

Week#2 Measure System Status and Metrics While Running TPC-C

Jaehun Lee
2017314626

1. INTRODUCTION

TPC-C is OLTP benchmark to measure the performance of databases. 5 types of transactions are used for benchmarking : New-Order, Payment, Delivery, Order-Status, Stock-Level. In this exercise, TPC-C will be used to benchmark MySQL performance and during TPC-C, system IO/CPU stat will be measured.

2. METHODS

The performance of MySQL can be checked by throughput of TPC-C, which is 'TpmC' (Transactions per minute Count). Also, TPC-C has Random I/O intensive workload, approximately 65% Reads and 35% writes. The purpose of this exercise is to learn how to use TPC-C and get TpmC and system status, metrics during TPC-C benchmark. First, check TpmC for measuring MySQL performance. Then check if read/write IO ratio approximates to 65:35. Finally, check CPU status.

Buffer Pool Size was set as 200MB, which is 10% of total DB size 2GB (20 warehouse). Benchmark was done with tpcc-mysql. Benchmark was runned for 1200s with 8 connections. Benchmark and system status results were recorded in seperate text files.

3. Performance Evaluation

3.1 Experimental Setup

[Table 1] System Setup

Type	Specification
OS	Ubuntu 20.04.3 LTS
CPU	Intel(R) Core(TM) i5-10400 CPU @ 2.90GHz (6 cores)
Memory	8GB
Kernel	5.11.0-43-generic
Device	samsung ssd 860 evo 500GB

[Table 2] 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

[Figure 1] TPC-C Output

```
<Raw Results>
[0] sc:1871 lt:199834 rt:0 fl:0 avg_rt: 23.1 (5)
[1] sc:126751 lt:74947 rt:0 fl:0 avg_rt: 6.4 (5)
[2] sc:12512 lt:7659 rt:0 fl:0 avg_rt: 6.5 (5)
[3] sc:16318 lt:3854 rt:0 fl:0 avg_rt: 53.3 (80)
[4] sc:120 lt:20049 rt:0 fl:0 avg_rt: 125.4 (20)
in 1200 sec.

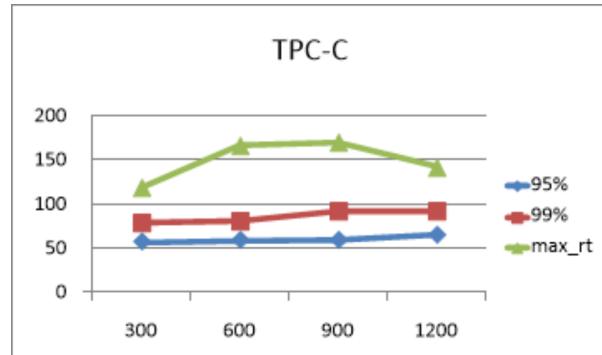
<Raw Results2(sum ver.)>
[0] sc:1871 lt:199834 rt:0 fl:0
[1] sc:126754 lt:74950 rt:0 fl:0
[2] sc:12512 lt:7659 rt:0 fl:0
[3] sc:16318 lt:3854 rt:0 fl:0
[4] sc:120 lt:20049 rt:0 fl:0

<Constraint Check> (all must be [OK])
[transaction percentage]
    Payment: 43.48% (>=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: 0.93% [NG] *
    Payment: 62.84% [NG] *
    Order-Status: 62.03% [NG] *
    Delivery: 80.89% [NG] *
    Stock-Level: 0.59% [NG] *

<TpmC>
10085.250 TpmC
```

TPC-C output gives 10085.25 TpmC. It means that 10085 New-Order transactions were processed per minute. Also, in the beginning (20~100 second), about 45~55 seconds were needed for 95% of transactions and 60~77 seconds were needed for 99 of transactions. Also max response time was from 99 to 123. However as benchmark goes on and at the end of benchmark (1110~1200 second), 60~66 seconds were needed for 95% of transactions and 87~106 seconds were needed for 99% of transactions. Also max response time was from 142 to 159.

[Figure 2] TPC-C Graph



At the beginning of benchmarking, r/s was 3550 and w/s was 1424, which is 71:29 ratio. At the end of benchmarking, r/s was 5227 and w/s was 2940, which is 64:36 ratio.

[Table 3] Read/Write Ratio

	Start	33%	66%	End
R/S	3550	5045	5015	5227
W/S	1424	2773	3127	2940
Ratio	71:29	65:35	61:39	64:36

In the CPU Status, about 25% time CPU was idle and the system did not have an outstanding disk I/O request. 15~35% of CPU utilization was occurred while executing at the user level (application) and 7~10% of CPU utilization was occurred while executing at the system level (kernel). 25~55% of iowait was occurred, which shows that the CPU was idle during which the system had an outstanding disk I/O request.

4. Conclusion

TPC-C can be used for MySQL benchmark tool. It shows the performance by TpmC. It also can show response time and transaction time with 95% and 99%. During benchmarking, read/write ratio was about 65:35, which shows that TPC-C has Random I/O intensive workload, approximately 65% Reads and 35% writes. Also, according to CPU status, CPU had to be idle for I/O request and when there is no I/O wait, CPU was idle for only 25% time. Also when CPU is not idle, CPU was utilized both in user level and system level.

5. REFERENCES

- [1] meejejin, “SWE3033-F2021”, Github repository,
<https://github.com/meejejin/SWE3033-F2021/>