POLYTECHNIQUE MONTRÉAL

LOG8415E

ADVANCED CONCEPTS OF CLOUD COMPUTING

Lab 1

Scaling Databases and Implementing Cloud Patterns

Lab Instructor:
Vahid Majdinasab

Authors:
2024785 - Alexandre Dufort

Instructor:
Amin Nikanjam

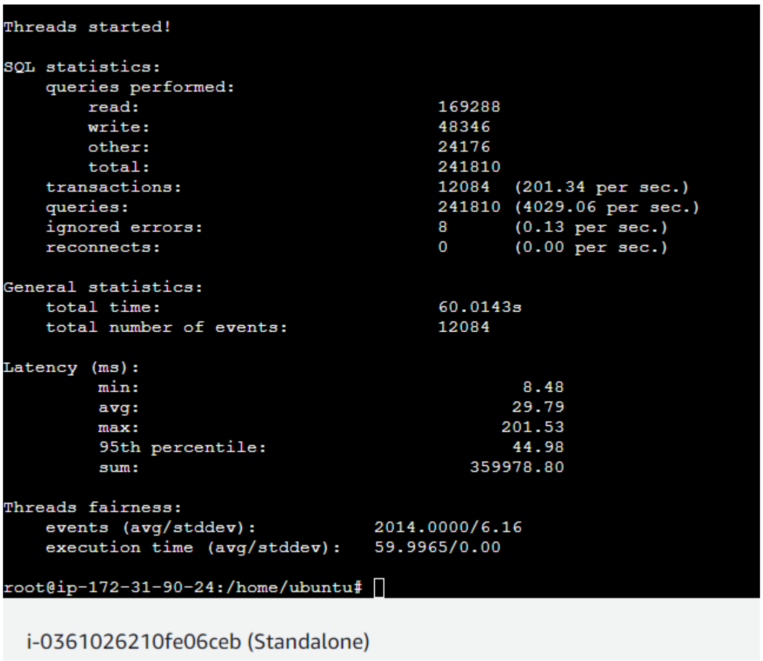
https://github.com/AlexDufort/projet8415

decembre 23, 2022

Contents

1 Benchmarking MySQL stand-alone vs. MySQL Cluster

After running MySQL on the stand-Alone server we obtain some data showing the performance on that server.

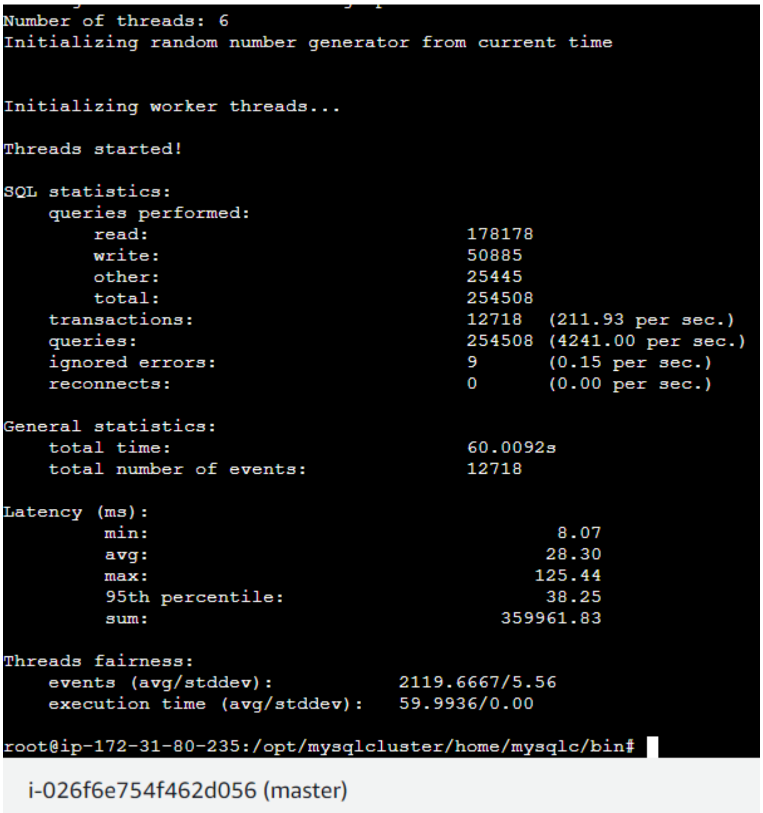


```
Threads started!

SQL statistics:
    queries performed:
        read:                           169288
        write:                          48346
        other:                          24176
        total:                          241810
    transactions:                       12084  (201.34 per sec.)
    queries:                            241810 (4029.06 per sec.)
    ignored errors:                     8      (0.13 per sec.)
    reconnects:                         0      (0.00 per sec.)

General statistics:
    total time:                         60.0143s
    total number of events:             12084

Latency (ms):
         min:                                    8.48
         avg:                                   29.79
         max:                                  201.53
         95th percentile:                       44.98
         sum:                               359978.80

Threads fairness:
    events (avg/stddev):           2014.0000/6.16
    execution time (avg/stddev):   59.9965/0.00

root@ip-172-31-90-24:/home/ubuntu#
```

i-0361026210fe06ceb (Standalone)

Figure 1.1 - Execution of command

We then proceed to connect three slave to a Master and then run the same benchmark on that master. The result obtained were slightly different and not big enough to distinguished quickly.

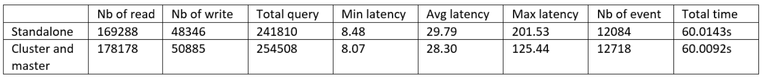


```
Number of threads: 6
Initializing random number generator from current time


Initializing worker threads...

Threads started!

SQL statistics:
    queries performed:
        read:                           178178
        write:                          50885
        other:                          25445
        total:                          254508
    transactions:                       12718  (211.93 per sec.)
    queries:                            254508 (4241.00 per sec.)
    ignored errors:                     9      (0.15 per sec.)
    reconnects:                         0      (0.00 per sec.)

General statistics:
    total time:                         60.0092s
    total number of events:             12718

Latency (ms):
         min:                                    8.07
         avg:                                   28.30
         max:                                  125.44
         95th percentile:                       38.25
         sum:                               359961.83

Threads fairness:
    events (avg/stddev):           2119.6667/5.56
    execution time (avg/stddev):   59.9936/0.00

root@ip-172-31-80-235:/opt/mysqlcluster/home/mysqlc/bin#
```

i-026f6e754f462d056 (master)

Figure 1.2 - Benchmark of the Stand-Alone Server

To understand more the slight difference between those benchmark i did a quick table to compare the major difference.



	Nb of read	Nb of write	Total query	Min latency	Avg latency	Max latency	Nb of event	Total time
Standalone	169288	48346	241810	8.48	29.79	201.53	12084	60.0143s
Cluster and master	178178	50885	254508	8.07	28.30	125.44	12718	60.0092s

Figure 1.2 - Benchmark of the Cluster Server

As you can see the standalone server did run a smaller number while being much slower than the Cluster with a Master and three slave. For numerical purposes, the Stand-Alone server did run 12 698 query less than the Cluster. The minimum latency,the average latency and maximum latency for the standalone was respectively 0.41 ms slower, 1.49 ms slower and 85.09 ms slower. This cause be cause by the help of the slave when doing all those SQL query.// Finally For a slightly bigger number of event my cluster run a tenth of a second faster.

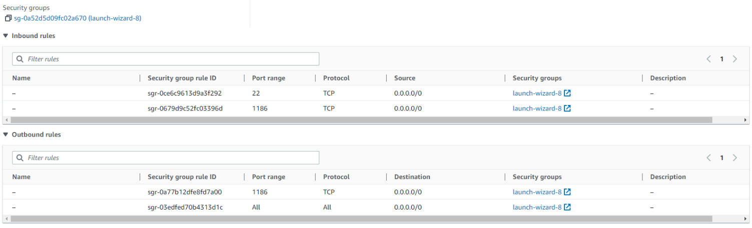
2 Implementation of The Proxy pattern

As you can see the proxy pattern was in development but i had no time left and face some difficulty with the sshTunnel. I know what i had to do to complete this stage but couldn't do it. The first step was to create an sshTunnel trough my proxy server so it could filter the request. After i had to implement 3 different type of request the first one being directly forward to my master. The second type was one were the request was being send to a random slave. The last one is a custom one here the node with the less response time was the one where the query was forwarded.

The next step was to connect my proxy to my github and have it run my app.py file where all the proxy could have been.

3 Describe clearly how your implementation works.

Let's start with the stand-Alone server. The first step was starting a t2.micro instance in which i had to installed MySQL and as a root create a user, a database and grant my user to have permission over the database. I finally use sysbench to send some benchmarking. For the Cluster, I opened 4 t2.micro instance 3 of them were slave and 1 one them is a master. I gave each and one of them a specific inbound rule and outboud rule for a better connection when time is due.



Security groups
sg-0a52d5d09fc02a670 (launch-wizard-8)

Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups	Description
–	sgr-0ce6c9613d9a3f292	22	TCP	0.0.0.0/0	launch-wizard-8	–
–	sgr-0679d9c52fc03396d	1186	TCP	0.0.0.0/0	launch-wizard-8	–

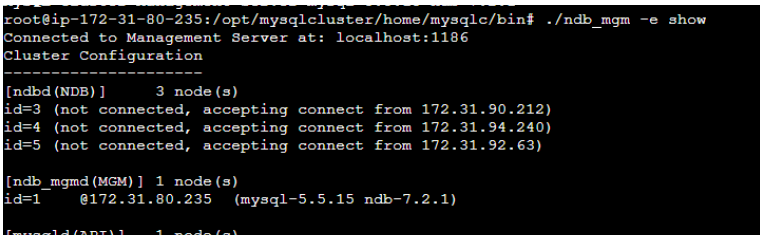
Outbound rules

Name	Security group rule ID	Port range	Protocol	Destination	Security groups	Description
–	sgr-0a77b12dfe8fd7a00	1186	TCP	0.0.0.0/0	launch-wizard-8	–
–	sgr-03edfed70b4313d1c	All	All	0.0.0.0/0	launch-wizard-8	–

Figure 3.1 - Inbound and Outbound rule for a slave

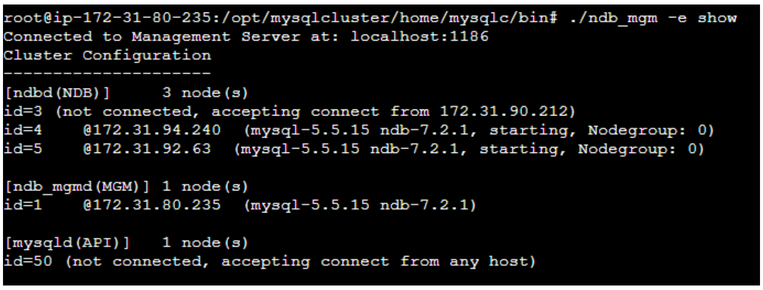
After starting each one of my instance, I had to follow some tutorial to connect each node to the master using their private IP. After assuring that MySQL was on my master I connected each of my node to the master and verify with the command **./ndb_mgm -e show**.

The figure 3.3 we can see two of the three node connected to the master.



```
root@ip-172-31-80-235:/opt/mysqlcluster/home/mysqlc/bin# ./ndb_mgm -e show
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     3 node(s)
id=3 (not connected, accepting connect from 172.31.90.212)
id=4 (not connected, accepting connect from 172.31.94.240)
id=5 (not connected, accepting connect from 172.31.92.63)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @172.31.80.235  (mysql-5.5.15 ndb-7.2.1)
```

Figure 3.2 - showing the command show before connecting the node



```
root@ip-172-31-80-235:/opt/mysqlcluster/home/mysqlc/bin# ./ndb_mgm -e show
Connected to Management Server at: localhost:1186
Cluster Configuration
---------------------
[ndbd(NDB)]     3 node(s)
id=3 (not connected, accepting connect from 172.31.90.212)
id=4    @172.31.94.240  (mysql-5.5.15 ndb-7.2.1, starting, Nodegroup: 0)
id=5    @172.31.92.63  (mysql-5.5.15 ndb-7.2.1, starting, Nodegroup: 0)

[ndb_mgmd(MGM)] 1 node(s)
id=1    @172.31.80.235  (mysql-5.5.15 ndb-7.2.1)

[mysqld(API)]   1 node(s)
id=50 (not connected, accepting connect from any host)
```

Figure 3.3 - showing the command show before connecting the node

Finally in the same way as the Stand-Alone server we run some query on a recently created database and show the result of these query.//

4 Summary of results and instructions to run your code.

As their is no automatic deployment and the proxy haven't been done their is no code to be run as a matter of fact my git was primarily use for the storage of my ssh key of all my instance and some benchmark file. I would have love to present more to you but unfortunately their is nothing more. You can see the beginning of try inside the flask app with some requirement.txt for my app but nothing more.