MySQL Cluster setup and MaxScale configuration

Massimiliano Pinto

Last Updated: 1st August 2014

Overview

The document covers the MySQL Cluster 7.2.17 setup and MaxScale configuration in order to load balancing the SQL nodes acces.

MySQL Cluster setup

The MySQL Cluster 7.2.17 setup is based on two virtual servers with Linux Centos 6.5

- server1:
    NDB Manager process
    SQL data node1
    MySQL 5.5.38 as SQL node1

- server2:
    SQL data node2
    MySQL 5.5.38 as SQL node2

Cluster configuration file is /var/lib/mysql-cluster/config.ini, copied on all servers

```
[ndbd default]
NoOfReplicas=2
DataMemory=60M
IndexMemory=16M

[ndb_mgmd]
hostname=178.62.38.199
id=21
datadir=/var/lib/mysql-cluster

[mysqld]
hostname=178.62.38.199

[mysqld]
hostname=162.243.90.81

[ndbd]
hostname=178.62.38.199

[ndbd]
hostname=162.243.90.81
```

Note, it's possible to specify all node ids and datadir as well for each cluster component

Example:

```
[ndbd]
hostname=162.243.90.81
id=43
datadir=/usr/local/mysql/data
```

and /etc/my.cnf, copied as well in all servers

```
[mysqld]
ndbcluster
ndb-connectstring=178.62.38.199
innodb_buffer_pool_size=16M

[mysql_cluster]
ndb-connectstring=178.62.38.199
```

Startup of MySQL Cluster

Each cluster node process must be started separately, and on the host where it resides. The management node should be started first, followed by the data nodes, and then finally by any SQL nodes:

- On the management host, server1, issue the following command from the system shell to start the management node process:

```
[root@server1 ~]# ndb_mgmd -f /var/lib/mysql-cluster/config.ini
```

- On each of the data node hosts, run this command to start the ndbd process:

```
[root@server1 ~]# ndbd —-initial -—initial-start

[root@server2 ~]# ndbd —-initial -—initial-start
```

- On each SQL node start the MySQL server process:

```
[root@server1 ~]# /etc/init.d/mysql start

[root@server2 ~]# /etc/init.d/mysql start
```

Check the Cluster status

If all has gone well, and the cluster has been set up correctly, the cluster should now be operational.
It's possible to test this by invoking the **ndb_mgm** management node client.
The output should look like that shown here, although you might see some slight differences in the output depending upon the exact version of MySQL that you are using:

```
[root@server1 ~]# ndb_mgm
-- NDB Cluster -- Management Client --
ndb_mgm> show
Connected to Management Server at: 178.62.38.199:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     2 node(s)
id=24 @178.62.38.199  (mysql-5.5.38 ndb-7.2.17, Nodegroup: 0, *)
id=25 @162.243.90.81  (mysql-5.5.38 ndb-7.2.17, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=21 @178.62.38.199  (mysql-5.5.38 ndb-7.2.17)

[mysqld(API)]   2 node(s)
id=22 @178.62.38.199  (mysql-5.5.38 ndb-7.2.17)
id=23 @162.243.90.81  (mysql-5.5.38 ndb-7.2.17)

ndb_mgm>
```

Working with NDBCLUSTER engine in MySQL

```
[root@server1 ~]# mysql

mysql> CREATE TABLE `t1` (   `a` int(11) DEFAULT NULL )
ENGINE=NDBCLUSTER;
Query OK, 0 rows affected (3.28 sec)

mysql> show create table t1;
+-------+-----------------------------------------------------------------
------------------------------+
| Table | Create Table
|
+-------+-----------------------------------------------------------------
------------------------------+
| t1    | CREATE TABLE `t1` (
  `a` int(11) DEFAULT NULL
) ENGINE=ndbcluster DEFAULT CHARSET=latin1 |
+-------+-----------------------------------------------------------------
------------------------------+
1 row in set (0.01 sec)



mysql> insert into test.t1 values(11);
Query OK, 1 row affected (0.15 sec)



mysql> select count(1) from t1;
+----------+
| count(1) |
+----------+
|        1 |
+----------+
1 row in set (0.07 sec)
```

From the MySQL client pointing to SQL node on server2

```
[root@server2 ~]# mysql
mysql> select count(1) from test.t1;
+----------+
| count(1) |
+----------+
|        1 |
+----------+
1 row in set (0.08 sec)
```

Configuring MaxScale for connection load balancing of SQL nodes

Add these sections in MaxScale.cnf config file:

```
[Cluster Service]
type=service
router=readconnroute
router_options=ndb
servers=server1,server2
user=test
passwd=test
version_string=5.5.37-CLUSTER

[Cluster Listener]
type=listener
service=Cluster Service
protocol=MySQLClient
port=4906

[NDB Cluster Monitor]
type=monitor
module=ndbclustermon
servers=server1,server2
user=monitor
passwd=monitor
monitor_interval=8000
```

```
[server1]
# SQL node1
type=server
address=127.0.0.1
port=3306
protocol=MySQLBackend


[server2]
#SQL node2
type=server
address=162.243.90.81
port=3306
protocol=MySQLBackend
```

Assuming MaxScale is installed in server1, start it

```
[root@server1 ~]# cd /usr/local/skysql/maxscale/bin
[root@server1 bin]#  ./maxscale -c ../
```

Using the debug interface it's possible to check the status of monitored servers

```
MaxScale> show monitors
Monitor: 0x387b880
     Name:      NDB Cluster Monitor
     Monitor running
     Sampling interval:    8000 milliseconds
     Monitored servers:    127.0.0.1:3306, 162.243.90.81:3306


MaxScale> show servers
Server 0x3873b40 (server1)
     Server:                   127.0.0.1
     Status:                   NDB, Running
     Protocol:                 MySQLBackend
     Port:                     3306
     Server Version:           5.5.38-ndb-7.2.17-cluster-gpl
     Node Id:                  22
     Master Id:                -1
     Repl Depth:               0
     Number of connections:    0
```

```
     Current no. of conns:      0
     Current no. of operations: 0
Server 0x3873a40 (server2)
     Server:                    162.243.90.81
     Status:                    NDB, Running
     Protocol:                  MySQLBackend
     Port:                      3306
     Server Version:            5.5.38-ndb-7.2.17-cluster-gpl
     Node Id:                   23
     Master Id:                 -1
     Repl Depth:                0
     Number of connections:     0
     Current no. of conns:      0
     Current no. of operations: 0
```

It's now possible to run basic tests with the read connection load balancing for the two configured SQL nodes

(1) test MaxScale load balancing requesting the Ndb_cluster_node_id variable:

```
[root@server1 ~]# mysql -h 127.0.0.1 -P 4906 -u test -ptest -e "SHOW
STATUS LIKE 'Ndb_cluster_node_id'"
+---------------------+-------+
| Variable_name       | Value |
+---------------------+-------+
| Ndb_cluster_node_id | 23    |
+---------------------+-------+
```

```
[root@server1 ~]# mysql -h 127.0.0.1 -P 4906 -u test -ptest -e "SHOW
STATUS LIKE 'Ndb_cluster_node_id'"
+---------------------+-------+
| Variable_name       | Value |
+---------------------+-------+
| Ndb_cluster_node_id | 22    |
+---------------------+-------+
```

The MaxScale connection load balancing is working.

(2) test a select statement on an NBDBCLUSTER table, database test and tabe t1:

```
[root@server1 ~] mysql -h 127.0.0.1 -P 4906 -utest -ptest -e “SELECT
COUNT(1) FROM test.t1”
+----------+
| COUNT(1) |
+----------+
|        1 |
+----------+
```

(3) test an insert

```
mysql -h 127.0.0.1 -P 4906 -utest -ptest -e “INSERT INTO test.t1
VALUES (19)”
```

(4) test again the select and check the number of rows

```
[root@server1 ~] mysql -h 127.0.0.1 -P 4906 -utest -ptest -e “SELECT
COUNT(1) FROM test.t1”

+----------+
| COUNT(1) |
+----------+
|        2 |
+----------+
```