LOG8415
Advanced Concepts of Cloud Computing

Simon Tran (1961278)

Département Génie Informatique et Génie Logiciel
École Polytechnique de Montréal, Québec, Canada

December 23 2022

1 Project source code

Source code is available on github: `https://github.com/tran-simon/log8415-final-project`

2 Project Setup

To install the necessary components, follow these steps. Consult the README.md file for more information.

1. Create 5 t2.micro EC2 instances (standalone, master, slave1, slave2, and slave3).
2. Create 1 t2.large instance for the proxy server.
3. Add inbount rules to the security group for the instances so that they accept requests from eachother.
4. Add the IP addresses of all the VMs in the .env file using the following command:

   ```
   aws ec2 describe-instances --output table --query \
   'Reservations[].Instances[].[Tags[?Key==Name] | [0].Value, PublicIpAddress, PrivateIpAddress]
   ```

5. Run the installation script:

   ```
   ./install.sh
   ```

3 Benchmark

To run the benchmark, use the following commands:

```
./scripts.sh standalone start sysbench
./scripts.sh master start sysbench
```

4 Proxy

To set up and start the proxy server, use the following commands:

```
./scripts.sh proxy dependencies
./scripts.sh proxy install

./scripts.sh proxy start
```

To test some queries using the proxy server, use the following commands:

```
./scripts.sh proxy query direct-hit "SHOW STATUS;"
./scripts.sh proxy query random "SELECT * FROM actor;"
./scripts.sh proxy query customized "SHOW DATABASES;"
```

5 Useful commands

To connect to the VM via Secure Shell (SSH), use the following command:

```
./scripts.sh master connect
```

To run commands on the VM using SSH, use the following command:

```
./scripts.sh master connect "ps aux"
```

To run SQL commands on the VM, use the following command:

```
./scripts.sh master sql "SHOW DATABASES;"
```

To stop processes on the VMs, use the ‘stop‘ command

```
./scripts.sh master stop ndb
./scripts.sh master stop mysql
```

To check the status of various components on the master VM, use the following commands:

```
./scripts.sh master status mysql
./scripts.sh master status cluster
./scripts.sh master status ndb
./scripts.sh master status sakila
```

To access the NDB Management Console on the master VM, use the following command:

```
./scripts.sh master manage
```

6 Benchmark Results

In the ‘results‘ folder, there are two text files containing the result of the benchmarks.

We can see that the standalone VM was able to process a total of 33400 transactions at a speed of 556.59 transactions per second.
There were a total of 534400 queries (8905.52 queries per second).

We can see that the VM cluster was able to process a total of 39227 transactions at a speed of 653.71 transactions per second.
There were a total of 627632 queries (10459.42 queries per second).

We can conclude that the cluster was able to process about 17% more transactions and queries.

7 Implementation

The ‘script.sh‘ file contains various scripts to interact with the VMs. It contains the function to install the dependencies and configure the VMs.

The master node runs the MySQL server and the MySQL NDB Cluster manager.
The slave nodes run NDB. They are configured so that they are read-only copies of the master database.
Sakila is installed on the master node.

The proxy VM runs a node application that consists of an Express server.
When it first launches, it creates 4 SSH tunnels, one on port 4000 for the master, one on port 4001 for the slave1, one on port 4002 for the slave2 and one on port 4003 for the slave3.
This makes it possible to query the slave nodes, because the SSH tunnel forwards the traffic to them.
We can simply make SQL queries to 'localhost:4001' to make a query on the slave1, for example.
Read queries can be made on any node, but write queries are only sent to the master node, because the slave nodes are read-only.

Then, we have 3 Express routes used to send SQL queries:

- '/direct-hit': Queries the master node. You can use the optional 'destination' query parameter to chose a slave node instead.
- '/random': Queries a random slave node
- '/customized': Pings all nodes to determine the one with the lowest latency using 'nmap'. The query will be sent to that one.

The body of the HTTP request contains the SQL query.

Here is an example request:

```
POST http://54.160.131.212:3000/direct-hit?destination=slave2
Content-Type: text/plain

SHOW STATUS;
```

You may use the 'query' function in 'scripts.sh' to easily run queries:

```
./scripts.sh proxy query direct-hit "SHOW STATUS;"
./scripts.sh proxy query random "SELECT * FROM actor;"
./scripts.sh proxy query customized "SHOW DATABASES;"
```

As we use a t2.large instance for the proxy, our application could scale well with lots of users and large requests. The proxy can effectively route requests to nodes that have less load.
Thanks to the clustering, we can have a unlimited number of slave instances, so that they can be in charge of handling read requests instead of having a single standalone server that handles everything.

8 Conclusion

In this project, we learned how to scale databases and implement cloud patterns using AWS. We created multiple EC2 instances and set up a proxy server, and learned how to run benchmarks and test queries using various scripts and commands. These skills are important for managing and optimizing database performance in the cloud.