Polytechnique Montreal
LOG8415: Final Project
Scaling Databases and Implementing Cloud Design Patterns

Abstract

In this lab assignment, you are asked to setup MySQL cluster on Amazon EC2 and implement Cloud patterns. You tasked with implementing and deploying an architecture by adding the Proxy and the Gatekeeper patterns. In the first part of the assignment you will install, configure, and benchmark MySQL stand-alone server against the MySQL cluster. Next, you will apply your newly acquired skills to implement and compare Cloud patterns in a distributed cluster. Finally, you will report your results by producing a report in the handover documentation format.

Objectives

The overall goals of this lab assignment are:

- Get hands-on experience running MySQL Cluster on Amazon EC2 and benchmark it against the standalone server.
- Learn how to implement patterns strategies to load data into a set of VM instances on Amazon EC2.
- In the lecture, we've seen the Cloud patterns and the strategies commonly used when distributing data across VM instances, now it's time to implement and test these strategies. First, you will go through the tutorial on the MySQL Cluster database engine to familiarize yourself with a distributed database environment. After this, you will implement one of the common Cloud patterns in an application. Finally, you will do a write-up about your results and submit your final work. As for previous labs, the progress of your work should be tracked using version control.

MySQL stand-alone server, MySQL Cluster, and Sakila

- Create a t2.micro instance and install MySQL stand-alone on it. To install MySQL, please follow the instructions described in [2].
- To install the MySQL Cluster, you must create four t2.micro instance, including one manager and three workers. You can follow the instructions described in [3].
    - Following the steps in [3], you may face some issues during your cluster setup. To solve these issues, follow the steps below:
        1. When you download the MySQL Cluster using this command: wget http://dev.mysql.com/get/Downloads/MySQL-Cluster-7.2/mysql-cluster-gpl-7.2.1-linux2.6-x86_64.tar.gz/from/http://mysql.mirrors.pair.com/
           it downloads it into a file named index.html. You can use this command instead wget http://dev.mysql.com/get/Downloads/MySQL-Cluster-7.2/mysql- cluster-gpl-7.2.1-linux2.6-x86_64.tar.gz

- The EC2 instances are not initialized with the libncurses5 package. This will cause problems during cluster initialization. Therefore:
    1. After running
       source /etc/profile.d/mysqlc.sh step
       run
       sudo apt-get update && sudo apt-get -y install libncurses5
       to install the package.
    2. When you want to run the
       ndb_mgmd
       command, make sure that you are in this directory
       /opt/mysqlcluster/home/mysqlc/bin/
       or alternatively, instead of running just the
       ndb_mgmd
       command, run this
       sudo /opt/mysqlcluster/home/mysqlc/bin/ndb_mgmd -f /opt/mysqlcluster/deploy/conf/config.ini --initial --configdir=/opt/mysqlcluster/deploy/conf/

- Once the setup for MySQL server is complete, you must install Sakila database on your stand-alone server and cluster. To know more about Sakila and how to install it, please refer to [1].

Benchmarking MySQL and MySQL Cluster

After you are done configuring both installations of MySQL and mounting the Sakila database, you will need to benchmark these two installations using Sysbench. Detailed instructions about the benchmarking of MySQL is available here: [4]. Your task is to find out how to report the performance of MySQL setup against the cluster.

Cloud Patterns

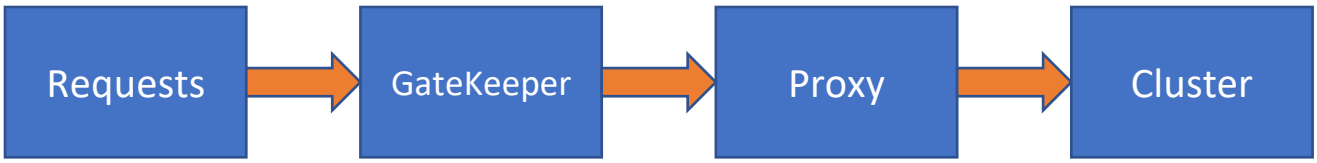
You should implement, and benchmark the two Cloud patterns described below:

Proxy:

This pattern uses data replication between manager/worker databases and a proxy to route requests and provides a read scalability on a relational database. *Write requests* are handled by the manager and replicated on its workers, while *Read requests* are processed by workers. When applying this pattern, components must use a local proxy whenever they need to retrieve or write data.

The Gatekeeper:

This pattern describes a way of brokering access to the storage. This is a typical security best practice and serves to minimize the attack surface of system roles. This is done by communicating over internal channels and only to other roles that are part of the pattern. The Gatekeeper pattern takes two roles that play the gate keeping game. There is one Internet-facing web role that handles requests from users. The Gatekeeper is suspicious and does not trust any requests it receives. It validates the input and runs in partial trust. When some hacker manages to successfully attack the web role, there is no sensitive data there.
Figure below shows the flow of how your solution should work:



Your requests are sent to the gatekeeper. The gatekeeper re-routes the requests to the proxy where depending on their type (read or write) they are re-routed to the corresponding workers.

Implementing Cloud Patterns

Proxy:

Three implementations are required for this pattern:

1. Direct hit: meaning that you will directly forward incoming requests to MySQL master node and there will be no logic to distribute data.
2. Random: you should implement a code that will randomly choose a slave node on MySQL cluster and forward the request to it.
3. Customized: you should measure the ping time of all the servers and forward the message to one with less response time.

Proxy pattern requires one t2.large instance as the proxy server which will route requests to MySQL Cluster.

The Gatekeeper:

You need to create one t2.large instance for the gatekeeper and another t2.large instance for the trusted host. The application must be deployed on the gatekeeper, and it will send requests to the manager of the clustered MySQL database. Please pay attention that the trusted host machine must be secured in anyway that might expose it. You must take care of all security aspects that might put this machine in risk. That means that the firewall, unused services, ports, and even IPtable should be refined, tuned, and well cared.

How This Relates to Your Previous Assignments

1. In your first and second assignments, you had to spin up instances and install libraries on them using code. Here, the process for setting up the instances and installing MySQL on them is the same.
2. In your second assignment, you needed to retrieve information about your instances in order to either setup Hadoop or your containers. Here, the process for setting up the MySQL cluster is the same.
3. In your first assignment, you needed to setup the security groups and rules for them using code. Here, for setting up the GateKeeper, you need to follow the same process, albeit with stricter rules.

Instructions

For this project, the use of AWS's SDK for programming languages is ***highly*** recommended. If you have not used the SDKs for your previous assignments, you can use what you have previously developed.

Report

You need to submit the report alongside your code base. To present your findings and answer questions, you must write a lab report on your results using the handover documentation format which you used for the previous assignments. For the demos, you can record a video of going over your code and upload it to Moodle. In your report, you should explain:

- Benchmarking MySQL stand-alone vs. MySQL Cluster
- Implementation of The Proxy pattern
- Implementation of The Gatekeeper pattern
- Describe clearly how your implementation works.
- Summary of results and instructions to run your code.

Evaluation

A single final submission for this assignment is due on the date specified in Moodle for each person. You need to submit one PDF file per person. All necessary codes, scripts and programs must be sufficiently commented and attached to your submission. For the demo, you need to record a video in which you explain your code, your decision process, and finally your results. Your assignment will be graded on presentation and content. Please submit your PDF report and push your code to a GitHub repository.

Acknowledgement

We would like to thank Amazon Web Services for supporting us through their and AWS Educate grants.

References

[1] - Install sakila database. https://dev.mysql.com/doc/sakila/en/sakila-installation.html
[2] - Installing mysql. https://www.linode.com/docs/databases/mysql/install-mysql-on-ubuntu-14-04/
[3] - Installing mysql cluster. https://stansantiago.wordpress.com/2012/01/04/installing-mysql-cluster-on-ec2/
[4] - Sysbench benchmark. https://www.jamescoyle.net/how-to/1131-benchmark-mysql-server-performance-with-sysbench