Task 6 - Use orchestration

Hongyi Zhang <hongyiz@kth.se> Lida Liu <lidal@kth.se>

I. Introduction

The Hadoop Distributed File System is designed to be suitable for distributed file systems running on commodity hardware. There are two types of nodes in the HDFS architecture, one is NameNode, also called "metadata node". The other is DataNode, also called "data node". These two types of nodes respectively perform the execution nodes of the specific tasks of Master and Worker. The general design idea: divide and conquer - distribute large batches of files on a large number of independent servers, in order to take the divide and conquer way of computing and analyzing massive data.

II. Implementation of HopsFS

We have encountered several troubles during the implementation. The version of hops-util located in pom.xml in vt18-id2210-cloud-master/hops/ should be 0.1.0, not ${hops.version}. Otherwise the maven will throw the error since it cannot find the corresponding version in SICS Release Repository.

And while we are doing ssh and scp operation, we could also include the private ssh key to accomplish the command.

After all the configuration and installation have been finished, we can get the result as followed.

```
ubuntu@gceharry-hopsfs-nn-1:~$ /srv/hops/hadoop/bin/hdfs dfs -ls /test
Found 1 items
-rw-r--r--   1 ubuntu ubuntu          0 2018-05-17 12:30 /test/test.txt
ubuntu@gceharry-hopsfs-nn-1:~$
```

Fig. 1. The file created in HopeFS

Also there are three virtual machines deployed on Google Cloud Platform.

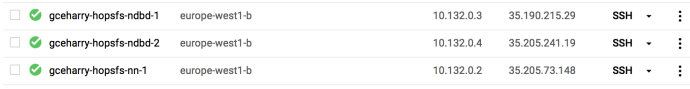

Fig. 2. Virtual Machines

Te following figure is the initial status of three machines on the karamel terminal.

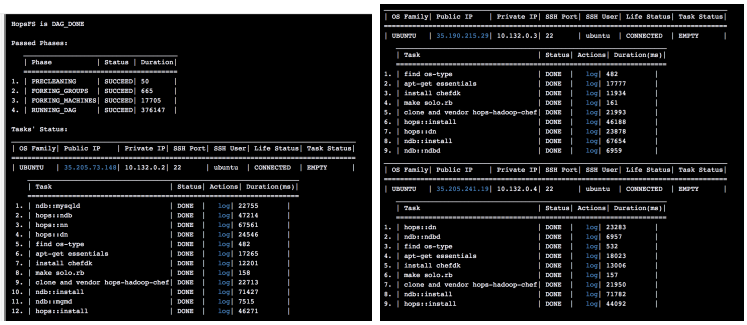

Fig. 3. Status of Virtual Machines

III. Benchmark of HopsFS

In this section we will benchmark the throughput of the HopeFS System. We will use 1o files with 557.47mb in total. With command of the File System (FS) shell, we can easily benchmark the reading and writing ability for the HDFS. The code for this section can also be found on Github.

The following figure shows all the files which has been stored in the HopeFS.

```
ubuntu@gceharry-hopsfs-nn-1:~$ /srv/hops/hadoop/bin/hdfs dfs -ls /rwtest
Found 10 items
-rw-r--r--   3 ubuntu ubuntu  110166656 2018-05-17 13:59 /rwtest/Advanced_Internetworking.zip
-rw-r--r--   3 ubuntu ubuntu    5180710 2018-05-17 13:59 /rwtest/ID2210-Blockchains.pdf
-rw-r--r--   3 ubuntu ubuntu   16603011 2018-05-17 13:59 /rwtest/ID2210.zip
-rw-r--r--   3 ubuntu ubuntu   29334049 2018-05-17 13:59 /rwtest/International_Consulting.zip
-rw-r--r--   3 ubuntu ubuntu   99783425 2018-05-17 13:59 /rwtest/Internet_Security.pdf
-rw-r--r--   3 ubuntu ubuntu   58556112 2018-05-17 13:59 /rwtest/JAVA.zip
-rw-r--r--   3 ubuntu ubuntu   80482229 2018-05-17 13:59 /rwtest/Java_Programming.pdf
-rw-r--r--   3 ubuntu ubuntu    8126464 2018-05-17 13:59 /rwtest/Line.sdf
-rw-r--r--   3 ubuntu ubuntu    9156764 2018-05-17 13:59 /rwtest/Project.zip
-rw-r--r--   3 ubuntu ubuntu  167177702 2018-05-17 13:59 /rwtest/Research_Methodology_and_Scientific_Writing.zip
```

Fig. 4. Files in HopeFS

Here is the result for reading test. For reading operation, we are using FS shell copyToLocal command *hdfs dfs -copyToLocal [-ignorecrc] [-crc] URI <localdst>*. At first we want to use the moveToLocal but this command haven't been implemented yet.

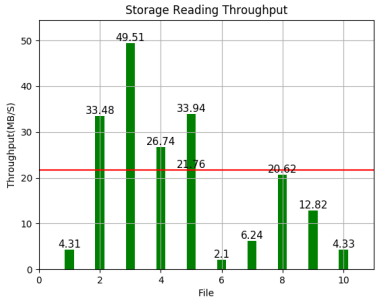

Fig. 5. Reading Throughput

The throughput is pretty high. The average reading rate is 21.76mb/s while the maximum can reach up to 49.51mb/s. The maximum rate is the transmission for the file Research_Methodology_and_Scientific_Writing.zip, which is also the maximum-size file. That means the HopeFS is really suitable to handle large files. You can also see the result in writing operation.

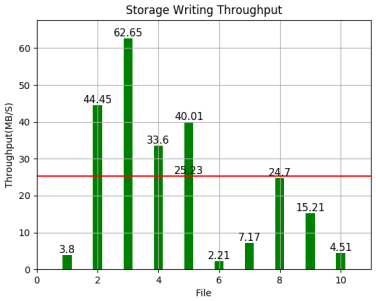

Fig. 6. Writing Throughput

The writing rate is really similar to reading process. The average is 25.23mb/s while the maximum can be 62.65mb/s also for that largest file. Besides that, all the large size file have a pretty high speed rate.

As HDFS is a Mater/Slave architecture. From the end user's point of view, it can perform CRUD (Create, Read, Update, and Delete) operations on a file through the directory path just like a traditional file system. However, due to the nature of distributed storage, HDFS clusters have a NameNode and some DataNodes. The NameNode manages the metadata of the file system and the DataNode stores the actual data. The client accesses the file system through interactions with the NameNode and DataNodes. The client contacts the NameNode to get the metadata of the file, and the real file I/O operation is directly interacting with the DataNode.

Since HDFS is generally used for "write once, read many times". We think it is not suitable for real-time interaction, and is not suitable for storing a large number of small files.

REFERENCES

[1] Apache Hadoop [Online]. Available: *http://hadoop.apache.org*
[2] Apache Hadoop Project Dist POM, Apache Hadoop 2.4.1 [Online]. Available: *https://hadoop.apache.org/docs/r2.4.1/hadoop-project-dist/hadoop-common/FileSystemShell.html*