LOG8415E - Final Project Report

William Bourret - 2005931

December 23rd 2022

1 Benchmarking MySQL stand-alone vs MySQL Cluster

For this project, I benchmarked a MySQL stand-alone setup and a MySQL Cluster using AWS and compared the to installations. The database used for the benchmarking is Sakila and the tool used is Sysbench. For each type of installation, we have a table size of 100 000, used 6 threads and ran it for a maximum time of 60 seconds. For the stand-alone server, we used one t2.micro Ubuntu instance, while we used four for the cluster, one as master and the three others as slaves.

2 Implementation of the Proxy pattern

To implement the Proxy pattern, we used a t2.large Ubuntu instance that was created in the same subnet as the cluster. On this instance, I implemented an application that creates an SSH tunnel between the proxy instance and the master node and sends a "SELECT" query to a data node depending on the chosen implementation. If we chose a direct implementation, the application sends a query to the master node. If the random one is chosen, the proxy queries a random slave node. Finally, if the custom implementation is selected, the proxy will send a ping to all nodes and choose the one with the least response time.

3 Description of the implementation

The solution I implemented is very simple. First off, there is a Python script that is ran in a Docker container that creates the instances and security groups needed for the implementation. The reason why it is done in a Docker container is because I had some difficulty configuring the AWS CLI with my credentials. Also, Docker allows this script to be more portable.

Next, we have five shell scripts for the setup the implementation. We have `standalone.sh` to setup the stand-alone server and benchmark it, `master.sh` to setup the master node on the cluster, `slave.sh` to setup the slave nodes, `cluster_benchmark.sh` to benchmark the cluster and finally, `proxy.sh` to install the Proxy application.

Finally, on the Proxy, we clone the project repository on to the Proxy instance and run the application that creates an SSH tunnel between the Proxy and the master node and queries one of the nodes on the cluster depending on the implementation that is chosen.

4 Summary of the results

For the stand-alone server, we simply used a t2.micro Ubuntu instance and installed MySQL and the Sakila database on it. In regards to the results, we have a total of 17 059 transactions (284.27 transactions/sec) and 341 180 queries (5685.48 queries/sec). In terms of latency, the minimum was at 7.85 ms, the maximum was at 81.16 ms and the average was at 21.10 ms.

For the cluster, we used 4 t2.micro instances. One was used as the master node, while the three others were slave nodes. When we look at the results, we see that there were 24 259 transactions (404.7 transactions/sec) and 485 906 queries (8097.51 queries/sec). For latency, we have a minimum of 5.17 ms, a maximum of 216.12 ms and an average of 14.84 ms.

As we can see, according the results, the MySQL cluster seems to be much better the the stand-alone version. This is probably due to the fact that in a cluster, the data is replicated across different nodes, causing less bottlenecks when it comes to multiple queries being processed at the same time. For the Proxy application, we ran the following query on the Sakila database: "SELECT * FROM actor". This is what the query returned:

```
actor_id first_name last_name last_update
0 1 PENELOPE GUINESS 2006-02-15 04:34:33
1 2 NICK WAHLBERG 2006-02-15 04:34:33
2 3 ED CHASE 2006-02-15 04:34:33
3 4 JENNIFER DAVIS 2006-02-15 04:34:33
4 5 JOHNNY LOLLOBRIGIDA 2006-02-15 04:34:33
.. ... ... ... ...
195 196 BELA WALKEN 2006-02-15 04:34:33
196 197 REESE WEST 2006-02-15 04:34:33
197 198 MARY KEITEL 2006-02-15 04:34:33
198 199 JULIA FAWCETT 2006-02-15 04:34:33
199 200 THORA TEMPLE 2006-02-15 04:34:33
```

5 Instructions to run the code

5.1 Prerequisites

In order to run this solution you must do the following:

1. Install Docker Engine (https://docs.docker.com/get-docker/).
2. Fill in the .env file with your AWS CLI credentials.
3. In the `infrastructure\_builder.py` file, replace the `SUBNET_ID` with your subnet id. This can be found in the AWS console on the "VPC" page, under the "Subnets" option.
4. Copy your `.pem` access key in the repository.

5.2 Benchmarking

In order to generate the instances needed for this solution, simply run `./run.sh`. This will generate the Docker container that will be used to setup the instances.

5.2.1 Benchmarking the MySQL stand-alone server

To benchamrk the MySQL Standalone server, connect to the server named "Standalone" via SSH, and copy the commands from the `setup_scripts/standalone.sh` or simply copy the file on the instance, using `scp` and your access key and run the script.

Make sure to run the commands as the `root` user. In order to do so, run `sudo -s`.

The results will be under `/ubuntu/home/results.txt`.

5.2.2 Benchmarking the MySQL Cluster

First off, as root user on the master instance (named "master"), you must copy and run the commands on from the `setup_scripts/master.sh` file in order too set up the master node.

Next, on each of the slaves (named "slave_1", "slave_2" and "slave_3"), as root user, you must copy and run the commands found in `setup_scripts/slave.sh`. This will setup the slave nodes.

Finally, back on the master node, you must copy and run the commands found `setup_scripts/benchmark_cluster.sh`, that will run the benchmarking. The results will be under `/ubuntu/home/results.txt` on the master node.

5.3 Proxy

Once the benchmarking is done, as root user once again on the Proxy server (named ”proxy”):

1. Copy the commands from the `setup_scripts/proxy.sh`.
2. `cd LOG8415-FinalProject`
3. Run `python3 proxy_app.py -p <implementation>` where `<implementation>` is either `direct`, `random` or `custom`.

Happy Holidays!