Week#2 TPC-C Benchmark

Siyeol Choi
2019315530

1. INTRODUCTION

In this lab experiment, it conducted the basic tpc-c benchmark to try the method to numerically estimate the performance of the DBMS. The experiment was based on MySQL 5.7.33 and used tpc-c to analyze the benchmark. After the tpc-c, plotted the data to see the tendency of the change of the data.

2. METHODS

The ultimate purpose of this experiment is to assess the performance of the MySQL DBMS at the Windows WSL environment. Since the Windows Subsystem for Linux is based on virtual disk, so the performance can be comparably lower than the actual machine who allocates real disks.

The main workflow was as follows, first enable the MySQL server with the desired benchmark setup with 200Mb of buffer pool size. Then run the tpc-c benchmark script to see the logged result of the benchmark. While conducting the benchmark test, keep track of the memory being used and how much the process are being possessed by the CPU. After the whole benchmark is logged, analyzed the Data with Python to extract the useful data and plot it to compare the tendency of the data.

3. Performance Evaluation

3.1 Experimental Setup

[System Setup]

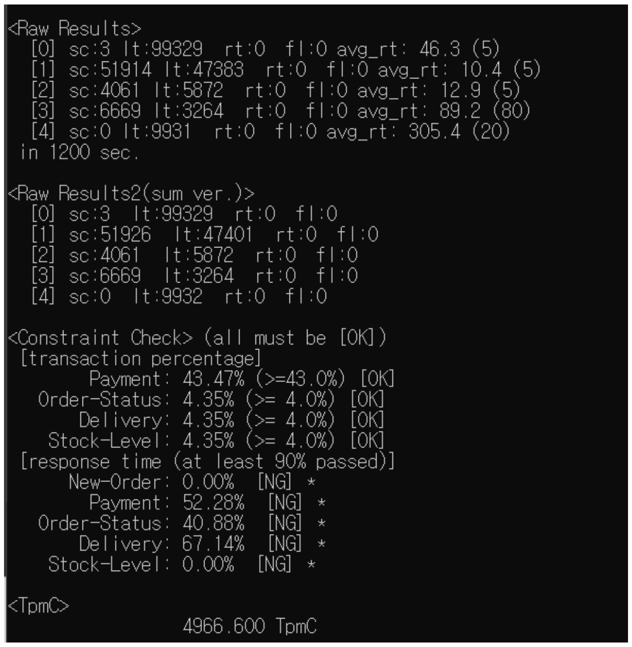
MySQL version	5.7.33
OS	Ubuntu 20.04.1 LTS (GNU/Linux 4.19.104-microsoft-standard
CPU model name	Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz
cpu cores	4
Vendor:	Msft
Product:	Virtual Disk
Revision:	1.0
Compliance:	SPC-3
User Capacity:	274,877,906,944 bytes [274 GB]
Logical block size:	512 bytes
Physical block size:	4096 bytes

[Benchmark setup]

Type	Configuration
DB size	2GB (20 warehouse)
Buffer Pool Size	200MB (10% of DB size)
Benchmark Tool	tpcc-mysql
Runtime	1200s
Connections	8

3.2 Experimental Results

[tpc-c benchmark result summary]



```
<Raw Results>
  [0] sc:3 lt:99329  rt:0  fl:0 avg_rt: 46.3 (5)
  [1] sc:51914 lt:47383  rt:0  fl:0 avg_rt: 10.4 (5)
  [2] sc:4061 lt:5872  rt:0  fl:0 avg_rt: 12.9 (5)
  [3] sc:6669 lt:3264  rt:0  fl:0 avg_rt: 89.2 (80)
  [4] sc:0 lt:9931  rt:0  fl:0 avg_rt: 305.4 (20)
 in 1200 sec.

<Raw Results2(sum ver.)>
  [0] sc:3  lt:99329  rt:0  fl:0
  [1] sc:51926  lt:47401  rt:0  fl:0
  [2] sc:4061  lt:5872  rt:0  fl:0
  [3] sc:6669  lt:3264  rt:0  fl:0
  [4] sc:0  lt:9932  rt:0  fl:0

<Constraint Check> (all must be [OK])
 [transaction percentage]
        Payment: 43.47% (>=43.0%) [OK]
   Order-Status: 4.35% (>= 4.0%) [OK]
       Delivery: 4.35% (>= 4.0%) [OK]
    Stock-Level: 4.35% (>= 4.0%) [OK]
 [response time (at least 90% passed)]
      New-Order: 0.00%  [NG] *
        Payment: 52.28%  [NG] *
   Order-Status: 40.88%  [NG] *
       Delivery: 67.14%  [NG] *
    Stock-Level: 0.00%  [NG] *

<TpmC>
                 4966.600 TpmC
```

Payment is taking up the largest portion among the transaction percentage. It exceeds the sum of all other components, so it is the part that needs the biggest room of improvement. Other three parts of transaction are equally distributed of its times being consumed.

On the other hand, the response time, the delivery had the best result among all while order-status had the worst among all. The new-order and stock-level remaining at 0% needs to be further discussed on future iterations of this tpc-c experiment.

[Aggregated Memory Usage]

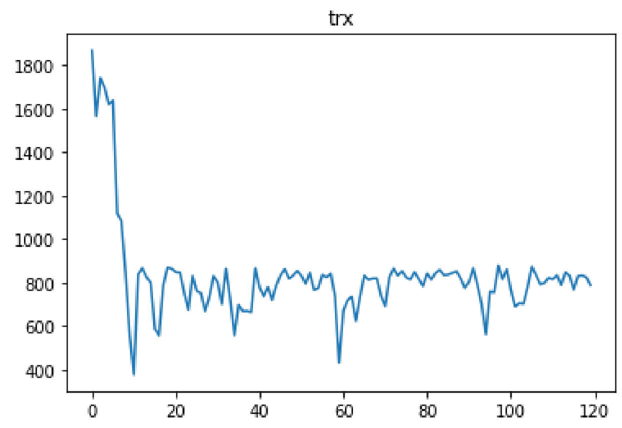
MemTotal: 6425984 kB

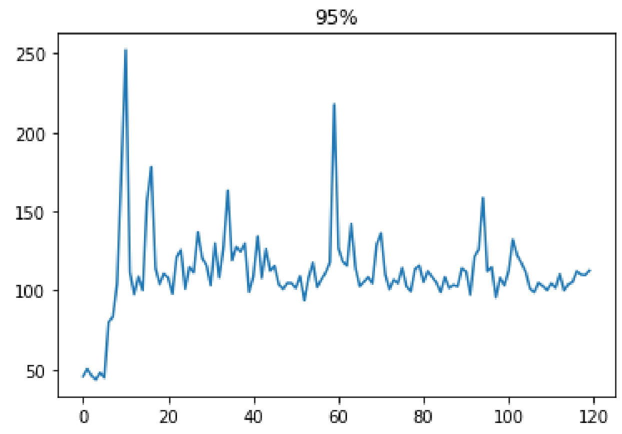
MemFree (AVG): 4537964 kB

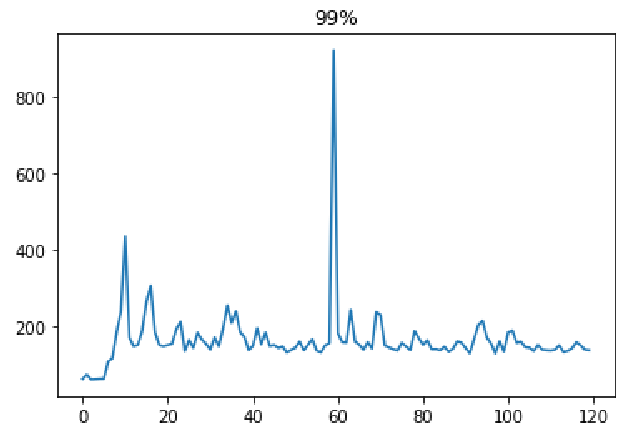
MemAvailable (AVG): 5622760 kB

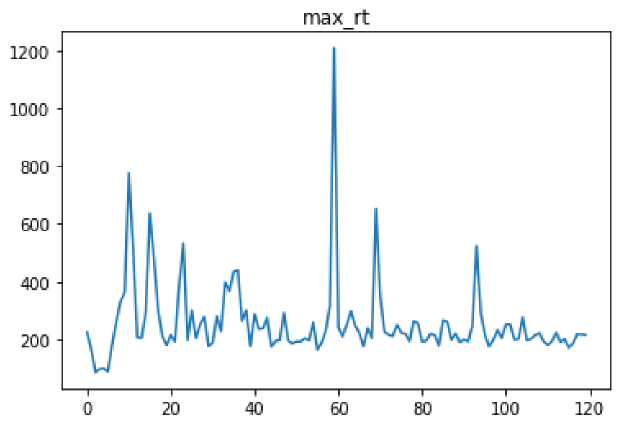
* Since the memory usage fluctuates, set time interval and logged the memory usage and aggregated as an average of the whole data

[Plot of trx, P95, P88, Max_rt]









P95, P99, max_rt all seems to align with each other, but the beginning of P95 had a high peak when it started. Which means that the distribution of data is unequally distributed at the timeslot [10,20] seconds.

4. Conclusion

While running the benchmark, overall usage of memory and CPU were high, which means it has huge overhead when running the process. Therefore, the improvement needs to be conducted especially on payment section and order-status section where it had the worst performance.

5. REFERENCES

[1] Plotting https://www.percona.com/blog/2013/07/01/tpcc-mysql-simple-usage-steps-and-how-to-build-graphs-with-gnuplot/

[2] Report format https://github.com/LeeBohyun/SWE3033-F2022/blob/main/report-guide.md

[3] TPC-C explanation https://ninako21.tistory.com/63