Week 10: SQLite and Androbench Benchmark

Siyeol Choi
2019315530

1. INTRODUCTION

In this lab experiment, we aim to experiment with SQLite and Androbench by varying Journal Mode, Page Cache Size, Page size, Locking Mode, Synchronous Mode. Also, we will be comparing SQLite with other client-server RDBMS and how it supports cross-platform database and its comparison with Filesystem.

2. METHODS

After building SQLite, we ran "time ./sqlite3 /{PATH}/androbench.db < androbench.sql &> /dev/null" Command to record time and analyze the SQLite trace file. Then, by modifying the PRAGMA command in androbench.sql, we experimented diverse cases. Values ranged from [page_cache size : 100, 500, 1000, 2000], [journal mode : off, delete, and wal mode], [page_size : 512, 1024, 2048, 4096, 8192, 16384], [locking mode : NORMAL, EXCLUSIVE], [synchronous mode : 0, 1, 2].

3. SQLite Analysis

[SQLite versus client-server RDBMS (i.e. MySQL)]

SQLite is an embedded, file-based relational DBMS which serverless enabling cross-platform compatible. While MySQL is a client-server RDBMS consisted of multi-threaded SQL server.

- Advantage of SQLite: File based; Standards-aware; Fit for developing and testing; Adequate for Embedded applications and disk access replacement.

- Disadvantage of SQLite: Not adequate for Multi-user applications due to lack of user management; Not good for apps requiring high right volumes; Hard to tune performance

[How SQLite supports cross-platfrom database]

VFS(OS portability layer) is invoked when any of the other modules in SQLite needs to communicate with the OS. Then it invokes the operating-specific code needed to satisfy the request. Accordingly, porting SQLite to cross-platform is resolved by simply writing a new OS interface layer.

[SQLite versus Filesystem]

SQLite is faster than Filesystem when it comes to r/w of small blobs. This is because in the SQLite database, the open() and close() system calls are invoked only once. Plus, the overhead of open() and close() syscalls is greater than the overhead of using the SQLite database.

Performance Evaluation

3.1 Experimental Setup

[System Setup]

** Google Colab **

CPU model	Intel(R) Xeon(R) CPU @ 2.20GHz
OS	Ubuntu 18.04.6 LTS
MemTotal:	13297228 kB
CPU cores	1
CPU threads:	2
Kernel	Linux 5.10.133+ x86_64

3.2 Experimental Results

Journal mode		page cache size		page size	
off	0m0.008s	100	0m0.005s	512	0m0.005s
delete	0m0.005s	500	0m0.005s	1024	0m0.007s
wal	0m0.013s	1000	0m0.006s	2048	0m0.007s
		2000	0m0.008s	4096	0m0.007s
				8192	0m0.011s
				16384	0m0.007s
locking mode		synchronous mode			
normal	0m0.006s	0	0m0.008s		
exclusive	0m0.007s	1	0m0.007s		
		2	0m0.005s		

4. 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.

5. REFERENCES

[1] SQLite-Androbench measurement method https://github.com/meeeejin/SWE3033-F2021/tree/main/week-10

[2] SQLite Pragma https://www.sqlite.org/pragma.html

[3] SQLite vfs https://www.sqlite.org/vfs.html