**A logo of a university

Description automatically generated Polytechnique de Montre´al**

**LOG8415e**

**ADVANCED CONCEPTS IN CLOUD COMPUTING**

**Mohamed Ben-Faras**

**2090619**

**December 31, 2023**

**Introduction**

This personal project is separated into 2 parts. The first part consists of conducting a comparison between MySQL Stand-Alone and MySQL Cluster. The library used for benchmarking is sysbench, and the database employed is Sakila. We aim to achieve the most comprehensive results in this part by running both the cluster and the standalone version in a similar environment. The second part involves implementing cloud patterns: Proxy and Gatekeeper using the MySQL Cluster that has already been implemented and explaining the implementation details in depth.

* 1. **MYSQL Stand-Alone Benchmark**

For the MySQL Stand-Alone configuration, I needed to create a single t2.micro instance and execute a script named 'stand\_alone\_config.sh' to set up the instance and prepare it for benchmarking. The library used for benchmarking is sysbench, and the parameters are as follows:

- Six threads run simultaneously.

- The table size we are working on is 100,000.

The benchmarking results are as follows:

|  |  |
| --- | --- |
| Queries | Amount |
| Read | 253190 |
| Write | 72340 |
| Other | 36170 |
| Total | 361700 |

**Table 1 : Amount of time a certain querie was performed**

|  |  |
| --- | --- |
| Latency | Time(ms) |
| Min | 6.76 |
| Avg | 19.91 |
| Max | 86.70 |
| Sum | 360049.98 |

**Table 2 : The duration of the latency**

|  |  |
| --- | --- |
| Threads Fairness | Avg/stddev |
| Event | 3014.1667/9.51 |
| Exec. Time | 60.0083/0 |

**Table 3 : Threads fairness**

**1.2 MySQL Cluster Benchmarking**

For the MySQL Cluster configuration, I needed to create four t2.micro instances where one serves as a master, and the other three act as workers. Depending on the role, the script running during the instance initialization is different (cluster\_master\_config.sh / cluster\_slave\_config.sh). Once the four instances have been set up and are ready for benchmarking, the library used for benchmarking is sysbench, and the parameters are as follows:

-Six threads run simultaneously.

-The table size we are working on is 100,000.

The benchmarking results are as follows:

|  |  |
| --- | --- |
| Queries | Amount |
| Read | 336994 |
| Write | 96208 |
| Other | 48112 |
| Total | 481314 |

**Table 1 : Amount of time a certain querie was performed**

|  |  |
| --- | --- |
| Latency | Time(ms) |
| Min | 4.38 |
| Avg | 14.97 |
| Max | 156.28 |
| Sum | 359954.35 |

**Table 2 : The duration of the latency**

|  |  |
| --- | --- |
| Threads Fairness | Avg/stddev |
| Event | 4006.8333/7.31 |
| Exec. Time | 59.9924/0.00 |

**Table 3 : Threads fairness**

**1.3 Analysis of the Benchmarking**

The MySQL Cluster exhibited a higher transaction rate of 400.60 per second compared to the Stand-Alone configuration, which achieved 301.30 transactions per second. The latency for the MySQL Cluster was slightly lower, with an average latency of 14.97 ms compared to the Stand-Alone's 19.91 ms. Threads fairness analysis suggests that the MySQL Cluster had a more consistent distribution of events among threads with an average of 4006.83 events per thread, while the Stand-Alone had an average of 3014.17 events per thread. Overall, the MySQL Cluster demonstrated better transaction throughput and lower latency, making it a favorable choice for scenarios with higher transactional demands. The Stand-Alone configuration, while still performing well, showed slightly lower throughput and a marginally higher latency in comparison. The choice between the two configurations should be based on specific use case requirements and scalability considerations.

**2.1 Proxy Cloud Pattern**