EE327 Project
Acemap-Hadoop

Yuan Yao

June 2016

Contents

1 Introduction

1.1 Background

Since nowadays the amount of the data on the internet grows much larger than before, it is necessary to use some corresponding platform to computer and process such big data. Hadoop is such a platform which consists of hdfs and mapreduce framework. Similarly, our acemap also need to handle big data, therefore our search engine group began to construct a Hadoop ecosystem and I am mainly responsible for this part.

1.2 Goal

In this project, I mainly focus on three parts : construction of Hadoop cluster, using Mahout to construct a LDA model and distributed crawler.

2 Environment

The following is the environment on which I worked for this project. In addition,

No. of Server	3
Total RAM	384G
Hadoop Version	2.7.2
Python Version	2.7

there are some software need to be installed before.

2.1 Hadoop

Hadoop is an open source distributed data processing platform, it consists of hdfs and mapreduce framework.

1. HDFS, a distributed file system.
2. YARN, yet another resource negotiator, used intead of the original mapreduce framework.

The installation and configuration steps are :

1. Install JAVA.
2. Define an account.
3. Generate SSH key pairs and send public key to each other server.
4. Install and configure Hadoop on master node.
5. Copy the Hadoop directory to the slave nodes.
6. Start HDFS and MapReduce and use jps to monitor them.

2.2 Mahout

Mahout is an open source machine learning library based on Hadoop.

2.3 Hadoop Streaming

Hadoop Streaming is a framework which allows running program written by any other language. Then test if the AVD can run normally in the default project. The SDK's version I used is 6.0 and the version of building tools is 23.0.3.

3 Implementation

3.1 Hadoop

The installation and configuration is mentioned before.

3.2 Mahout

There are three main steps :

1. Use sqoop to transfer data from database to HDFS.
2. Adjust the format of data to match the input format of mahout.
3. Use mahout to extract the topic model.

3.3 Distributed Crawler

The implementation of the distributed crawler is as follow :

1. Write a standalone crawler program in Python. The code had been put on my github, the link is :`https://github.com/rozentill/PaperCrawler/tree/master/crawlers/ideas.repec.org`
2. Split the crawler program into mapper.py and reducer.py.
3. Put the input and output directory onto HDFS.
4. Use Hadoop Streaming to run the mapper and reducer program.

3.4 Map Reduce

It is essential to explain the principle of mapreduce. It mainly consists of two parts :

1. Mapper, computer and process the separated data.
2. Reducer, sum up the results of all the mappers.

The idea of mapreduce is as follow :

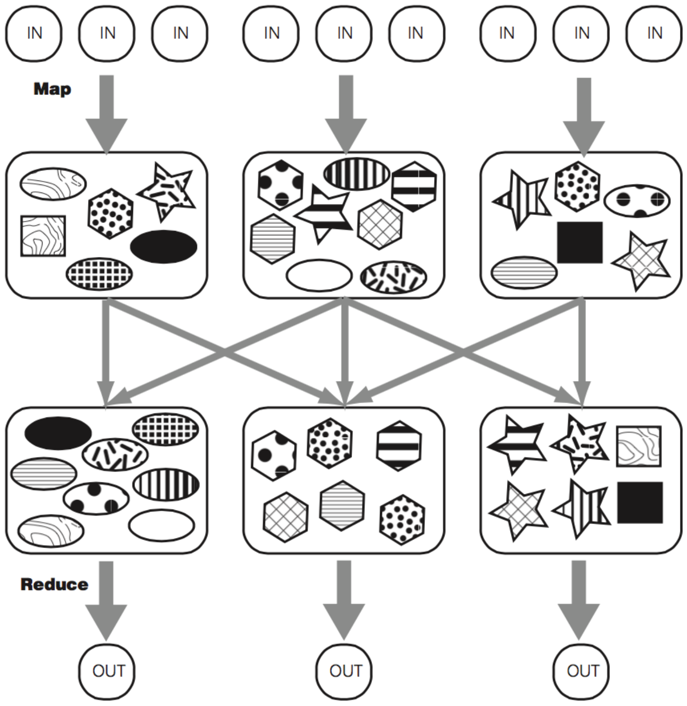


Figure 1: MapReduce

In practice, there is a trick when combining mapreduce with hdfs. The global idea of Hadoop is as foolow :

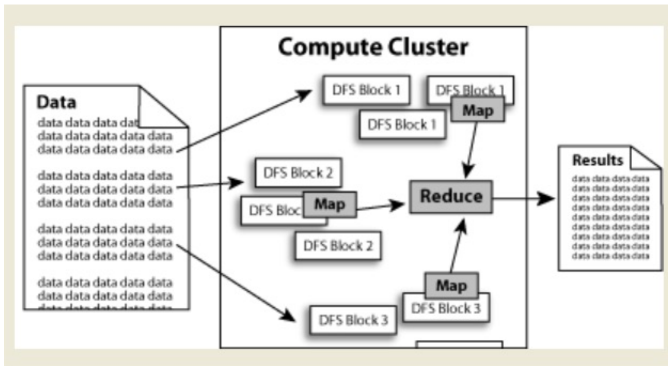


Figure 2: HDFS with MapReduce

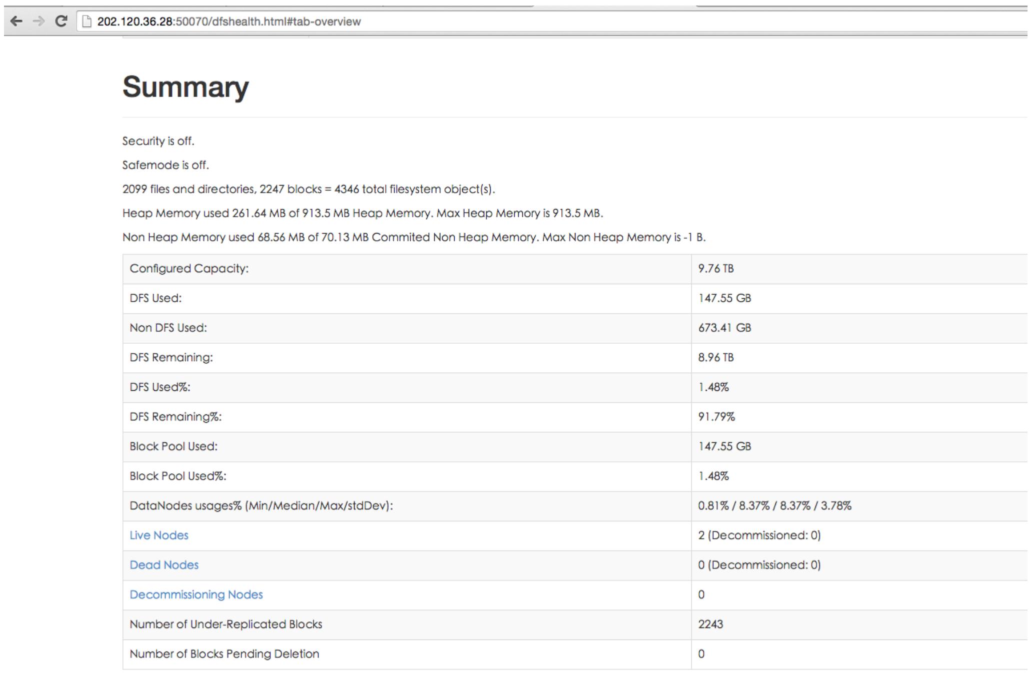
This figure means every mapper is closed to the data they access, which can reduce the cost of access.

4 Result

The result of the LDA was removed since the cluster had been re-installed once. Therefore here are the other two parts' results.

4.1 Hadoop

After the configuraion of Hadoop, we can use web pages to monitor the HDFS and YARN. The following are the results of HDFS: This means we can use port



202.120.36.28:50070/dfshealth.html#tab-overview

Summary

Security is off.

Safemode is off.

2099 files and directories, 2247 blocks = 4346 total filesystem object(s).

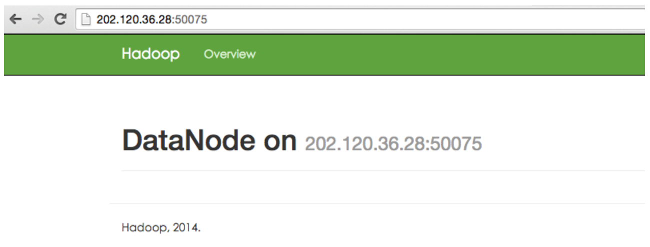
Heap Memory used 261.64 MB of 913.5 MB Heap Memory. Max Heap Memory is 913.5 MB.

Non Heap Memory used 68.56 MB of 70.13 MB Commited Non Heap Memory. Max Non Heap Memory is -1 B.

Configured Capacity:	9.76 TB
DFS Used:	147.55 GB
Non DFS Used:	673.41 GB
DFS Remaining:	8.96 TB
DFS Used%:	1.48%
DFS Remaining%:	91.79%
Block Pool Used:	147.55 GB
Block Pool Used%:	1.48%
DataNodes usages% (Min/Median/Max/stdDev):	0.81% / 8.37% / 8.37% / 3.78%
Live Nodes	2 (Decommissioned: 0)
Dead Nodes	0 (Decommissioned: 0)
Decommissioning Nodes	0
Number of Under-Replicated Blocks	2243
Number of Blocks Pending Deletion	0

Figure 3: Name Node

50070 to monitor the whole usage and status of HDFS and namenode.



202.120.36.28:50075

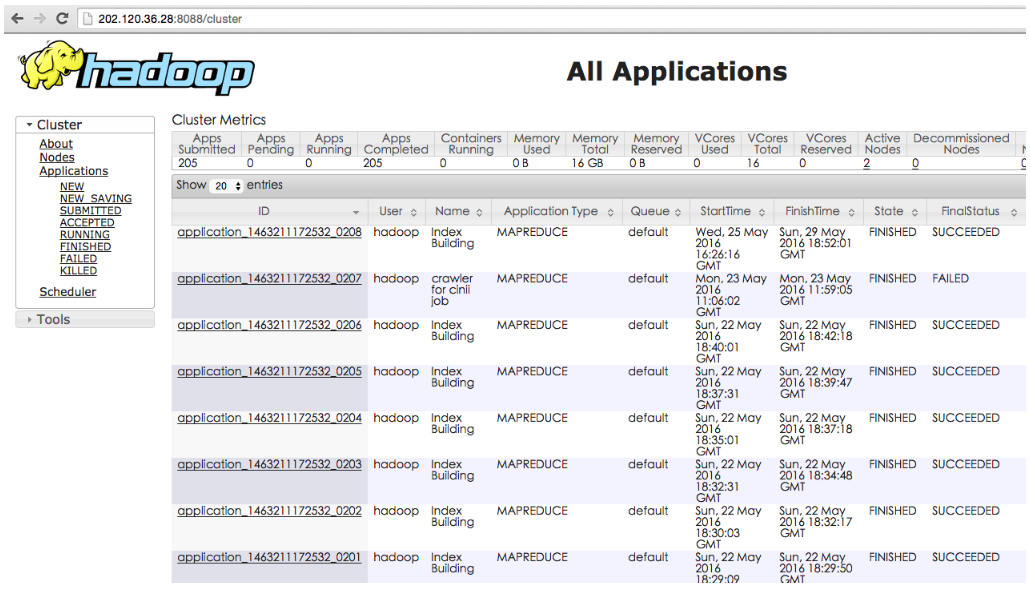
Hadoop Overview

DataNode on 202.120.36.28:50075

Hadoop, 2014.

Figure 4: Data Node

This means we can use port 50075 to monitor the data node(one of them). Also the following shows we can use port 8088 to monitor all the jobs running or ran on the cluster.



202.120.36.28:8088/cluster

All Applications

Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes
205	0	0	205	0	0 B	16 GB	0 B	0	16	0	2	0

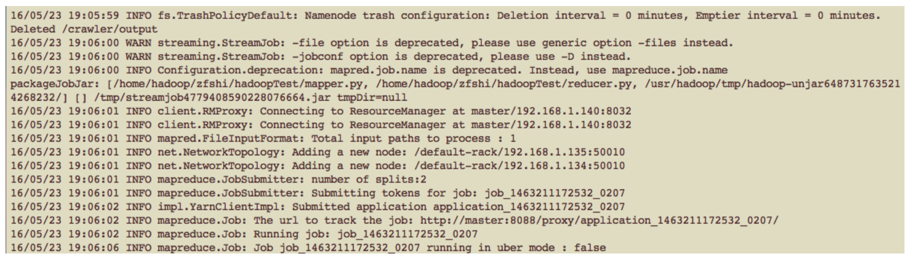
Show 20 entries

ID	User	Name	Application Type	Queue	StartTime	FinishTime	State	FinalStatus
application_1463211172532_0208	hadoop	Index Building	MAPREDUCE	default	Wed, 25 May 2016 16:26:16 GMT	Sun, 29 May 2016 18:52:01 GMT	FINISHED	SUCCEEDED
application_1463211172532_0207	hadoop	crawler for cinii job	MAPREDUCE	default	Mon, 23 May 2016 11:06:02 GMT	Mon, 23 May 2016 11:59:05 GMT	FINISHED	FAILED
application_1463211172532_0206	hadoop	Index Building	MAPREDUCE	default	Sun, 22 May 2016 18:40:01 GMT	Sun, 22 May 2016 18:42:18 GMT	FINISHED	SUCCEEDED
application_1463211172532_0205	hadoop	Index Building	MAPREDUCE	default	Sun, 22 May 2016 18:37:31 GMT	Sun, 22 May 2016 18:39:47 GMT	FINISHED	SUCCEEDED
application_1463211172532_0204	hadoop	Index Building	MAPREDUCE	default	Sun, 22 May 2016 18:35:01 GMT	Sun, 22 May 2016 18:37:18 GMT	FINISHED	SUCCEEDED
application_1463211172532_0203	hadoop	Index Building	MAPREDUCE	default	Sun, 22 May 2016 18:32:31 GMT	Sun, 22 May 2016 18:34:48 GMT	FINISHED	SUCCEEDED
application_1463211172532_0202	hadoop	Index Building	MAPREDUCE	default	Sun, 22 May 2016 18:30:03 GMT	Sun, 22 May 2016 18:32:17 GMT	FINISHED	SUCCEEDED
application_1463211172532_0201	hadoop	Index Building	MAPREDUCE	default	Sun, 22 May 2016 18:29:09	Sun, 22 May 2016 18:29:50 GMT	FINISHED	SUCCEEDED

Figure 5: Job Tracker

4.2 Distributed Crawler

The running result of distributed crawler after using hadoop streaming is as follow :



```
16/05/23 19:05:59 INFO fs.TrashPolicyDefault: Namenode trash configuration: Deletion interval = 0 minutes, Emptier interval = 0 minutes.
Deleted /crawler/output
16/05/23 19:06:00 WARN streaming.StreamJob: -file option is deprecated, please use generic option -files instead.
16/05/23 19:06:00 WARN streaming.StreamJob: -jobconf option is deprecated, please use -D instead.
16/05/23 19:06:00 INFO Configuration.deprecation: mapred.job.name is deprecated. Instead, use mapreduce.job.name
packageJobJar: [/home/hadoop/zfshi/hadoopTest/mapper.py, /home/hadoop/zfshi/hadoopTest/reducer.py, /usr/hadoop/tmp/hadoop-unjar648731763521
4268232/] [] /tmp/streamjob4779408590228076664.jar tmpDir=null
16/05/23 19:06:01 INFO client.RMProxy: Connecting to ResourceManager at master/192.168.1.140:8032
16/05/23 19:06:01 INFO client.RMProxy: Connecting to ResourceManager at master/192.168.1.140:8032
16/05/23 19:06:01 INFO mapred.FileInputFormat: Total input paths to process : 1
16/05/23 19:06:01 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.135:50010
16/05/23 19:06:01 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.1.134:50010
16/05/23 19:06:01 INFO mapreduce.JobSubmitter: number of splits:2
16/05/23 19:06:01 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1463211172532_0207
16/05/23 19:06:02 INFO impl.YarnClientImpl: Submitted application application_1463211172532_0207
16/05/23 19:06:02 INFO mapreduce.Job: The url to track the job: http://master:8088/proxy/application_1463211172532_0207/
16/05/23 19:06:02 INFO mapreduce.Job: Running job: job_1463211172532_0207
16/05/23 19:06:06 INFO mapreduce.Job: Job job_1463211172532_0207 running in uber mode : false
```

It shows the number of mappers generated.

5 Conclusion

This project is very helpful and challenging, it took me much time but I think it is deserved.