PA#2 Space Amplification in DBMS

Siyeol Choi
2019315530

1. INTRODUCTION

In this lab experiment, we target to measure the space amplification of MySQL and RocksDB. Also, we would figure out the way to optimization techniques to mitigate space amplification of RocksDB.

2. METHODS

Space amplification can be calculated by dividing DB size on file system / actual user data size. For the MySQL experiment we install innodb_ruby for benchmarking. Then we calculate user data size in B+- tree nodes after the tpc-c benchmark is finished. Then with the computed tpcc DB size and user data size we calculate the space amplification of MySQL.

Net, we can calculate the rocksdb space amplification by running db_bench. We can specify the max bytes for level base and level multipliers and compaction style depending on the hardware constraints and environment. After running the benchmark, we can calculate the space amplification by adding the L1 and L2 size and dividing my L2 size.

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 - MySQL]

Type	Configuration
DB size	2GB (20 warehouse)
Buffer Pool Size	100MB (5% of DB size)
Benchmark Tool	tpcc-mysql
Runtime	1200s
Connections	8

[Benchmark setup - RockDB]

RocksDB: version 7.8.0

CPU: 8 * Intel(R) Core(TM) i5-8250U CPU

@ 1.60GHz

CPUCache: 6144 KB

Keys: 16 bytes each (+ 0 bytes user-defined timestamp)

Values: 100 bytes each (50 bytes after compression)

Entries: 1000000

Prefix: 0 bytes

Keys per prefix: 0

RawSize: 110.6 MB (estimated)

FileSize: 62.9 MB (estimated)

Compression: Snappy

Memtablerep: SkipListFactory

Perf Level: 1

max_bytes_for_level_multiplier= 5

max_bytes_for_level_base=33554432

target_file_size_base=2097152

write_buffer_size=2097152

runtime = 7200

3.2 Experimental Results

[MySQL]

Total Pages = 128792

Total DB size = 128792*16384 = 2110128128

Total Data Size = 1594782394

Space Amplification = DB size / Data Size = 1.32

[RocksDB]

Level Files Size Score Read(GB) Rn(GB) Rnp1(GB) Write(GB) Wnew(GB) Moved(GB) W-Amp Rd(MB/s) Wr(MB/s) Comp(sec) CompMergeCPU(sec) Comp(cnt) Avg(sec) KeyIn KeyDrop Rblob(GB) Wblob(GB)

L0 1/0 930.08 KB 0.2 0.0 0.0 0.0 0.1 0.1 0.0 1.0 0.0 39.3 3.72 1.70 161 0.023 0 0 0.0 0.0

L1 10/0 29.23 MB 0.9 1.1 0.1 0.9 1.0 0.1 0.0 7.2 59.3 56.2 18.52 13.23 40 0.463 17M 858K 0.0 0.0

L2 21/0 47.80 MB 0.3 0.0 0.0 0.0 0.0 0.0 0.0 1.3 60.4 39.9 0.55 0.30 5 0.111 520K 160K 0.0 0.0

(L1 size + L2 size) / L2 size = (29.23 + 47.80) / 47.80 = 1.61

Space amplification for RocksDB = 4.43

4. Mitigate Space Amplification

Two strategies can be used to enhance space amplification of RocksDB. First, it can adapt the dynamic level size to the size of the data. Also, it can be done by applying a number of compression strategies.

If the size of each level to be 10% of the size of the data on the next level, the space amplification can be reduced to less than 1.1111… Plus when the level size multiplier is larger, the space and read amplification can be benefitted more by sacrificing the write amplification. The multiplier can vary among each level for the truly optimal solution.

On top of that, Compressing SST files can be applied. Multiple solutions can be applied simultaneously, including key prefix encoding which prevents repeated write of prefixes of previous keys and sequence ID garbage collection which removes old snapshot IDs. Also, data compression and dictionary-based compression can be beneficial. Data compression applied on per-block basis reduces the size and dictionary-based compression is advantageous when small data blocks are dominantly used.

5. Conclusion

Space Amplification of RocksDB is better than the space amplification of MySQL since RocksDB uses LSM tree. Plus there are multiple methods to even more improve RocksDB's space amplification. We can mitigate Space Amplification by adapting dynamic level size and various compression strategies of SST files.

6. REFERENCES

[1] RocksDB measurement method https://github.com/LeeBohyun/RocksDB/blob/main/measure_SAF.md

[2] MySQL measurement method https://github.com/LeeBohyun/mysql-tpcc/blob/master/innodb_b%2Btree/measure_space_utilization.md

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

[4] . Dong, Siying, Mark D. Callaghan, Leonidas Galanis, DhrubaBorthakur, T. Savor and Michael Strum. "Optimizing Space Amplification in RocksDB." CIDR (2017)