LOG8415
Advanced Concepts of Cloud Computing
Scaling Databases and Implementing Cloud Design Patterns

Vahid Majdinasab
Département Génie Informatique et Génie Logiciel
École Polytechnique de Montréal, Québec, Canada
vahid.majdinasab[at]polymtl.ca

Student(s)
Ana Karen López Baltazar (2311301)

GitHub:
https://github.com/karen-baltazar/LOG8415

Demo:
https://youtu.be/HO63CIU5dCI

December 28, 2023

1 Implementation Notes

Before proceeding with the detailed description of the implementation of this project, it should be clarified that the current implementation has not been thoroughly tested. This fact falls squarely on my responsibilities, as I did not adequately prepare to avoid the situation where I would lack the means and credits necessary to perform extensive testing due to a lack of foresight on my part. I acknowledge this situation and accept the possible sanctions, as I will not be able to fully verify the functionality of the implementation.

In this report, my focus is on providing a detailed conceptual explanation of how this implementation operates and the theoretical steps required for its execution. It is important to note that the development of this implementation was based on consulting various tutorials and materials, which will be detailed in the references. I thank you in advance for your understanding in this matter.

2 Benchmarking MySQL stand-alone vs. MySQL Cluster

2.0.1 Explanation

The overall approach to implementation focuses on automating the process. The necessary instances are first created in AWS and then the relevant configuration files are copied according to the role of each instance. In the `scripts.sh` file, benchmarking corresponds to the first three steps. The order of creation of the instances is as follows: stand-alone, master node, slave nodes and proxy.

Below is a brief explanation of each file:

- `scripts.sh`: This main script is responsible for creating the instances in AWS, updating the IP addresses in the scripts, and copying configuration files to the corresponding instances.
- `aws_setup.py`: This script is responsible for creating instances in Amazon Web Services (AWS). This script uses the `boto3` Python library to interact with the AWS API and create EC2 instances. [1]
- `update_config_ips.sh`: This script is responsible for updating the private IP addresses in the corresponding scripts after the creation of instances in AWS.
- `master_setup.sh`: This script runs on the master node and configures the instance to function as the primary node in the cluster. [3]
- `slave_setup.sh`: This script runs on each slave node and configures the instances to function as secondary nodes in the cluster. [3]
- `mysql_stand_alone.sh`: This script is used in stand-alone environment and configures the instance to run MySQL independently. [2]
- `benchmark.sh`: This script performs performance tests and saves the results to a text file (`/home/ubuntu/results.txt`). [4] [6]

2.0.2 Execution Steps

To perform benchmarking (after executing `scripts.sh`), follow these steps:

Stand-alone environment:

1. Connect via SSH to the stand-alone instance.
2. Run `mysql_stand_alone.sh` on the corresponding instance.
3. Run `benchmark.sh` to perform performance tests and save the results to `/home/ubuntu/results.txt`.
4. To view the results, navigate to the `/home/ubuntu/` directory and open the file using a text editor such as `nano` or `cat`.

Cluster Environment:

1. Connect via SSH to the master node instance.
2. Run `master_setup.sh` on the master node.
3. Connect via SSH to each slave node and run `slave_setup.sh`.
4. Reconnect to the master node and run `benchmark.sh` to perform performance tests. The results will be saved in `/home/ubuntu/results.txt`.
5. To view the results, navigate to the `/home/ubuntu/` directory and open the file using a text editor such as `nano` or `cat`.

2.0.3 Results

In terms of results, although there is no practical demonstration of the implementation, instead what we can do is make inferences from them based on the characteristics and nature of the MySQL environments. Specifically, MySQL Cluster is designed to offer high availability and improved performance compared to a single node configuration [3]. When examining transaction statistics, which indicate the number of transactions completed and the speed in transactions per second [6], one would expect the cluster environment to outperform the stand-alone environment.

However, to obtain practical validation, testing in a real environment and verifying performance under varied loading conditions would be necessary.

2.0.4 Important note

During the execution of the scripts, errors may be encountered in the stand-alone environment related to the specific user configuration. Likewise, there may be errors or problems related to the configuration of the private IP addresses of the cluster nodes. It is recommended to verify and ensure the correct configuration of private IP addresses in configuration scripts such as`master_setup.sh` and `slave_setup.sh` before execution. Additionally, you are encouraged to review the IP address settings in the `proxy_setup.py` script later in the process. The accuracy of these settings is crucial to guarantee test results.

3 Cloud Patterns: Proxy

To implement the proxy pattern, the file `proxy_setup.py` was developed, designed to run on the proxy instance. This file uses the MySQL Connector/Python library for connection to the MySQL cluster [7], and a socket for communication with the trusted host [8].

3.1 `proxy_setup.py`

The proxy configuration file, `proxy_setup.py`, operates as a socket server to receive queries from the trusted host and as a client to connect to the MySQL cluster. The proxy logic is based on processing a query based on its nature. So, if it is a request that modifies the database, the *direct hit* strategy is used, sending the query to the master node of the cluster. On the other hand, if it is a read query, the *custom* strategy is used, where the slave node with the least load is selected to process the query.

3.2 Execution Steps

Before running the proxy, make sure you have completed MySQL cluster configuration and system benchmarking. Then follow these steps:

1. Connect via SSH to the proxy instance.

2. Run the following commands to install the dependencies:

```
sudo apt update
sudo apt install python3
pip3 install mysql-connector-python
pip3 install ping3
```

3. Run the file `proxy_setup.py` with the following command:

```
python3 proxy_setup.py
```

3.3 Important note

It is crucial to verify that the IP addresses in the configuration file `proxy_setup.py` match the private IP addresses of the instances in the cluster and also the private IP of the proxy.

Also, note that in the current implementation, using the *random* strategy to randomly direct queries to slave nodes has been omitted. This decision was made to fit the system specifications (project instructions).

4 Cloud Patterns: Gatekeeper

For the implementation of the gatekeeper, two Python scripts have been developed, `gatekeeper_setup.py` and `trusted_host_setup.py`, respectively. Communication between them is carried out through sockets [8], where the gatekeeper acts as a client of the trusted host and the latter as a server. The trusted host also functions as a proxy client, receiving queries from the gatekeeper and sending them to the proxy for processing in the MySQL cluster.

4.1 `gatekeeper_setup.py`

This script, when run on the gatekeeper instance, establishes a socket connection to the trusted host. It acts as an interface to the external audience, processing incoming requests and sending them to the Trusted Host for validation and execution on the MySQL cluster.

4.2 `trusted_host_setup.py`

When run on the trusted host instance, this script functions as a server for the gatekeeper and as a client for the proxy. It receives requests from the gatekeeper, performs basic validation of the structure using regular expressions [9], and then sends the queries to the proxy for processing in the MySQL cluster.

4.3 Execution Steps

Trusted Host:

1. Connect via SSH to the trusted host instance.
2. Update the system and download Python:

   ```
   sudo apt update
   sudo apt install python3
   ```

3. Run the file `trusted_host_setup.py` with the following command:

   ```
   python3 trusted_host_setup.py
   ```

Gatekeeper:

1. Connect via SSH to the trusted host instance.
2. Update the system and download Python:

   ```
   sudo apt update
   sudo apt install python3
   ```

3. Run the file `gatekeeper_setup.py` with an example query against the Sakila database:

   ```
   python3 gatekeeper_setup.py '<query>'
   ```

4.4 Important note

It is essential to keep in mind that for the successful execution of the scripts, a prior configuration of the environment must be carried out. Additionally, the order of execution is critical for the proper functioning of the solution. First, the proxy script must be executed, followed by the trusted host and finally the gatekeeper. This order guarantees the logical flow for processing queries in the MySQL cluster.

On the other hand, it should be noted that the current implementation of cloud patterns still has room for improvement. Since it is possible to refine the sanitization of queries on the trusted host, implement encryption for communication via sockets and strengthen security rules and groups, especially with the trusted host. It is recommended to consider blocking ports 80 and 443 if they are not necessary, in order to improve the overall security of the deployment. The decision to keep them open is more due to personal uncertainty as to whether blocking them could hinder the system update and Python installation (in the case of the trusted host).

5 Cloud Patterns: Results

Although I do not have the means to verify the following results, the following queries are proposed to determine the correct functioning of the implementation:

Update Query for the `actor` table:

```
UPDATE actor SET last_name = 'Smith' WHERE actor_id = 1;
```

Expected results:

```
Query OK, 1 row affected
```

Read Query for the `film` table:

```
SELECT title, description, release_year, rating
FROM film WHERE rating = 'PG-13'
ORDER BY release_year DESC LIMIT 5;
```

Expected results:

```
+----------------------+-------------------------+--------------+--------+
| title                | description             | release_year | rating |
+----------------------+-------------------------+--------------+--------+
| Movie 1              | Thrilling adventure...  | 2022         | PG-13  |
| Exciting Journey     | Heroes on journey...    | 2021         | PG-13  |
| Mystery Mansion      | Explore old mansion...  | 2020         | PG-13  |
| Lost in Time         | Time-traveling adventure| 2019         | PG-13  |
| Beyond the Stars     | Journey to the stars... | 2018         | PG-13  |
+----------------------+-------------------------+--------------+--------+
```

These results are fictitious and are provided for illustrative purposes only of what would be expected.

6 Conclusion

Despite not being able to debug and see how the implementation fully worked, this project was challenging, but at the same time satisfying, since it encouraged me to learn and investigate different media and tools in order to devise a concrete implementation. So, I feel satisfied with what I have learned. If possible, I would like to request for future courses that students be notified, if one is aware of it, of the dates on which the AWS course ends in order to avoid situations like in my case. I request this, since it would be of great help to plan the project and know with certainty the times that are not only for delivery, but also for testing the entire infrastructure. Without anything else to add, I thank you and wish you happy holidays.

7 References

1. Boto3 Documentation
   `https://boto3.amazonaws.com/v1/documentation/api/latest/index.html`

2. How To Install MySQL on Ubuntu 20.04:
   `https://www.digitalocean.com/community/tutorials/how-to-install-mysql-on-ubuntu-20-04`

3. How To Create a Multi-Node MySQL Cluster on Ubuntu 18.04:
   `https://www.digitalocean.com/community/tutorials/how-to-create-a-multi-node-mysql-cluster-on-ubuntu-18-04`

4. Sakila Sample Database:
   `https://dev.mysql.com/doc/sakila/en/sakila-installation.html`

5. Sakila Database Installation [**ALT**]:
   `https://www.sqliz.com/sakila/installation/`

6. Benchmark MySQL server Performance with Sysbench:
   `https://www.jamescoyle.net/how-to/1131-benchmark-mysql-server-performance-with-sysbench`

7. Connecting to MySQL Using Connector/Python:
   `https://dev.mysql.com/doc/connectors/en/connector-python-example-connecting.html`

8. Socket Programming in Python:
   `https://realpython.com/python-sockets/`

9. re [Regular Expression] Documentation:
   `https://docs.python.org/3/library/re.html`