CMPT 756

Systems For Big Data

Cloud Computing with Apache Cloudstack and Cloudera Cluster Creation Report

Submitted By,

Namita Shah

Shawn Anderson

Sethuraman Annamalai

Introduction

Utilization of cloud computing is an essential skill for the contemporary Data Scientist. In this report we document the process of deploying a functional cloud based cluster of virtual machines, software ready. Apache CloudStack is used for virtual instance creation, and Cloudera for cluster deployment and software installation. We address security and scalability concerns to be aware of during installation. We also highlight many pitfalls and gotchas that we encounter, in hopes that this report may serve as a reference for those that wish to follow a similar route in cluster deployment.

Background

I. Cloud Computing

Cloud computing is an economic paradigm of computing in which computation is served from a central source, to consumers as an on-demand, elastic utility. For cloud users, there is no risk of under, or over provisioning of infrastructure. Using a cloud-based virtual machine for 1000 hours costs the same as using 1000 machines for 1 hour. Big Data tasks often require heavy provisioning of resources for short periods of time, for example training a machine learning model. Thus, access to cloud computing platforms is essential to a Big Data practitioner.

A cloud computing service can fit into one of three categories. From highest to lowest level, these categories are the following: Software as a Service(SaaS), Platform as a Service(Paas), and Infrastructure as a Service(IaaS). This report is concerned with IaaS, provided by Apache Cloudstack.

II. Apache CloudStack

Apache Cloudstack is an open source Java based IaaS platform, provided by the Apache Software Foundation. CloudStack provides a friendly web interface for virtual machine template creation and instantiation, resource management, security groups, and account separation. In addition, CloudStack provides a native API and an S3/EC2 compatible API for programmatic provisioning of resources.

III. Cloudera

Cloudera is a suite of tools that allows for cluster creation and management. By installing cloudera manager on each virtual machine, it becomes easy to install and configure distributed software packages such as Hadoop and Spark. The cluster creation process is a very difficult one where many things can go wrong; one must be very mindful of networking, configuration, stack selection, security, and scalability.

IV. Security

Security is a vital aspect of cloud deployment. Cloud services are by definition online, thus any vulnerabilities in the system will be exposed to the world. In order for a cluster to function properly, nodes must be able to communicate with each other, thus there will be a need for selectively opening ports and whitelisting IPs. This process must be done with maximum caution. Only the necessary ports should be opened, and only internal IP's should be granted access. Access to virtual machines should always be done over SSL with the use of two key authentication. In any case that login credentials are used (cloudstack, cloudera, web interfaces) defaults such as 'admin admin' must be changed to something more robust.

Master node - 199.70.17.231

Slave node - 199.70.17.7

Ports - Initially, to ssh remotely into the VM, port 22 was opened for the ip of the 'big data lab' (142.58.0.0/16) was opened. Also, port 7180 was opened to view the cloudera manager UI. Then, while installing the services all the ports in both the nodes were opened because different services access different ports to get the installation libraries (Refer point 5 in pitfalls). After which, experiments on spark and hadoop were done. For this, the specific ports ‘8088’ and ‘4040’ were opened in the master to view the service UI. All the other services in the created cluster will run properly, but in order to view the respective service UI, the respective ports must be opened for the master node. Additionally, all the ports for both the master and slave were opened for each other to enable communication.

V. Scalability

In theory, cloud computing allows for practically infinite compute resources available for scaling any project to its required allocation. In practice, however, one must always be mindful of bottlenecks, there are many. Things to consider are disk space per VM, memory per VM, number of data nodes, number of jobs, and application specific configuration, such as number of workers for a spark job.

Process

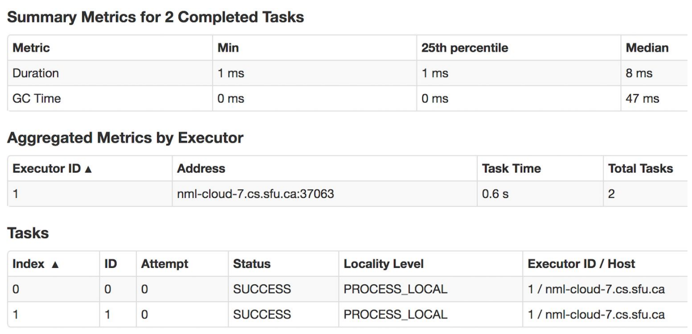
The process of deploying the whole cluster model is explained in the points below

1. An account in apache cloud stack (SFU’s cloud infrastructure) was created.
2. A private key file, (in .pem format) was created to authenticate ssh remote logins to the created VMs
3. Two separate virtual instances, running Ubuntu 16.04 , were created, with specifications of 4x2.4 GHz CPU, with 16GB memory and 1G network, on a 20GB disk drive.
4. One of these instances is chosen as the master node, and “cloudera” manager is installed in this machine by remotely logging into the VM.
5. Then the cloudera UI is accessed, and both the instances are searched through IPs or FQDNs to be included in one cluster.
6. Then, all the necessary packages and services are installed to the cluster setup.
7. For each of the services installed, the master node and data node are specified.
8. Spark and MapReduce are ran as experiments to test the functionality of the cluster.

Experiments

1. **Spark** - A basic spark program was written to to make a spark RDD, and print the full RDD by collecting it from different worker nodes (1 executor in this case). The program was run setting the number of executors and the number of cores to 1. It was also observed that the master node, assigned the work to the one data node, which was viewed in the spark cluster UI.
2. **MapReduce** - Then, another basic mapreduce program was executed to approximate the value of ‘pi’. It was also observed that, the name node assigned the work to the data node.

Both the cluster’s UI can be seen below, where the work is assigned to the data node.



Summary Metrics for 2 Completed Tasks

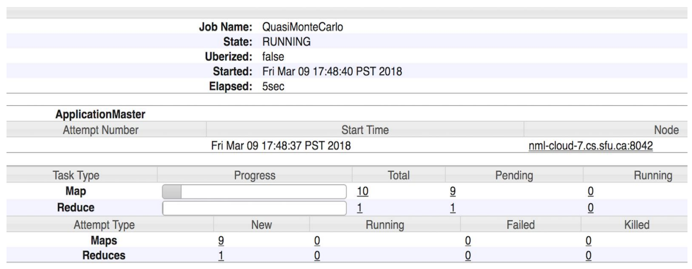
Metric	Min	25th percentile	Median
Duration	1 ms	1 ms	8 ms
GC Time	0 ms	0 ms	47 ms

Aggregated Metrics by Executor

Executor ID ▴	Address	Task Time	Total Tasks
1	nml-cloud-7.cs.sfu.ca:37063	0.6 s	2

Tasks

Index ▲	ID	Attempt	Status	Locality Level	Executor ID / Host
0	0	0	SUCCESS	PROCESS_LOCAL	1 / nml-cloud-7.cs.sfu.ca
1	1	0	SUCCESS	PROCESS_LOCAL	1 / nml-cloud-7.cs.sfu.ca



Job Name:	QuasiMonteCarlo
State:	RUNNING
Uberized:	false
Started:	Fri Mar 09 17:48:40 PST 2018
Elapsed:	5sec

ApplicationMaster

Attempt Number	Start Time	Node
	Fri Mar 09 17:48:37 PST 2018	nml-cloud-7.cs.sfu.ca:8042

Task Type	Progress	Total	Pending	Running
Map		10	9	0
Reduce		1	1	0

Attempt Type	New	Running	Failed	Killed
Maps	9	0	0	0
Reduces	1	0	0	0

Pitfalls and Gotchas

1. Hostname does not resolve on second vm instance - 'sudo' command resulted in 'hostname cannot be resolved'. This was resolved by editing the /etc/hosts file in the host, and setting the proper hostname with the appropriate IP address.

2. Linux IPTables and UFW - The cloudera manager UI was disabled - citing the reason that it was unsafe to access the address. Once the IPv6 tables and the firewall was disabled on the VM, it was accessible.

3. Hosts in bad health during cluster installation - During the installation of the services, both the hosts went into 'bad health' and became unresponsive. This was solved by deleting the /var/lib/cloudera-scm-agent/cm_guid file and restarting the cloudera agent on both the hosts.

4. Improper installation of Spark (not installed on slave) - When, the 'bad-health' issue was resolved, unless, the guid file in both the hosts are deleted and restarted, spark does not get configured in both the VMs.

5. Oozie Sharelib URL not found - During the installation of Impala Statestore, one process tried to access some library folders in a specific location that was initially inaccessible. Once the port number of the location was opened up, the configuration was successful.

6. Cloudera configuration - log files exceeding - It was found that all activities in the cloudera monitor was logged and the size of these log files were increasing at a very rapid pace. Once, the space limit has been exceeded, then the service stop abruptly since none of the activities can be monitored anymore. This issue need further study to come up with a reasonable solution.

7. Running Spark

```
--num-executors=1
--executor-cores=1
--executor-memory=512m
```

By default, the number of executors assigned to any spark program is 3. Since the spark cluster only has 1 manager and 1 more executor, the spark program was unable to run because it was waiting for additional executors while there weren't any. Hence, these parameters were modified as shown above to run those spark programs in the deployed spark cluster.

8. Unhealthy due to swappiness - This issue was reported during the installation of services. It was found that the optimal swappiness value is from 1 to 10. Hence,

"sudo sysctl -w vm.swappiness=10" was used.

Conclusion

The ability to provision virtual machines, and deploy an operational cluster is an extremely powerful tool in the data science toolbox. To master the realm of cloud computing is to essentially grant yourself limitless computational and storage capacities. As Uncle Ben said: "with great power comes great responsibility." Cluster configuration is rich with configuration options, one must proceed slowly, wisely, and with great caution to ensure functionality, scalability, and security requirements are all met.