
# Final project - MySQL benchmark & cloud patterns

Presented to Vahid Majdinasab

By : Ismail Bakkouri - 1954157

LOG8415E - Advanced Concepts of Cloud Computing

October 31st, 2022

# 1. Introduction

For this final project, I had to set up and benchmark mysql standalone server, mysql cluster with a master and 3 nodes, and a proxy pattern that prevents querying the cluster node directly. This report explains how the deployment procedure is done, the results of the benchmark, how the proxy pattern works and provides some instructions to run the code.

# 2. Deployment Procedure

The code recycles the EC2_instances_creator.py developed for Assignment 1 and 2 to create AWS EC2 instances, tweaked a bit to receive additional parameters such as private IP address. We started the new instances with some new bash scripts: master_launch_script.sh, which does the configuration for the master node; slave_launch_script.sh, which does the configuration for the 3 slave nodes; stand-alone_launch_script.sh, which does the configuration and benchmarking of the standalone finally, the proxy_launch_script.sh that setups the proxy between the local client and the cluster.

During the deployment of the proxy, a repository (https://github.com/ismail19988/files) is cloned, and it contains the private key to do a remote connection to the cluster instances, and a custom flask servers capable of receiving the requests from the client and reroutes them to the cluster.

# 3. Benchmark results

| statistics | Cluster | Standalone |
|---|---|---|
| Queries performed | 171 720 | 281 076 |
| Transaction per sec. | 286.0 | 467.9 |
| Queries per sec. | 5720 | 9365 |
| Average Latency (ms) | 20,97 | 12.82 |

The benchmark above has been done with sysbench, over the Sakila database, populated with 50,000 records. and for 30 seconds, with 6 threads.
We can observe that the standalone is much faster than the cluster (about twice as fast). I expected the cluster to be much faster. Since he has a master and 3 slaves, I expected it to be 4 times faster. The average latency also looks much lower in the standalone. I expected the standalone to have a lower latency because he doesn't have to establish any connection with other nodes to get the results unlike the cluster. I think that the cluster has lower speeds and worse latency, because he has to establish communications with the nodes which reduces processing and querying time. The best option between the cluster and the standalone would

depend on the usage that we would like to do. if we want a system not scalable for simple projects, we would need to take the standalone. If we want a scalable system, we would need to use a cluster, that is much easier to scale up and down.

## 4. Proxy pattern

The proxy pattern consists of an instance, with a flask server running on it. The instance has a few routes that allow it to receive the hit type:

- **proxy_instance_public_ip/direct_hit**: this route is used to receive a query to perform directly over the master node.
- **proxy_instance_public_ip/random_hit:** this route is used to receive a query to perform over a random node.
- **proxy_instance_public_ip/custom_hit:** this route is used to receive a query to perform over the node with the lowest latency.

The proxy pattern is able to establish the connection with the ip addresses that have been hardcoded to each node of the cluster.

Basically, the proxy.py file is the client, and it sends http requests to the proxy instance, then the instance reroutes them to the cluster nodes through SSH.

The proxy instances clones the git https://github.com/ismail1999888/files/ which contains the server, and the key to do the remote connection to the cluster from the proxy instance.

## 5. Instructions to run the code

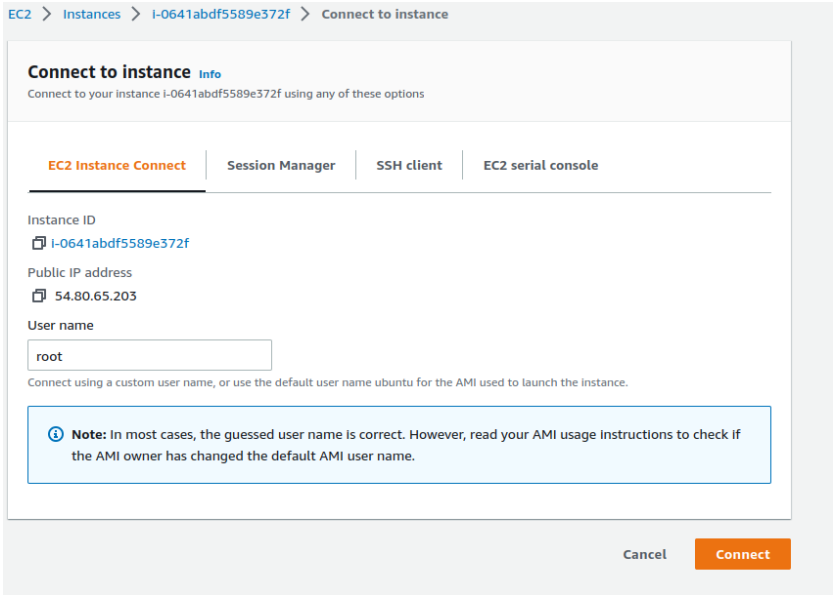
Before executing the main Python scripts, the following prerequisites must be met (these are the same steps and tips that have been given in the previous assignments):

- The code must be located on the local machine
- Python 3 must be installed locally on the machine
- The Boto3 library (Python SDK for AWS) must be installed locally on the machine

The next step is to set up the authentication credentials for the AWS account. If it does not already exist, a file named "credentials" should be created under "~/.aws/". Inside this file, both the aws access key id and secret access key should be specified. This will allow the boto3 client to have access to AWS. Another important file to have is the "config" file, also located under "~/.aws/". This file should specify a default region to use such as "region=us-east-1".

When the you have all the prerequisites you have to edit the constant.py, where you have to edit the value of "SUBNET_ID" with your own subnet (IP range 172.31.80.0) if you don't have a subnet with that range, just create it, because otherwise you will need to change the

configuration scripts that take the hardcoded node IP addresses. after you enter your own subnet, you can run the main.py. The main.py will set up the standalone and run its benchmark, and will set up the cluster. to run the cluster benchmark you will have to connect to the master node like this:




And then copy paste the script cluster_benchmark.sh in the instance. I explained in the video why I was not able to automate it like the standalone.

When both the benchmarkings are complete, you will find a file named stand-alone_banchmark in the standalone instance, and cluster_benchmark in the master instance of the cluster (both files are gonna be found in the /~ folder).

for the proxy instance, the flask server is gonna be automatically deployed. You will need to get the public address of the proxy instance and change it in the proxy.py, line 6 with the correct instance public ip, so the script can send requests.

To query the proxy, you will also need to select the SSH key available in the github ismail19988/files. Download it from the git and set it the key to be used in your instances by uploading it to your AWS account and setting it up in the config file, line 15.

When you run the client, you will need to pass 2 arguments, which are the query, followed by type (direct, random, custom) allshould be  in lowercase.

All the python code and bash scripts are available here:

https://github.com/ismail19988/final-project-8415 (source code)

https://github.com/ismail19988/files (code cloned by the proxy instance during the setup)