

Database Project (SWE3033) (Fall 2023)

Homework #2 (100pts, Due date: 09/13)

Student ID: 2020315798

Student Name: Choi Jin Woo

1. [10pts] Write the system setup of your environment. Fill in the blank below.

Type	Specification
OS	Ubuntu 18.04
CPU	Intel(R) Core(TM) i5-8265U CPU @ 1.60GHz (2 Cores)
Memory (RAM)	6 GB
Kernel	Linux version 5.4.0-84-generic
Data Device (Optional)	
Log Device (Optional)	

2. [10pts] Write the benchmark setup of your environment. Fill in the blank below.

Type	Specification
DB Size	2GB
Buffer Pool Size	1GB
Benchmark Tool	tpcc-mysql
Runtime	1200s
Connections	8

3. [50pts] Assume that you have a database with a size of 10 warehouses. Determine the **optimal percentage of buffer pool size to maximize TPC-C benchmark performance**. Present experimental results and analyze the results yourself. You must include **1. chart of TpmC and hit ratio, 2. result of least three buffer pool sizes**. Explain the reason for the performance shift as the buffer size varies. (The buffer pool size should not exceed the database size.)

I have conducted 3 different buffer pool sizes: 256MB (25%), 512MB (50%), 768MB (75%)

1) 256 MB

Buffer Pool Hit Rate: 48.3%

TpmC: 2574.35

```
Jlhn011258@Jlhn011258: /usr/local/mysql
File Edit View Search Terminal Help
-----
BUFFER POOL AND MEMORY
-----
Total large memory allocated 824573952
Dictionary memory allocated 103406
Buffer pool size 49152
Free buffers 48864
Database pages 288
Old database pages 0
Modified db pages 0
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 0, not young 0
0.00 youngs/s, 0.00 non-youngs/s
Pages read 254, created 34, written 36
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
Buffer pool hit rate 483 / 1000, young-making rate 0 / 1000 not 0 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 288, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
```

<Raw Results>

```
[0] sc:12 lt:51475 rt:0 fl:0 avg_rt: 103.0 (5)
[1] sc:7969 lt:43511 rt:0 fl:0 avg_rt: 23.9 (5)
[2] sc:182 lt:4966 rt:0 fl:0 avg_rt: 26.6 (5)
[3] sc:272 lt:4875 rt:0 fl:0 avg_rt: 191.0 (80)
[4] sc:0 lt:5148 rt:0 fl:0 avg_rt: 393.2 (20)
in 1200 sec.
```

<Raw Results2(sum ver.)>

```
[0] sc:12 lt:51475 rt:0 fl:0
[1] sc:7969 lt:43514 rt:0 fl:0
[2] sc:182 lt:4966 rt:0 fl:0
[3] sc:272 lt:4875 rt:0 fl:0
[4] sc:0 lt:5148 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: 0.02% [NG] *
  Payment: 15.48% [NG] *
  Order-Status: 3.54% [NG] *
  Delivery: 5.28% [NG] *
  Stock-Level: 0.00% [NG] *
```

<TpmC>

2574.350 TpmC

2) 512 MB

Buffer Pool Hit Rate: 57.5%

TpmC: 3734.2

```
jlhn011258@jlhn011258: /usr/local/mysql
File Edit View Search Terminal Help
Last checkpoint at 2746995
0 pending log flushes, 0 pending chkp writes
10 log i/o's done, 0.10 log i/o's/second
-----
BUFFER POOL AND MEMORY
-----
Total large memory allocated 274857984
Dictionary memory allocated 103406
Buffer pool size 16384
Free buffers 16116
Database pages 268
Old database pages 0
Modified db pages 0
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 0, not young 0
0.00 young/s, 0.00 non-young/s
Pages read 234, created 34, written 36
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
Buffer pool hit rate 575 / 1000, young-making rate 0 / 1000 not 0 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.00/s
LRU len: 268, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
```

```
<Raw Results>
[0] sc:120 lt:74564 rt:0 fl:0 avg_rt: 73.8 (5)
[1] sc:18295 lt:56384 rt:0 fl:0 avg_rt: 19.3 (5)
[2] sc:860 lt:6608 rt:0 fl:0 avg_rt: 18.7 (5)
[3] sc:896 lt:6573 rt:0 fl:0 avg_rt: 164.8 (80)
[4] sc:0 lt:7471 rt:0 fl:0 avg_rt: 180.8 (20)
in 1200 sec.
```

```
<Raw Results2(sum ver.)>
[0] sc:120 lt:74564 rt:0 fl:0
[1] sc:18296 lt:56392 rt:0 fl:0
[2] sc:860 lt:6608 rt:0 fl:0
[3] sc:896 lt:6573 rt:0 fl:0
[4] sc:0 lt:7471 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: 0.16% [NG] *
    Payment: 24.50% [NG] *
    Order-Status: 11.52% [NG] *
    Delivery: 12.00% [NG] *
    Stock-Level: 0.00% [NG] *
```

```
<TpmC>
3734.200 TpmC
```

3) 768 MB

Buffer Pool Hit Rate: 78.8%

TpmC: 4202.7

```
Log sequence number 2746920
Log flushed up to 2746920
Pages flushed up to 2746920
Last checkpoint at 2746911
0 pending log flushes, 0 pending chkp writes
14 log i/o's done, 0.23 log i/o's/second
-----
BUFFER POOL AND MEMORY
-----
Total large memory allocated 1099431936
Dictionary memory allocated 141965
Buffer pool size 65536
Free buffers 65213
Database pages 323
Old database pages 0
Modified db pages 0
Pending reads 0
Pending writes: LRU 0, flush list 0, single page 0
Pages made young 0, not young 0
0.00 young/s, 0.00 non-young/s
Pages read 289, created 34, written 36
0.00 reads/s, 0.00 creates/s, 0.00 writes/s
Buffer pool hit rate 788 / 1000, young-making rate 0 / 1000 not 0 / 1000
Pages read ahead 0.00/s, evicted without access 0.00/s, Random read ahead 0.
LRU len: 323, unzip_LRU len: 0
I/O sum[0]:cur[0], unzip sum[0]:cur[0]
```

<Raw Results>

```
[0] sc:342 lt:83712 rt:0 fl:0 avg_rt: 64.8 (5)
[1] sc:25577 lt:58468 rt:0 fl:0 avg_rt: 17.1 (5)
[2] sc:2792 lt:5613 rt:0 fl:0 avg_rt: 11.2 (5)
[3] sc:2300 lt:6106 rt:0 fl:0 avg_rt: 140.0 (80)
[4] sc:0 lt:8407 rt:0 fl:0 avg_rt: 180.5 (20)
in 1200 sec.
```

<Raw Results2(sum ver.)>

```
[0] sc:342 lt:83712 rt:0 fl:0
[1] sc:25577 lt:58472 rt:0 fl:0
[2] sc:2792 lt:5613 rt:0 fl:0
[3] sc:2300 lt:6106 rt:0 fl:0
[4] sc:0 lt:8407 rt:0 fl:0
```

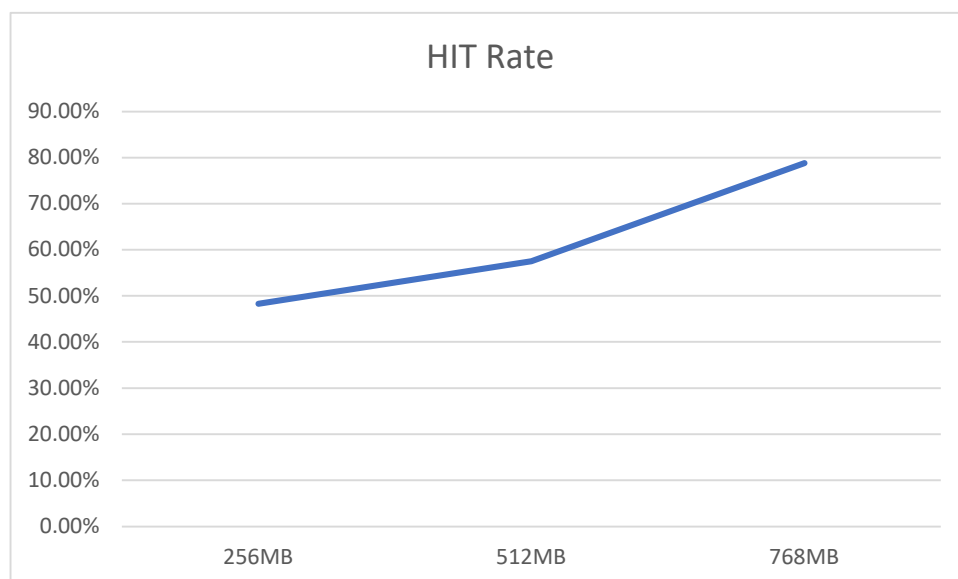
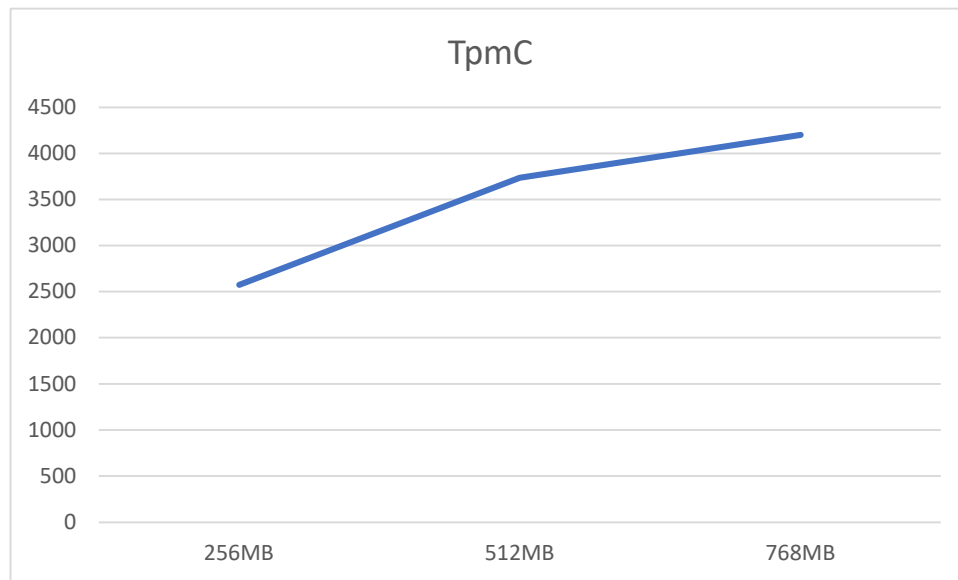
<Constraint Check> (all must be [OK])

```
[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: 0.41% [NG] *
  Payment: 30.43% [NG] *
  Order-Status: 33.22% [NG] *
  Delivery: 27.36% [NG] *
  Stock-Level: 0.00% [NG] *
```

<TpmC>

4202.700 TpmC

As a result, a chart goes as follows:



Conclusion:

Increasing the buffer pool size is expected to lead to an increase in TpmC (Transactions per Minute in the TPC-C benchmark) and the Buffer Hit Rate. As the allocated buffer size increases, the capacity to cache data grows, resulting in a higher hit rate. Consequently, this is likely to lead to an increase in the number of transactions processed per minute.

I couldn't attach the overall graph of the log files, because Gnuplot mentioned in the Github repository referred in the course slides does NOT work somehow. I followed the instructions and modified with 3 input text files from log files, however, it didn't work with even 2 inputs also.

4. [30pts] Why is it difficult to store the entire database in memory in real world, and why do we need to use buffer pool in database management system?

Storing an entire database in memory can be challenging for the following reasons.

- Databases tend to grow as more data is added. So it will eventually run out of memory.
- Many databases are massive in size, ranging from gigabytes to terabytes or even more.
- Storing the entire database in RAM makes it vulnerable to data loss in case of power outages or system crashes.
- RAM is more expensive than HDD or SSD. To store an entire database in RAM, it is required to invest significantly in high-capacity RAM modules, increasing the overall cost of the system.

As a result, with the above reasons, it is difficult to store the entire database in memory in real world.

While there are some difficulties, we use buffer pool for the following reasons.

- Most commonly accessed data is available in memory, providing fast access times.
- By caching data in memory, buffer pools reduce the need to read data from slower storage devices like hard drives or SSDs.
- It is cost-effective compared to the amount of RAM usage storing the entire database, since it uses only a smaller amount of RAM.
- Data is persistently stored on on-volatile storage, so ensures the durability.