Week#13 Buffer Management in SQLite

Hyuksoo Yeo

2016312761

1. INTRODUCTION

Run TPC-C benchmark by varying the cache size. For example, change cache size to 50, 100, 150, 200. Then observe how TPS (txn/s) changes. Record and analyze the TPS for each transaction (DELIVERY, NEW_ORDER, ORDER_STATUS, PAYMENT, STOCK_LEVEL). Lastly, present and analyze experimental results.

2. METHODS

For this experiment, install SQLite library and python. Because we will use pytpcc benchmark. Then setup pytpcc benchmark by preparing the SQLite configuration file and manipulate the path to the SQLite database. Then load TPC-C database and run to evaluating TPC-C benchmark. Repeating running by changing page cache size. We can change the size by editing the value in --buffer in command.

3. Performance Evaluation

3.1 Experimental Setup

System setup:

Type	Specification
OS	Ubuntu 20.04.3 LTS
CPU	Intel® Core™ i3-9100F CPU @ 3.60GHz
Memory	16GB
Kernel	5.11.0-27-generic
Data Device	Western Digital WD Blue 500GB
Log Device	Western Digital WD Blue 500GB

Benchmark setup:

Type	Configuration
DB size	1GB (10 warehouse)
Buffer Pool Size	300MB (30% of DB size)
Benchmark Tool	tpcc-mysql
Runtime	1200s
Connections	8

3.2 Experimental Results

Cache size = 50 (pages)

```
yhs@yhs-VirtualBox:~/SWE3033-F2021/week-13/pytpcc$ python tpcc.py --warehouse=10
 --config=./sqlite.config --no-load --duration=1800 --buffer=50 sqlite
3.31.1
journal mode delete
cache_size 50
11-28-2021 19:20:20 [<module>:240] INFO : Initializing TPC-C benchmark using Sql
iteDriver
11-28-2021 19:20:20 [execute:056] INFO : Executing benchmark for 1800 seconds
=====================================================================
Execution Results after 1800 seconds
---------------------------------------------------------------------
                Executed        Time (µs)       Rate
  DELIVERY      4875            203413285.494   23.97 txn/s
  NEW_ORDER     53806           1035843201.16   51.94 txn/s
  ORDER_STATUS  4921            8661961.07864   568.12 txn/s
  PAYMENT       51595           527723878.145   97.77 txn/s
  STOCK_LEVEL   4873            13237733.1257   368.11 txn/s
---------------------------------------------------------------------
  TOTAL         120070          1788880059.0    67.12 txn/s
```

Cache size = 100

```
yhs@yhs-VirtualBox:~/SWE3033-F2021/week-13/pytpcc$ python tpcc.py --warehouse=10
 --config=./sqlite.config --no-load --duration=1800 --buffer=100 sqlite
3.31.1
journal mode delete
cache_size 100
11-28-2021 19:56:06 [<module>:240] INFO : Initializing TPC-C benchmark using Sql
iteDriver
11-28-2021 19:56:06 [execute:056] INFO : Executing benchmark for 1800 seconds
=====================================================================
Execution Results after 1800 seconds
---------------------------------------------------------------------
                Executed        Time (µs)       Rate
  DELIVERY      5055            168905270.1     29.93 txn/s
  NEW_ORDER     56287           1066466877.7    52.78 txn/s
  ORDER_STATUS  4982            7875113.72566   632.63 txn/s
  PAYMENT       53622           532592760.801   100.68 txn/s
  STOCK_LEVEL   5045            12825190.5441   393.37 txn/s
---------------------------------------------------------------------
  TOTAL         124991          1788665212.87   69.88 txn/s
```

Cache size = 150

```
yhs@yhs-VirtualBox:~/SWE3033-F2021/week-13/pytpcc$ python tpcc.py --warehouse=10
 --config=./sqlite.config --no-load --duration=1800 --buffer=150 sqlite
3.31.1
journal mode delete
cache_size 150
11-28-2021 20:43:27 [<module>:240] INFO : Initializing TPC-C benchmark using Sql
iteDriver
11-28-2021 20:43:27 [execute:056] INFO : Executing benchmark for 1800 seconds
=====================================================================
Execution Results after 1800 seconds
---------------------------------------------------------------------
                Executed        Time (µs)       Rate
  DELIVERY      4803            169011243.343   28.42 txn/s
  NEW_ORDER     54590           1070619347.81   50.99 txn/s
  ORDER_STATUS  4876            6813793.42079   715.61 txn/s
  PAYMENT       52162           532573827.744   97.94 txn/s
  STOCK_LEVEL   4933            9299520.73097   530.46 txn/s
---------------------------------------------------------------------
  TOTAL         121364          1788317733.05   67.86 txn/s
```

Cache size = 200

```
yhs@yhs-VirtualBox:~/SWE3033-F2021/week-13/pytpcc$ python tpcc.py --warehouse=10
 --config=./sqlite.config --no-load --duration=1800 --buffer=200 sqlite
3.31.1
journal mode delete
cache_size 200
11-28-2021 21:18:06 [<module>:240] INFO : Initializing TPC-C benchmark using Sql
iteDriver
11-28-2021 21:18:06 [execute:056] INFO : Executing benchmark for 1800 seconds
===============================================================
Execution Results after 1800 seconds
---------------------------------------------------------------
                Executed        Time (µs)       Rate
  DELIVERY      4743            166993699.551   28.40 txn/s
  NEW_ORDER     52298           1085202093.36   48.19 txn/s
  ORDER_STATUS  4642            5009377.00272   926.66 txn/s
  PAYMENT       50097           525443260.431   95.34 txn/s
  STOCK_LEVEL   4666            4809072.49451   970.25 txn/s
---------------------------------------------------------------
  TOTAL         116446          1787457502.84   65.15 txn/s
```

	cache size (pages)	50	100	150	200
transaction					
DELIVERY		23.97	29.93	28.42	28.40
NEW_ORDER		51.94	52.78	50.99	48.19
ORDER_STATUS		568.12	632.63	715.61	926.66
PAYMENT		97.77	100.68	97.94	95.34
STOCK_LEVEL		368.11	393.37	530.46	970.25
TOTAL		67.12	69.88	67.86	65.15

In case of DELIVERY, NEW_ORDER, and PAYMENT, TPS is highest when cache size is 100 pages and decreases by increasing cache size after 100 pages. While the case of ORDER_STATUS and STOCK_LEVEL, TPS continuously goes up by increasing cache size.

TOTAL TPS is highest at cache size = 100.

4. Conclusion

In this experiment, I learned how to manage buffer in SQLite by using pytpcc benchmark. TPS is different at each transaction type and cache size.

5. REFERENCES

[1] https://github.com/meeeejin/SWE3033-F2021/tree/main/week-13