Week#14 Logging & Recovery in SQLite

Hyuksoo Yeo

2016312761

1. INTRODUCTION

Run TPC-C benchmark for two journal modes, delete mode and wal mode. Then observe how TPS (txn/s) changes. Record and analyze the TPS for each transaction type such as DELIVERY, NEW_ORDER, ORDER_STATUS, PAYMENT, STOCK_LEVEL. Then present and analyze the experimental results. I will explain the root cause of the performance gap between delete mode and wal mode.

2. METHODS

For this experiment, we should prepare setup like week13 experiment. Same as last experiment, loading database with 10 warehouses, and run TPC-C benchmark. In this week, we should input '–journal' in running command to change journal mode. Repeating running after change journal mode from del to wal.

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

Journal mode = wal

```
yhs@yhs-VirtualBox:~/SWE3033-F2021/week-13/pytpcc$ python tpcc.py --warehouse=10
 --config=./sqlite.config --no-load --duration=1800 --journal=wal sqlite
3.31.1
journal mode wal
cache_size 1024
11-30-2021 00:09:24 [<module>:240] INFO : Initializing TPC-C benchmark using Sql
iteDriver
11-30-2021 00:09:24 [execute:056] INFO : Executing benchmark for 1800 seconds
===============================================================
Execution Results after 1800 seconds
---------------------------------------------------------------
                Executed        Time (µs)       Rate
  DELIVERY      10033           199642950.296   50.25 txn/s
  NEW_ORDER     111763          1136385735.99   98.35 txn/s
  ORDER_STATUS  10004           15705872.0589   636.96 txn/s
  PAYMENT       107536          408742728.472   263.09 txn/s
  STOCK_LEVEL   9970            16320948.8392   610.87 txn/s
---------------------------------------------------------------
  TOTAL         249306          1776798235.65   140.31 txn/s
```

Journal mode = del

```
yhs@yhs-VirtualBox:~/SWE3033-F2021/week-13/pytpcc$ python tpcc.py --warehouse=10
 --config=./sqlite.config --no-load --duration=1800 --journal=del sqlite
3.31.1
journal mode del
cache_size 1024
11-30-2021 00:40:34 [<module>:240] INFO : Initializing TPC-C benchmark using Sql
iteDriver
11-30-2021 00:40:34 [execute:056] INFO : Executing benchmark for 1800 seconds
===============================================================
Execution Results after 1800 seconds
---------------------------------------------------------------
                Executed        Time (µs)       Rate
  DELIVERY      4803            145904710.531   32.92 txn/s
  NEW_ORDER     54443           1084508245.47   50.20 txn/s
  ORDER_STATUS  4868            10221263.8855   476.26 txn/s
  PAYMENT       51695           536710806.608   96.32 txn/s
  STOCK_LEVEL   4867            11552169.323    421.31 txn/s
---------------------------------------------------------------
  TOTAL         120676          1788897195.82   67.46 txn/s
```

The total TPS is 140.31, and 67.46 txn/s each. Wal journal mode shows much more higher performance than del mode. It is because write ahead logging is significantly faster in most scenarios. Disk I/O operations tends to be more sequential when using WAL mode. Because WAL uses fewer fsync() operations in transaction. While deleting file uses fsync() in flushing RBJ. Deleting file is expensive.

4. Conclusion

In this experiment, I learned how to evaluate performance between two journal mode (RBJ and WAL) on SQLite database engine using TPC-C benchmark. (pytpcc). Wal mode is much faster than del mode in SQLite.

5. REFERENCES

[1] https://github.com/meeeejin/SWE3033-F2021/tree/main/week-14