Yarn @ Phonepe

Task1:

- ### Setup yarn in pseudo-distributed mode

  Install the hadoop single node cluster as done in this https://docs.google.com/document/d/1vABhXIpB5irtR1e0MIyetXGR_vrRZp0m7tDb5MiaO2A/edit?usp=sharing

  Add the following configuration to setup the yarn in it along with the configurations given in the above documentation.

```
hduser@hadoop-VirtualBox:~$ sudo nano /usr/local/hadoop/etc/hadoop/mapred-site.xml
hduser@hadoop-VirtualBox:~$ sudo nano /usr/local/hadoop/etc/hadoop/yarn-site.xml
```

  /usr/loca/hadoop/etc/hadoop/mapred-site.xml

```
<property>
  <name>mapreduce.framework.name</name>
  <value>yarn</value>
 </property>
```

  /usr/local/hadoop/etc/hadoop/yarn-site.xml

```
<configuration>

<!-- Site specific YARN configuration properties -->
<property>
<name>yarn.nodemanager.aux-services</name>
<value>mapreduce_shuffle</value>
</property>
<property>
<name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
<value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>

</configuration>
```

  The configuration is set up. We just have to start the yarn and dfs now.

```
hduser@hadoop-VirtualBox:~$ start-yarn.sh
starting yarn daemons
starting resourcemanager, logging to /usr/local/hadoop/logs/yarn-hduser-resourcemanager-hadoop-Virt
ualBox.out
Enter passphrase for key '/home/hduser/.ssh/id_rsa':
localhost: starting nodemanager, logging to /usr/local/hadoop/logs/yarn-hduser-nodemanager-hadoop-V
irtualBox.out
```

Start dfs

```
hduser@hadoop-VirtualBox:~$ start-dfs.sh
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr
/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.10.1.jar) to method sun.security.krb5.Config.getInstance
()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util
.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
22/03/12 14:39:22 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
g builtin-java classes where applicable
Starting namenodes on [localhost]
Enter passphrase for key '/home/hduser/.ssh/id_rsa':
localhost: starting namenode, logging to /usr/local/hadoop/logs/hadoop-hduser-namenode-hadoop-VirtualBox.out
Enter passphrase for key '/home/hduser/.ssh/id_rsa':
localhost: starting datanode, logging to /usr/local/hadoop/logs/hadoop-hduser-datanode-hadoop-VirtualBox.out
localhost: WARNING: An illegal reflective access operation has occurred
localhost: WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil
```

VBox_GAs_6.1.32

To check if all the elements are set up or not do the following.

```
hduser@hadoop-VirtualBox:~$ jps
2272 NameNode
3186 HQuorumPeer
7699 Jps
3302 HMaster
2471 DataNode
3482 HRegionServer
7180 ResourceManager
2684 SecondaryNameNode
7534 NodeManager
3951 Main
```

The setup is done and we now have yarn installed on the machine.

Task2:

- ## Run sample mapreduce jobs on a input file of your choice & share the results in a word doc

```
hadoop@hadoop-VirtualBox:~$ sudo su hduser
[sudo] password for hadoop:
hduser@hadoop-VirtualBox:/home/hadoop$ cd
```

Enter the following directory and export the Yarn mapreduce Examples directory path.

```
hduser@hadoop-VirtualBox:~$ cd /usr/local/hadoop/bin
```

```
hduser@hadoop-VirtualBox:/usr/local/hadoop/bin$ export YARN_EXAMPLES=/usr/local/hadoop/share/hadoop/mapreduc
e
```

We'll be the using the wordcount class sample.

Create a file and input some data into it and upload the file on hdfs.

```
hduser@hadoop-VirtualBox:/usr/local/hadoop/bin$ cd
hduser@hadoop-VirtualBox:~$ sudo nano input
[sudo] password for hduser:
```

```
hduser@hadoop-VirtualBox:~$ sudo mv input input.txt
```

This is the data we have written in the input.txt

```
  GNU nano 4.8                                    input.txt
HELLO WORLD
hello world
HELLO world
hello WORLD
```

```
hduser@hadoop-VirtualBox:~$ hadoop fs -put input.txt /
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr
/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.10.1.jar) to method sun.security.krb5.Config.getInstance
()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util
.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
22/03/12 14:40:06 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
g builtin-java classes where applicable
```

Run the sample examples jar file. Here wordcount is the class name. Input.txt is the input text file. And wordoutput is the output text file.

```
hduser@hadoop-VirtualBox:/usr/local/hadoop/bin$ ./yarn jar $YARN_EXAMPLES/hadoop-mapreduce-examples-2.10.1.j
ar wordcount /input.txt wordoutput
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr
/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.10.1.jar) to method sun.security.krb5.Config.getInstance
()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util
.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
22/03/12 14:56:24 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
g builtin-java classes where applicable
22/03/12 14:56:25 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
22/03/12 14:56:26 INFO input.FileInputFormat: Total input files to process : 1
22/03/12 14:56:26 INFO mapreduce.JobSubmitter: number of splits:1
22/03/12 14:56:26 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1647076281889_0007
22/03/12 14:56:27 INFO conf.Configuration: resource-types.xml not found
22/03/12 14:56:27 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
22/03/12 14:56:27 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = C
OUNTABLE
22/03/12 14:56:27 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTA
BLE
22/03/12 14:56:27 INFO impl.YarnClientImpl: Submitted application application_1647076281889_0007
22/03/12 14:56:27 INFO mapreduce.Job: The url to track the job: http://hadoop-VirtualBox:8088/proxy/applicat
ion_1647076281889_0007/
22/03/12 14:56:27 INFO mapreduce.Job: Running job: job_1647076281889_0007
22/03/12 14:56:34 INFO mapreduce.Job: Job job_1647076281889_0007 running in uber mode : false
22/03/12 14:56:34 INFO mapreduce.Job:  map 0% reduce 0%
22/03/12 14:56:40 INFO mapreduce.Job:  map 100% reduce 0%
22/03/12 14:56:45 INFO mapreduce.Job:  map 100% reduce 100%
22/03/12 14:56:47 INFO mapreduce.Job: Job job_1647076281889_0007 completed successfully
22/03/12 14:56:47 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=54
                FILE: Number of bytes written=417567
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=145
                HDFS: Number of bytes written=32
                HDFS: Number of read operations=6
                HDFS: Number of large read operations=0
```

The mapreduce operation is done now.

Here’s the output.

```
hduser@hadoop-VirtualBox:/usr/local/hadoop/bin$ hadoop fs -cat wordoutput/*
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.hadoop.security.authentication.util.KerberosUtil (file:/usr
/local/hadoop/share/hadoop/common/lib/hadoop-auth-2.10.1.jar) to method sun.security.krb5.Config.getInstance
()
WARNING: Please consider reporting this to the maintainers of org.apache.hadoop.security.authentication.util
.KerberosUtil
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
22/03/12 16:18:35 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... usin
g builtin-java classes where applicable
HELLO   2
WORLD   2
hello   2
world   2
hduser@hadoop-VirtualBox:/usr/local/hadoop/bin$
```

- Read up on various failure scenarios for an application and share your observations in a word doc.

For the mapreduce program running on YARN, there are a number of failure scenarios that need to be considered :

1. Task Failure
2. Application Master
3. Node Manager
4. Resource Manager

Task Failure :

This occurs due to runtime exceptions or sudden exit of JVM. The signal is sent by task to the application master every 3 seconds. So when the application master didn't receive any signal or update for a long time , it considered the task as failed and rerun the task attempt.

The task will get rescheduled by the application master. It will try to reschedule only 4 times. If it will not be retried again. The job will return failed status.

A task may be killed because it is duplicate or because node manager stopped running or application master marked task attempt as killed.

Resource Manager:

Resource manager is the single point of failure in YARN. to achieve high availability , it is necessary to run a pair of resource managers . If the resource manager fails , then standby can take over without any interruption .

Information about all the running applications is stored in a highly available state store (backed by ZooKeeper or HDFS), so that the standby can recover the core state of the failed active resource manager.

The transition of a resource manager from standby to active is handled by a failover controller. The default failover controller is an automatic one, which uses ZooKeeper leader election to ensure that there is only a single active resource manager at one time.

Node Manager:

Node manager send heartbeat signal for every 3 seconds to the resource manager. If the node manager fails due to crash or running very slowly the resource manager wait for 10 minutes. If not recieved , it will remove the node from its pool to reschedule the container.

If the application master is running on the failed node manager , then application manager will be launchd on the other node and it will not consider as an task attempt.

Node managers may be blacklisted if the number of failures for the application is high, even if the node manager itself has not failed. Blacklisting is done by the application master if more than three tasks fail on a node manager.

Application Master:

If an Application Master fails in Hadoop, then all the information related to that particular job execution will be lost. The maximum number of attempts to run an application master is 2, so if an application master fails twice it will not be tried again and the job will fail. Though, This can be controlled using the property yarn.resourcemanager.am.max-attempts.

An application master sends periodic heartbeats to the Resource Manager, and during application master failure, the Resource Manager will detect the failure and starts a new instance of the master running in a new container, it will use the job history to recover the state of the tasks that were already run by the (failed) application so they don't have to be rerun. Recovery is enabled by default, but can be disabled by setting yarn.app.mapreduce.am.job.recovery.enable to false.

The MapReduce client polls the application master for progress reports, but if its application master fails, the client needs to locate the new instance. During job initialization, the client asks the resource manager for the application master's address, and then caches it so it doesn't overload the resource manager with a request every time it needs to poll the application master. If the application master fails, however, the client will experience a timeout when it issues a status update, at which point the client will go back to the resource manager to ask for the new application master's address.