Topics:

- Vertica cluster management
- Hadoop Hands on sessions
- SQOOP Hands on sessions

Vertica Cluster Management

- Adding nodes to an existing cluster
- Removing nodes from a cluster
- Replacing nodes
- Rebalancing data across nodes

Adding nodes to an existing cluster:

- There are many reasons for adding one or more nodes to an installation of Vertica:

1. **Increase system performance:**

   Add additional nodes due to a high query load or increase disk space without adding storage locations to existing nodes.

2. **Make the database K-safe > 1**

3. **Swap a node for maintenance.** Use a spare machine to temporarily take over the activities of an existing node that needs maintenance. The node that requires maintenance is known ahead of time so that when it is temporarily removed from service, the cluster is not vulnerable to additional node failures.

4. Replace a node. Permanently add a node to remove obsolete or malfunctioning hardware.

Adding nodes consists of the following general tasks:

1. Back up the database

   HPE strongly recommends that you back up the database before you perform this significant operation because it entails creating new projections, refreshing them, and then deleting the old projections.

2. Configure the hosts you want to add to the cluster.

   You will also need to edit the hosts configuration file on all of the existing nodes in the cluster to ensure they can resolve the new host.

3. Add one or more hosts to the cluster

   After you have backed up the database and configured the hosts you want to add to the cluster

4. Add the hosts you added to the cluster (in step 3) to the database.

- Once you have added one or more hosts to the cluster, you can add them as nodes to the database.

Note: When you add a "host" to the database, it becomes a "node." You can add nodes to your database using either the Administration Tools or the Management Console

Removing nodes from a cluster:

- Although less common than adding a node, permanently removing a node is useful if the host system is obsolete or over-provisioned.
- You cannot remove nodes if your cluster would not have the minimum number of nodes required to maintain your database's current K-safety.
- If you really wish to remove the node or nodes from the database, you first must reduce the K-safety level of your database.

- Removing one or more nodes consists of the following general steps:

1. Back up the database

   HPE recommends that you back up the database before performing this significant operation because it entails creating new projections, deleting old projections, and reloading data.

2. Lower the K-safety of your database

   if the cluster will not be large enough to support its current level of K-safety after you remove nodes.

3. Remove the nodes from the database.

4. Remove the hosts from the cluster

   if they are not used by any other databases.

Replacing nodes:

- If you have a K-Safe database, you can replace nodes, as necessary, without bringing the system down.
- For example, you might want to replace an existing node if you:
  - Need to repair an existing host system that no longer functions and restore it to the cluster
  - Want to exchange an existing host system for another more powerful system

Note: Vertica does not support replacing a node on a K-safe=0 database.

- The process you use to replace a node depends on whether you are replacing the node with:
  - A host that uses the same name and IP address
  - A host that uses a different name and IP address

Rebalancing data across nodes:

- Vertica can rebalance your database when you add or remove nodes. Manually trigger a rebalance by using Administration Tools, SQL functions, or using Management Console.
- For segmented projections, Vertica creates new (renamed), segmented projections that are identical in structure to the existing projections, but which have their data distributed across all nodes.
- The rebalance process then refreshes all new projections, and drops all of the old segmented projections.
- All new buddy projections have the same base name so they can be identified as a group.
- For unsegmented projections, leaves existing projections unmodified, creates new projections on the new nodes, and refreshes them.
- After the data has been rebalanced, Vertica drops:
  - Duplicate buddy projections with the same offset
  - Duplicate replicated projections on the same node
- Before data rebalancing completes, Vertica operates with the existing K-safe value. After rebalancing completes, Vertica operates with the K-safe value specified during the rebalance operation.

List of Hadoop & Sqoop programs

1. Checking hadoop configuration files.
2. Loading a file from local file system to hadoop file system.
3. Perform analysis on loaded files using hadoop mapreduce programs and verify the output using hadoop commands as well as browser.

   (a) Count (b) Grep

4. Verifying Sqoop status through cloudera manager

5. Hand-on Practice on various Sqoop basic commands

   (a) List-database (b) List-table (c) Eval

6. Import of tables from Mysql database to hdfs

   (a) Import of all tables

   (b) Import of specific tables to default directory /target directory

   (c) Import of subset of tables using ‘where’ clause

   (d) Incremental import

7. Export files from hdfs to mysql database

Hadoop Hands on sessions

1. Checking hadoop configuration files.

(a) To verify that all softwares are in good health or not

- go to browser
- click on “Cloud Manager”
- enter Username: cloudera

  Password: cloudera
- click on “login”

(b) To check java path

- open the terminal
- cd /usr/lib/jvm
- ls

(c) To check hadoop location

- cd /usr/lib/hadoop-0.20-mapreduce/
- ls

(d) To verify hadoop installation files

- cd conf
- ls
- gedit core-site.xml (similarly we can open other files)

2. Loading a file from local file system to hadoop file system.

- open the terminal
  - to verify whether all daemons are running or not

```
sudo jps
```

- To create a file in local file system

```
gedit test
```

enter some sample data in it and then save & close

- To verify whether or not the file is created

```
ls
```

- To put the local file into hadoop file system

```
hadoop fs –put test  /user/cloudera
```

- To verify whether or not the local file is loaded into hadoop file system

```
hadoop fs –ls /user/cloudera
```

- To check the content of loaded file

```
hadoop fs –cat /user/cloudera/test
```

3. Perform analysis on loaded files using hadoop mapreduce programs and verify the output using hadoop commands as well as browser.

(a) Count

(b) Grep

- To see the list of jar files available in hadoop

```
cd /usr/lib/hadoop-0.20-mapreduce/
ls
```

- To see the content of jar file

```
hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples-2.0.0-mr1-cdh4.4.0.jar
```

- To run word count program on loaded file and creating output file path.

```
hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples-2.0.0-mr1-cdh4.4.0.jar  wordcount /user/cloudera/nh001/test  /user/cloudera/nh001/output1
```

- To verify output files

```
hadoop fs –ls /user/cloudera/nh001
hadoop fs –ls /user/cloudera/nh001/output1
```

- To see the content of output file

```
hadoop fs –cat /user/cloudera/nh001/output1/part-r-00000
```

- To see the output through browser

  click on “HDFS Namenode”→Browse the file system→user→cloudera →nh001 →output1 → part-r-00000

(b) Grep

- To run Grep program on loaded file and creating output file path.

```
hadoop jar /usr/lib/hadoop-0.20-mapreduce/hadoop-examples-2.0.0-mr1-cdh4.4.0.jar  grep  /user/cloudera/test /user/cloudera/output2 key_word
```

- Remaining steps as before

To install mysql

Open terminal and type

Sudo yum –y install mysql-server

Sudo yum –y install mysql-connector-java

Sqoop help (display all commands)

SQOOP Hands on sessions

1. Verifying Sqoop status through cloudera manager

- open the terminal
- To start mysql services

```
sudo service mysqld start
```

- To connect to mysql

```
mysql –u root
```

- To create Database

```
create database sampledb;
```

- To show the existing data bases

```
show databases;
```

- To drop database

```
drop database databasename;
```

- Exit

- To verify the database names

  ```
  show databases;
  ```

- To choose the database you want to use

  ```
  use sampledb;
  ```

- To create tables

  ```
  create table std(rno int);
  create table emp(id int, name char);
  ```

- To insert records into the tables

  ```
  insert into stud values (101), (102), (103), (104), (105);
  insert into emp values (1,’a’), (2,’b’), (3,’c’), (4,’d’), (5,’e’);
  ```

- Exit
- Open the browser and select “Cloudera Manager”
- Check whether or not SQOOP is in good helath

2. Hand-on Practice on various Sqoop basic commands

(a) List-database (b) List-table (c) Eval

- (a) To list the databases

```
sqoop list-databases  --connect  “jdbc:mysql://localhost”
     --username  root  --password  root
```

- (b) To list the tables

```
sqoop list-tables  --connect "jdbc:mysql://localhost/sampledb"
      --username  root  --password  root
```

- (c) To run sql queries from hadoop using eval

```
sqoop eval --connect "jdbc:mysql://localhost/sampledb"
    --username root --password root
      --query "select * from emp“
```

```
sqoop eval --connect "jdbc:mysql://localhost/sampledb"
    --username root --password root
      --query "select count(*) from stud“
```

```
sqoop eval --connect "jdbc:mysql://localhost/sampledb"
    --username root --password root
      --query "select * from emp where id>2"
```

```
sqoop eval --connect "jdbc:mysql://localhost/sampledb"
    --username root --password root
      --query “insert into emp values(5,’y’)”
```

3. Import of tables from Mysql database to hdfs

(a) Import of all tables

(b) Import of specific tables to default directory /target directory

(c) Import of subset of tables using ‘where’ clause

(d) Incremental import

(a) Import of all tables:

```
sqoop  import-all-tables  --connect
    "jdbc:mysql://localhost/sampledb“
        --username  root  --password  root  -m  1
```

To check whether or not tables are imported

```
hadoop  fs  -ls  /user/cloudera
```

To check for a particular table

```
hadoop  fs  -ls  /user/cloudera/stud
```

To see the records in a table

```
hadoop  fs  -cat  /user/cloudera/stud/part-m-00000
```

Note: if name node is in safe mode import wont work

To remove the directory

```
hadoop  fs  -rm  -R  /user/cloudera/stud
hadoop  fs  -rm  -R  /user/cloudera/emp
```

(b) Import of specific tables to default directory /target directory

To import specific table to the default directory

```
sqoop import  --connect  "jdbc:mysql://localhost/sampledb“
   --username root  --password root  --table emp  -m 1
```

To check the imported table in the default directory

```
hadoop  fs  -ls  /user/cloudera/
hadoop fs  -ls  /user/cloudera/emp
hadoop fs -cat /user/cloudera/emp/part-m-00000
```

To make a new directory

```
hadoop fs -mkdir /user/cloudera/hp2
```

To import specific table to the target directory

```
sqoop  import  --connect "jdbc:mysql://localhost/sampledb"
--username root --password root --table emp
--target-dir  /user/cloudera/hp2/sqooplab1 –m 1
```

To check the imported table in the target directory

```
hadoop  fs  -ls  /user/cloudera/hp2
hadoop fs  -ls  /user/cloudera/hp2/sqooplab1
hadoop fs -cat /user/cloudera/hp2/sqooplab1/part*
```

Importing single or multiple tables to specific directory

```
sqoop  import-all-tables  --connect
    "jdbc:mysql://localhost/sampledb"
    --username root --password root
        --warehouse-dir  /user/cloudera/hp2  –m 1
```

```
sqoop  import  --connect "jdbc:mysql://localhost/sampledb"
    --username root --password root
    --table emp
    --warehouse-dir  /user/cloudera/hp2  –m 1
```

(c) Import of subset of tables using ‘where’ clause

To import subset of data

```
    sqoop  import --connect  "jdbc:mysql://localhost/sampledb"
      --username root   --password root
       --table emp  --where “id>‘2'"
      --target-dir  /user/cloudera/hp2/sqooplab2 –m 1
```

To check the imported table in the target directory

```
        hadoop  fs  -ls  /user/cloudera/hp2
        hadoop fs  -ls  /user/cloudera/hp2/sqooplab2
        hadoop fs -cat /user/cloudera/hp2/sqooplab2/part*
```

Import of subset of tables using ‘where’ clause and columns

```
sqoop  import --connect
   “jdbc:mysql://localhost/sampledb"
      --username root   --password root
       --table emp  --columns “col1,col2”
      --where “id>‘2'"
       --target-dir
   /user/cloudera/hp2/sqooplab3 –m 1
```

(d) Incremental import

- To insert new records into the table

```
 sqoop eval --connect "jdbc:mysql://localhost/sampledb"
                    --username root --password root
                      --query “insert into stud values (106),(107)“
```

- To import new records into hadoop

```
   sqoop import --connect "jdbc:mysql://localhost/sampledb"
    --username root --password root  --table stud
     --target-dir  /user/cloudera/hp2/sqooplab1
        --incremental append   --check-column rno
               --last-value 105 –m 1
```

4. Export files from hdfs to mysql database

- To create a file in local file system

```
gedit test
1,a
2,b
3,c
```

save & exit

- To put the local file into hadoop file system

```
hadoop fs –put test  /user/cloudera/
```

- To verify whether or not the local file is loaded into hadoop file system

```
hadoop fs –ls /user/cloudera/
```

- To check the content of loaded file

```
hadoop fs –cat /user/cloudera/test
```

- To create a table structure in mysql

```
sqoop eval --connect "jdbc:mysql://localhost/sampledb"
                  --username root
                 --query “create table test_table(a int,b char)“
```

- To export the file from hadoop to mysql

```
 sqoop export --connect  "jdbc:mysql://localhost/sampledb"
  --username root    --table test_table
  --export-dir  /user/cloudera/test
```

- To check the table data in mysql

```
sqoop eval --connect "jdbc:mysql://localhost/sampledb"
                  --username root
                  --query “select * from  test_table“
```

Importing single or multiple tables to specific directory

```
sqoop  import  --connect
   "jdbc:mysql://localhost/sampledb"
    --username root --password root
   --table emp
    --warehouse-dir  /user/cloudera/hp2 –m 1
```

```
sqoop  import-all-tables  --connect
   "jdbc:mysql://localhost/sampledb"
    --username root --password root
    --warehouse-dir  /user/cloudera/hp2 –m 1
```

Note:

- -m 1 option is necessary when there is no primary key to the table
- -m n you can increase the mappers if you have primary key to the table
- --target-dir will not work on importing all the tables.(you are given choice to mention the directory name where as in warehouse-dir directories are created with same name as mysql table.

To turn off safemode

```
sudo –u hdfs hdfs dfsadmin –safemode leave
```