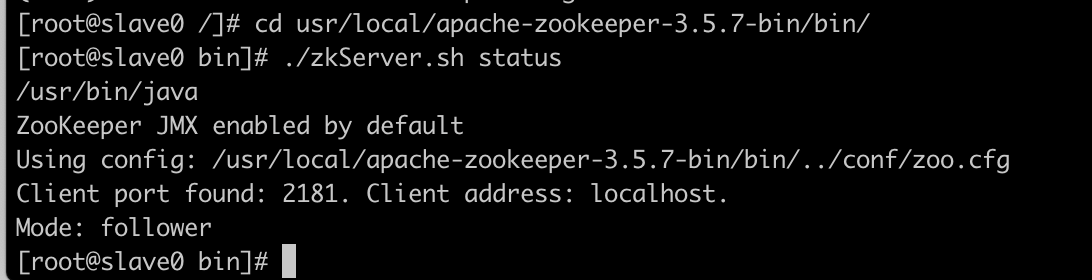
(3). ***slave0***

The screenshot of jps for slave0:



As shown above, we can see the slave0 is controlled by ZooKeeper. (QuorumPeerMain)

Let’s check the status of:





This slave is a follower.

(4). ***slave1***

The screenshot of jps for slave1:



As shown above, we can see the slave1 is controlled by ZooKeeper. (QuorumPeerMain)

Let’s check the status of:

A screenshot of a cell phone

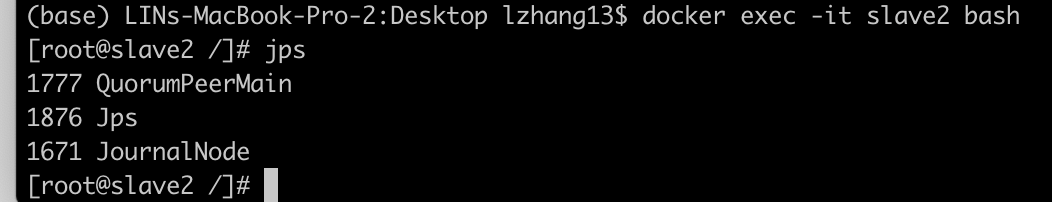
Description automatically generated



This slave is a leader.

(5). ***slave2***

The screenshot of jps for slave2:



As shown above, we can see the slave2 is controlled by ZooKeeper. (QuorumPeerMain)

Let’s check the status of:

A screenshot of a cell phone

Description automatically generated



This slave is a follower.

**Step2**: Word Count in the cluster environment

(Since master0 is the active namenode, all operations below are in master0 container)

1. INPUT:

I copied all book content from the https://www.gutenberg.org/files/11/11.txt, saved it in input.txt&input2.txt&input3.txt file and put the file into HDFS and the path is /assign4

Here is the screenshot of the file in HDFS:

A close up of a logo

Description automatically generated



1. OUTPUT:

I created a java maven project called ‘Assign4MR’ LOCALLY, build artifacts and generate the .jar file. Then I copied the .jar file into the master0 container.

Then created the shell file to process MapReduce using hadoop.

M project requires input and output file path of the mapreduce operation, so I set the input file path as ‘/assign4’, output file path as ‘/assign4\_output’.

Here’s the screenshot when I run the .jar file under HDFS:

A screenshot of a cell phone

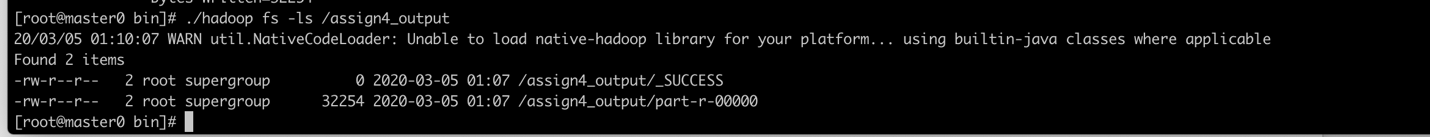
Description automatically generated



A close up of a logo

Description automatically generated

And the output has been generated to /aasign4\_output, here is the screenshot:





We can see that the output file is part-r-00000.

Let’s cat the part-r-00000, here’s the screenshot:

A screenshot of a cell phone

Description automatically generated

Here we got the unsorted word count.

Then, re-run the MapReduce again, this time I use the .jar file which builds MapReduce by sorting the word count:

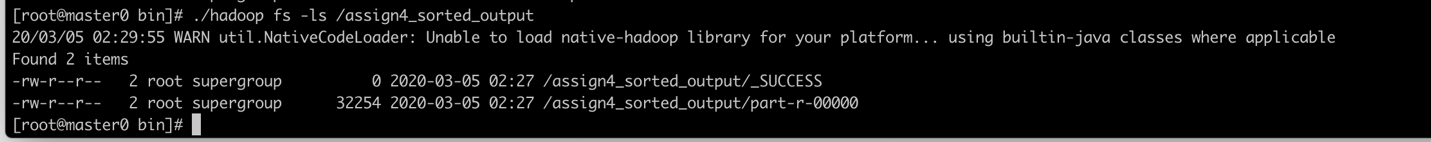
A screenshot of a cell phone

Description automatically generated

A close up of a logo

Description automatically generated

And the output has been generated to /assign4\_sorted\_output, here is the screenshot:





We can see that the output file is part-r-00000.

Let’s cat the part-r-00000, here’s the screenshot:

Words with most frequency – top 20:

A close up of a logo

Description automatically generated

Words with least frequency – top 20:

A screenshot of a cell phone

Description automatically generated

(actually, the words are sorted alphabetically, there could be more words only appeared once).